Xi Software

GNUbatch Release 1

System Reference Manual

Table of Contents

1 Introduction.....1
1.1 Typographical Conventions.....1
1.2 Command Line Program Options.....1
2 Free Documentation Licence.....3
3 Overview.....11
3.1 Directory and File Structure.....12
3.1.1 Internal Directories.....12
3.1.2 Internal Programs.....12
3.1.3 Batch directory files.....13
3.1.4 Help and Message files.....13
3.1.5 Configuration files held in /usr/local/etc.....14
3.1.5.1 GNUbatch Hosts File.....14
3.1.5.1.1 Multiple IP addresses.....15
3.1.5.2 GNUbatch Master Configuration File.....16
3.1.5.3 User Mapping file.....16
3.1.5.4 GNUbatch Static Environment File.....16
3.2 Job and Variable Modes.....17
3.2.1 Change of owner and group.....17
3.2.2 Initialisation of modes.....18
3.3 Standard Exit Codes.....18
4 User Administration.....20
4.1 Privileges.....20
4.2 Priorities.....21
4.3 Charging.....22
4.4 Modes.....22
5 Job control variables.....23
5.1 Dependency on files.....24
5.2 What's in a Variable?.....24
5.3 More about Modes.....25
5.4 Examples of Dependencies Handled by Variables.....26
5.4.1 Running Jobs in a Simple Chain.....26
5.4.2 Running jobs in a chain with exception handling.....28
5.4.3 Running Jobs in Parallel.....28
5.4.4 The Parallel Example with an Exception Handler.....29
5.4.5 Mutual Exclusion & Semaphores.....30
5.4.6 Passing Data between Jobs.....31
5.5 System variables and logging.....32
5.5.1 Using CLOAD & LOADLEVEL.....33
5.5.1.1 Running fewer batch jobs in office hours.....33
5.5.1.2 Stopping GNUbatch gracefully.....33
5.5.1.3 Starting Administration activities, when Batch work completes.....34
5.5.2 Controlling peak activity.....34
5.5.3 Job Logging via LOGJOBS.....34
5.5.4 Variable Logging via LOGVARS.....35
6 Jobs and related entities.....37
6.1 Time.....37
6.1.1 Scheduled run start time.....37
6.1.2 Retention.....38
6.1.2.1 Auto delete after execution.....38
6.1.3 Repetition.....38
6.1.3.1 Monthly Repeat Intervals.....39
6.1.3.2 Days to Avoid.....40
6.1.3.3 Time adjustments on error.....40
6.2 Job Completion Messages.....40

- 6.3 Redirection of Input and Output....41
- 6.4 Arguments....41
- 6.5 Environment & process parameters....42
  - 6.5.1 Environment Variables....42
  - 6.5.2 ulimit and umask....42
  - 6.5.3 Working Directory....43
  - 6.5.4 Normal and Error Exit Codes....43
  - 6.5.5 Network Scope....43
  - 6.5.6 Time-out parameters for stopping Runaway Jobs....44
- 6.6 Owners, Groups and Modes....44
  - 6.6.1 Owners and Groups....44
  - 6.6.2 Modes....44
- 6.7 Job Identifiers - Queues, Titles and Job ID numbers....45
- 6.8 Priority and Command Interpreter....46
- 6.9 Job control variables - Conditions and Assignments....46
  - 6.9.1 Conditions....46
  - 6.9.2 Assignments....47
    - 6.9.2.1 Flag Options....47
    - 6.9.2.2 Assignment Operation....48
- 6.10 Meta-Data....49
- 6.11 Command Interpreters....50
- 6.12 Queues....51
  - 6.12.1 Examples....52
    - 6.12.1.1 Naming Conventions for Overlapping Sets....52
    - 6.12.1.2 Naming Conventions for Sub-Queues or Hierarchies....53
- 6.13 Holidays....53

7 Internal Programs and file formats....54
- 7.1 Core Programs....54
  - 7.1.1 btsched....54
    - 7.1.1.1 Files used....54
    - 7.1.1.2 IPC Facilities used....55
    - 7.1.1.3 Internet ports used....55
  - 7.1.2 xbnetserv....55
    - 7.1.2.1 Internet ports....56
    - 7.1.2.2 Diagnostics....56
- 7.2 btexec....56
- 7.3 Bgtksave....56
- 7.4 Bgtkldsv....56
- 7.5 Message Handlers....57
  - 7.5.1 btmdisp....57
    - 7.5.1.1 Notes....57
  - 7.5.2 btwrite....57
    - 7.5.2.1 Notes....58
  - 7.5.3 dosbtwrite....58
    - 7.5.3.1 Notes....58
  - 7.5.4 Jobdump....58
- 7.6 File Formats....58
  - 7.6.1 /usr/local/etc/gnubatch.conf....58
  - 7.6.2 /usr/local/etc/gnubatch.hosts....59
    - 7.6.2.1 Host name....59
    - 7.6.2.2 Alias name....59
    - 7.6.2.3 Flags....59
    - 7.6.2.4 Timeout....60
    - 7.6.2.5 Local address....60

8 User Programs....61
- 8.1 Syntax of batch commands....61
  - 8.1.1 Option types....61

8.1.2 Option syntax....61
8.1.3 Configuration files....62
8.1.4 Option path....63
8.1.5 Message files....63
8.1.6 Location of message files....64
8.1.7 Path to locate message files....65
8.2 Submitting Batch Jobs....65
8.2.1 gbch-r and gbch-rr....65
8.2.1.1 Options....65
8.2.1.2 Redirection format....71
8.2.1.3 Repeat periods....73
8.2.1.4 Conditions....73
8.2.1.5 Assignments....74
8.2.1.6 Mode arguments....75
8.2.2 atcover....76
8.2.3 Gbch-xr and gbch-xmr....76
8.3 Managing the batch scheduler....76
8.3.1 gbch-start....76
8.3.1.1 Options....77
8.3.2 gbch-quit....77
8.3.2.1 Diagnostics....77
8.3.3 gbch-conn....78
8.3.4 gbch-disconn....78
8.3.5 gbch-cichange....78
8.3.5.1 Options....78
8.3.5.2 Examples....80
8.3.6 gbch-cilist....80
8.3.6.1 Options....80
8.3.7 gbch-hols....80
8.3.7.1 Options....81
8.3.8 gbch-hostedit....81
8.3.8.1 Options....82
8.3.8.2 Commands....82
8.4 Querying/managing batch jobs from the command line....82
8.4.1 gbch-jchange....82
8.4.1.1 Options....82
8.4.1.2 Mode arguments....86
8.4.1.3 Note on mode and owner changes....87
8.4.2 gbch-jdel....87
8.4.2.1 Options....88
8.4.2.2 Examples....89
8.4.3 gbch-jlist....89
8.4.3.1 Options....89
8.4.3.2 Wild cards....91
8.4.3.3 Format codes....91
8.4.3.4 Examples....93
8.4.4 gbch-jstat....93
8.4.4.1 Options....94
8.4.4.2 Example....94
8.4.5 gbch-jgo, gbch-jgoadv, gbch-jadv....95
8.4.6 gbch-dst....95
8.5 Querying/managing variables from the command line....96
8.5.1 gbch-vlist....96
8.5.1.1 Options....97
8.5.1.2 Format codes....97
8.5.2 gbch-var....98
8.5.2.1 Options....98

8.5.2.2 Conditions....99
8.5.2.3 Use of options....100
8.5.2.4 Note on mode and owner changes....100
8.6 Interactive job and variable administration....100
8.6.1 gbch-q....100
8.6.1.1 Options....101
8.6.2 Gbch-xq and gbch-xmq....102
8.7 File Monitoring....102
8.7.1 gbch-filemon....102
8.7.1.1 Options....103
8.7.1.2 File matching....105
8.7.1.3 Criteria....105
8.7.1.4 Pre-existing files....106
8.7.1.5 Recursive searches....106
8.7.1.6 Examples....106
8.7.2 gbch-xfilemon and gbch-xmfilemon....107
8.8 User administration and charging....107
8.8.1 gbch-charge....107
8.8.1.1 Options....107
8.8.1.2 Examples....108
8.8.2 gbch-uchange....108
8.8.2.1 Options....109
8.8.2.2 Users or default....110
8.8.2.3 Rebuilding the user control file....110
8.8.2.4 Dumping the password file....110
8.8.2.5 Privileges....111
8.8.2.6 Mode arguments....111
8.8.3 gbch-ulist....112
8.8.3.1 Options....112
8.8.3.2 Format argument....113
8.8.3.3 Privileges format....114
8.8.3.4 Modes....114
8.8.4 gbch-user....114
8.8.4.1 Options....115
8.8.5 gbch-xmuser....116
8.9 Web browser interface support....116
8.10 File management....116
8.10.1 gbch-uconv....116
8.10.1.1 Options....117
8.10.2 gbch-cjlist....117
8.10.2.1 Options....118
8.10.3 gbch-cvlist....118
8.10.3.1 Options....119
8.10.4 gbch-ciconv....119
8.10.4.1 Options....120
8.10.5 gbch-ripc....120
8.10.5.1 Options....121
8.10.5.2 Example....121
8.10.6 gbpasswd....122
8.10.6.1 Options....122
9 Text screen-based programs....123
9.1 gbch-q - interactive batch queue manager....123
9.1.1 User Interface Settings on Entry....123
9.1.2 General gbch-q screen commands....123
9.1.2.1 Context Sensitive Help....124
9.1.2.2 Macro Commands....124
9.1.3 The Job Screen and Commands....125

- 9.1.3.1 View job script....127
- 9.1.3.2 Change title....127
- 9.1.3.3 Time Specification....127
  - 9.1.3.3.1 Turn on or off time constraint....128
  - 9.1.3.3.2 Editing the time....128
  - 9.1.3.3.3 Editing the repetition factor....128
  - 9.1.3.3.4 Days to avoid....129
  - 9.1.3.3.5 Reschedule options....129
- 9.1.3.4 Change priority....129
- 9.1.3.5 Change load level....129
- 9.1.3.6 Progress Code....129
- 9.1.3.7 Force to run....130
- 9.1.3.8 Kill or cancel job....130
- 9.1.3.9 Process Parameters....130
- 9.1.3.10 Change command interpreter....132
- 9.1.3.11 Unqueue Job....133
- 9.1.3.12 Set mail/write message on job completion flags....134
- 9.1.3.13 Setting job arguments....134
- 9.1.3.14 Editing the environment....135
- 9.1.3.15 Editing redirections....136
- 9.1.3.16 Job Assignment Editing....136
  - 9.1.3.16.1 Choosing the Variable....137
  - 9.1.3.16.2 Specifying the Value to be Assigned....137
  - 9.1.3.16.3 Specifying the Assignment Operation....137
  - 9.1.3.16.4 Setting the flags....138
- 9.1.3.17 Set job conditions....138
  - 9.1.3.17.1 Choosing the Variable....138
  - 9.1.3.17.2 Specifying the Comparison Operator....139
  - 9.1.3.17.3 Specifying the Value to Compare Against....139
- 9.1.3.18 Change Owner....139
- 9.1.3.19 Change Group....139
- 9.1.3.20 Mode Editing....140
- 9.1.4 The Variables Screen and Commands....141
  - 9.1.4.1 Assign new value....142
  - 9.1.4.2 Arithmetic operations....142
  - 9.1.4.3 Change comment....142
  - 9.1.4.4 Create new variable....143
  - 9.1.4.5 Rename variable....143
  - 9.1.4.6 Mode Editing....144
  - 9.1.4.7 Change Owner....144
  - 9.1.4.8 Change of group....145
- 9.1.5 Command interpreter list....145
  - 9.1.5.1 Setting Up A Command Interpreter....146
- 9.1.6 Edit holiday list....146
- 9.1.7 Setting program options....147
- 9.1.8 Setting Display Contents....148
  - 9.1.8.1 Display Format for the Jobs Screen....148
  - 9.1.8.2 Display Format for the Variables Screen....149
  - 9.1.8.3 Editing the Display Formats....150
- 9.2 gbch-user - Interactive user administration tool....151
  - 9.2.1 Display current permissions....152
  - 9.2.2 Mode Edit....152
  - 9.2.3 View and edit permissions....153
    - 9.2.3.1 Setting user priorities....155
    - 9.2.3.2 Setting default priorities....155
    - 9.2.3.3 Setting user load levels....156
    - 9.2.3.4 Setting default load levels....156

9.2.3.5 Applying default settings to one or all users....156
9.2.3.6 Displaying charge....156
9.2.3.7 Setting user's privileges....156
9.2.3.8 Setting default privileges....157
9.2.3.9 Setting default and user modes....157
10 X Programs....159
10.1 gbch-xq – GTK GUI Batch Queue Tool....159
10.1.1 File Menu....160
10.1.1.1 View options menu....160
10.1.2 Action menu....161
10.1.3 Job Menu....161
10.1.3.1 View job script....162
10.1.3.2 Set job times....162
10.1.3.3 Title / Priority Load Level....163
10.1.3.4 Process parameters....164
10.1.3.5 Time Limits....165
10.1.3.6 Mail / Write....166
10.1.3.7 Job Permissions....167
10.1.3.8 Job Owner and Group....168
10.1.3.9 Job Arguments....168
10.1.3.10 Environment variables....169
10.1.3.11 Redirections....170
10.1.4 Create menu....170
10.1.4.1 Command Interpreters....170
10.1.4.2 Holidays....171
10.1.5 Delete menu....172
10.1.6 Conditions menu....172
10.1.6.1 Conditions list....173
10.1.6.2 Assignments list....173
10.1.7 Variables menu....174
10.1.8 Search menu....174
10.1.9 Macro editing....175
10.2 gbch-xmq - Optional Motif GUI Batch Queue Tool....177
10.2.1 Options....177
10.2.2 Useful Resources....177
10.2.3 Examples....178
10.2.4 The Main Window....179
10.2.5 The Menus and Shortcut Buttons....181
10.2.5.1 The Options Menu....181
10.2.5.2 The Action Menu & Buttons....181
10.2.5.3 The Jobs Menu & Buttons....182
10.2.5.4 The Create Menu....183
10.2.5.5 The Delete Menu....183
10.2.5.6 The Condition Menu....184
10.2.5.7 The Variable Menu....184
10.2.5.8 The Search Menu....185
10.2.5.9 The Jobmacro Menu....185
10.2.5.10 The Varmacro Menu....186
10.2.5.11 Help....186
10.2.6 Setting the View Options....186
10.2.6.1 Setting the Confirmation level....187
10.2.6.2 Restricting the display....187
10.2.6.2.1 Restricting the display to the local host....187
10.2.6.2.2 Restricting the display by job queue....187
10.2.6.2.3 Restricting the display by user & group....188
10.2.6.3 Changing the fields displayed and their format....188
10.2.6.3.1 Changing the Job Display....189

10.2.6.3.2 Changing the Variable Display....191
10.2.6.4 Saving the Format Changes....191
10.2.6.5 Saving the View Options (excluding Format Changes)....191
10.2.7 Viewing a Batch Job....192
10.2.8 Changing Job and Variable parameters....193
10.3 gbch-xr and gbch-xmr - Job Submission & Editing Tool....194
10.3.1 Options....195
10.3.2 Useful Resources....195
10.3.3 Examples....195
10.3.4 The Main Window....195
10.3.5 The Menus and Shortcut Buttons....197
10.3.5.1 The Options Menu....197
10.3.5.2 The Defaults Menu....198
10.3.5.3 The File Menu & Buttons....199
10.3.5.4 The Jobs Menu and Buttons....199
10.3.5.5 Help....200
10.3.6 Choosing a Directory....200
10.3.7 Creating a New Job....201
10.3.8 Loading an Unqueued or Previously Saved Job....202
10.3.9 Setting up or Editing the Job Specification....202
10.3.10 Editing the Job Script....202
10.3.11 Selecting a different Text Editor....202
10.3.12 Submitting Jobs....202
10.3.13 Saving, Closing and Deleting Jobs....202
10.3.14 Specifying Defaults....203
10.4 Gbch-xuser, gbch-xmuser - GUI User Administration....203
10.4.1 Options....203
10.4.2 Useful Resources....204
10.4.3 Examples....204
10.4.4 The Main Window....204
10.4.5 The Menus and Options....205
10.4.5.1 The Options Menu....205
10.4.5.2 The Defaults Menu....206
10.4.5.3 The Users Menu....206
10.4.5.4 The Charges Menu....207
10.4.5.5 The Usermacro Menu....207
10.4.5.6 The Help Menu....207
10.4.6 Selecting multiple users for menu options....208
10.4.7 Copying defaults to all users....208
10.4.8 Resetting a user to the default....208
10.5 gbch-xfilemon and gbch-xmfilemon - Interfaces to gbch-filemon....208
10.5.1 Useful Resources in gbch-xmfilemon....208
11 Configurability....210
11.1 Configuration files and environment variables....211
11.1.1 Environment Variables....211
11.1.2 Configuration files....212
11.1.3 Environment variable or keyword names....212
11.2 User reconfiguration....213
11.2.1 Message files....214
11.2.2 File format....215
11.2.2.1 Key definitions....216
11.2.2.1.1 Special key sequences....218
11.2.2.1.2 Help and error messages....219
11.2.2.1.3 Option syntax....220
11.2.2.1.4 Alternatives....221
11.2.2.1.5 Prompts....222
11.2.2.1.6 Numeric parameters....222

11.2.2.1.7 Titles....222
11.2.2.1.8 Enhancements and line drawing in headers....223
11.2.2.2 Changing message files....224
11.2.3 Environment variables....225
11.3 Variation of search order....225
11.3.1 Search order for message files....226
11.3.2 Search order for program options....226
11.3.3 Freezing options....227
12 Extensibility....228
12.1 Message Handling....228
12.1.1 Customising the system message file....228
12.1.2 Specifying a customised message file....228
12.1.3 Specifying alternative job completion - system wide....229
12.1.4 Specifying alternative job completion programs - per user....229
12.2 Command Interpreters....229
12.3 Custom Tools & Scripts....230
12.3.1 Custom Tools....230
12.3.2 Shell Scripts....231
12.4 Macros for Interactive User Programs....232
12.4.1 Inserting the commands....233
12.4.2 Menu Options in the Motif Programs....233
12.4.3 Binding the keys in gbch-q and gbch-user....234
12.4.4 Example - Adding the "cancel all jobs in queue" to gbch-q....234
12.5 File & Event Monitoring....235
12.5.1 Polling for Arrival of a File....236
12.5.2 Continuous Polling for a constantly changing list of Files....236

1 Introduction

GNUbatch is a fully functioned, high performance Job Scheduler and Management System which is available for a wide range of machines running a Unix or GNU/Linux Operating System. This manual provides the System Reference Information for all of the Unix platforms on which GNUbatch may be run, covering the basic product, shell and "curses" interfaces and the Motif Interface.

Separate manuals discuss the NT Agent, the Windows Interface, the Web Browser Interface and the API.

The original version was written by John Collins at Xi Software Ltd between 1990 and 2009 as “Xi-Batch” and GNUbatch is based on Release 6 of Xi-Batch. The names, including most of the program and service names, have been changed to GNUbatch or derivatives and the installation directories have been changed to conform to GNU standards. (For this reason some of the diagrams where there is no other need to redraw them may refer to Xi-Batch rather than GNUbatch).

1.1 Typographical Conventions

These manuals use various character fonts to indicate different types of information as follows:

`File names and quotations within the text`
`Examples and user script`
`Generic data (where you should put a value appropriate to your own environment)`
Program names, whether for Xi-Text or standard Unix facilities
`Warnings and important advice`

1.2 Command Line Program Options

Almost all of the programs that make up GNUbatch can take (or require) options and arguments supplied on the command line. As much flexibility as possible is allowed in the specification of these options and arguments. The examples in the manual use which ever notation is clearest.

White space may be inserted into flag arguments as in

```
gbch-r -c COUNT=0 -T 10:16
```

or it may be left out as in

```
gbch-r -cCOUNT=0 -T10:16
```

Single character options may be strung together with one minus sign:

```
gbch-r -mwC
```

or separated, as in

```
gbch-r -m -w -C
```

If mutually contradictory arguments are permitted, the rightmost (or rather the most recently specified) applies.

The ability to redefine option letters has been provided, together with the `+keyword` or `--keyword` style of option. Such options should be given completely surrounded by spaces or tabs to separate them from each other and their arguments, for example

```
gbch-r --condition COUNT=0 --time 10:16
```

In addition, all the commands have an option `-?` or *`+explain`* (or `--explain`) whose function is to list all the other options and exit.

There is a mechanism for picking up options from environment variables or so-called configuration files called `.gnubatch` containing the relevant keyword.

2 Free Documentation Licence

GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright ©2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. www.fsf.org

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent.

An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition.

Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice.

These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit.

When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See www.gnu.org/copyleft.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME.

> Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
>
> A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

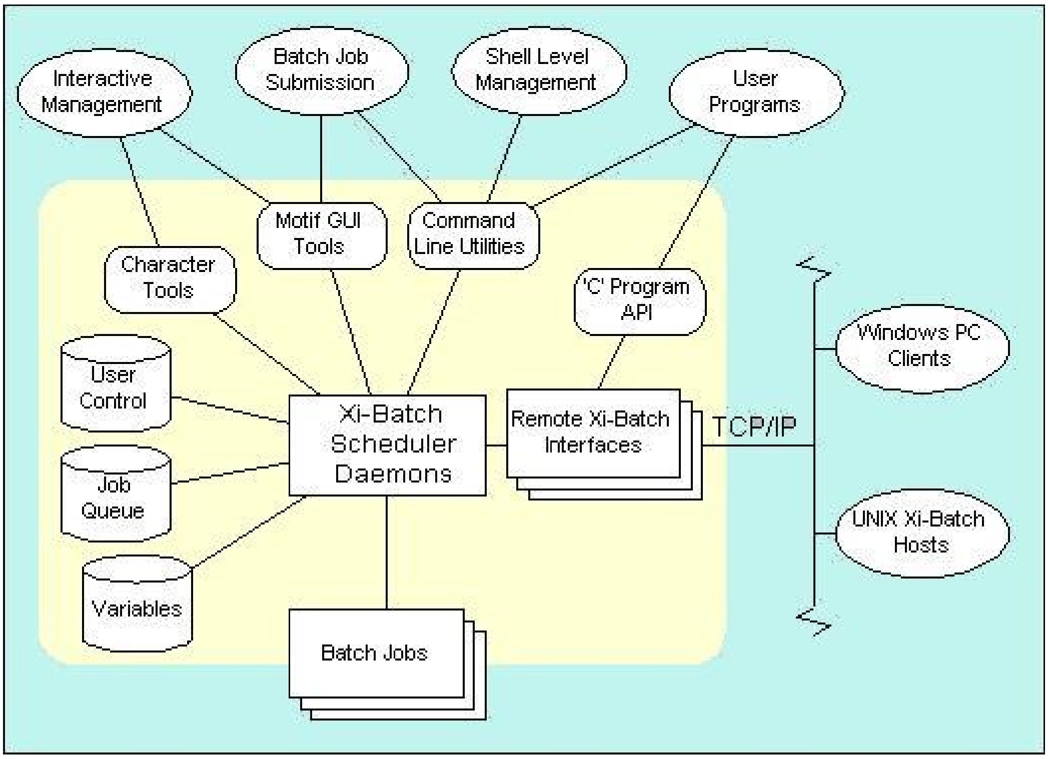
> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

3 Overview

GNUbatch can run on a single Unix host or several co-operating machines. The central white area of the block diagram shows the possible components of GNUbatch on a Unix machine. The shaded area indicates the entities, outside of that machine's GNUbatch system boundary, which use or provide services to it.


Interactive Management
Batch Job Submission
Shell Level Management
User Programs
Motif GUI Tools
Command Line Utilities
Character Tools
'C' Program API
Windows PC Clients
User Control
Xi-Batch Scheduler Daemons
Remote Xi-Batch Interfaces
TCP/IP
Job Queue
UNIX Xi-Batch Hosts
Variables
Batch Jobs

At the heart of GNUbatch is the scheduler daemon btsched. This daemon manages the batch jobs and the job control variables, which are used for handling dependencies. There are two instances of btsched running on a stand alone system, or three if co-operating with other GNUbatch hosts. Co-operating GNUbatch hosts require connection via a network that provides TCP/IP services.

btsched maintains the job and variable shared memory segments, periodically writing them out to file when changes are made. It forks to provide one process to monitor running processes and one to accept messages on the message queue. It forks again to provide the third btsched to handle the network interface if this mode of operation is used.

When an interactive queue management tool (e.g. gbch-q, gbch-xq or gbch-xmq) process is started, it arranges with btsched to be sent signals to advise it of changes in the job queue or variable list.

All requests, by gbch-q and other processes, are dealt with by sending a message on the message queue and receiving replies on the same message queue.

IPC used by GNUbatch

GNUbatch uses one message queue to communicate with the scheduler process, btsched. Two shared memory segments are used to hold records of jobs and variables. These records are periodically written out to the files `btsched_jfile` and `btsched_vfile` respectively. A further shared memory segment is used as buffer space for passing job details, as the size of messages which may be sent on message queues is limited on many systems. One group of semaphores controls access to the shared memory segments, and another group is used for network locking.

The IPC facilities can be recognised by running ipcs. The items in question are owned by `batch` with a key of `0x5869Bxxx`.

3.1 Directory and File Structure

The files which comprise GNUbatch are held in various directories depending upon their nature. With the exception of global configuration files the installation can be tailored to suit local practices and standards.

- Global configuration files are always held in the `/usr/local/etc` directory.
- User programs can be placed in any directory which is on the GNUspool users' `PATH` variable (if built with the default options these are `/usr/local/bin` and `/usr/local/sbin`).
- Internal programs and data are held in three or sometimes four separate directories.

3.1.1 Internal Directories

GNUbatch uses various directories to hold the internal programs and data.

These directories may be relocated from the values set when GNUbatch is built by assignment to environment variables. These environment variable assignments may (and probably should) be placed in the master configuration file, `/usr/local/etc/gnubatch.conf`, to ensure consistency. The directories and corresponding environment variables are as follows:

Directory	Environment Variable	Function
`libexec/gnubatch`	`SPROGDIR`	Internal programs
`share/gnubatch`	`SDATADIR`	Data and control files
`share/gnubatch/help`	`SPHELPDIR`	Help files for programs
`var/gnubatch`	`SPOOLDIR`	Spool directory for pending jobs

Take care not to assign values to these environment variables arbitrarily; very strange things will happen if one part of **GNUbatch** is using one set of directories and some other part is using another!

3.1.2 Internal Programs

These include the scheduler, network connection daemon, xbnetserv and the utilities used by them. They are held in the internal programs directory. With certain exceptions it is not intended that users should ever invoke these programs.

The file structure of the internal programs is flat within their directory.

3.1.3 Batch directory files

The following GNUbatch internal files are held in the spool directory, which by default is `/usr/local/var/gnubatch`:

File	**Purpose**
`btufile1`	User permissions
`gbch-charges1`	User charges (deprecated)
`cifile`	Specification of command interpreters
`holfile`	Days set to be holidays
`btsched_jfile`	Saved record of jobs
`btsched_vfile`	Saved record of variables
`btsched_reps`	Report file holding any messages output by btsched
`pwdump1`	Optional saved password map file (see under gbch-uchange)
`SPnnnnnnnn`	Queued jobs
`SOnnnnnnnn`	Standard output of pending jobs
`ERnnnnnnnn`	Standard error of pending jobs
`NTnnnnnnnn`	Local copy of remote job

The above files are owned by `batch`. Unused copies of the last four kinds of files may safely be deleted. The *nnnnnnnn* component of the file name is derived from the batch job number.

3.1.4 Help and Message files

GNUbatch reads all of its messages from a series of text files (Apart from the "help I cannot find the message file" messages). The user may adjust these to tailor the command interface, help and error messages to be suitable for the particular installation. These are *system-wide* message files. It is also possible to set up customised versions for individual users or applications.

The following files are, by default, owned by `batch` and held in the directory, by default `/usr/local/share/gnubatch/help`.

File	**Purpose**
`btq.help`	Screen layout, messages and key assignments for gbch-q
`btuser.help`	Screen layout, messages and key assignments for gbch-user
`btrest.help`	Messages and arguments for other user programs
`btint-config`	Message file for btsched, btwrite and xbnetserv
`filemon.help`	Message file for file monitor program, gbch-filemon.
`xmbtq.help`	Message file for gbch-xq and gbch-xmq
`xmbtr.help`	Message file for gbch-xr and gbch-xmr
`xmbtuser.help`	Message file for gbch-xuser and gbch-xmuser

Please refer to the chapters on Configurability and Extensibility for details of how to modify these files.

3.1.5 Configuration files held in /usr/local/etc

GNUbatch uses up to three files held in the system directory `/usr/local/etc`.

3.1.5.1 GNUbatch Hosts File

The file `/usr/local/etc/gnubatch.hosts` is used on networked installations of GNUbatch to denote details of the remote hosts and clients to which connection is to be made.

Each line in the file other than blank lines or comment lines (introduced with a # sign) consists of up to 4 fields. These are as follows:

1. The *hostname* to attach to or an internet address such as 197.3.9.1. For DHCP clients, this gives the Windows user name to be recognised (case insensitive).
2. An *alias name* by which the remote host is to be referred to within GNUbatch. The user can give either the host name or the alias name in commands such as gbch-conn but displays (as in gbch-q or gbch-vlist) will always use the alias. For DHCP clients, this gives the Unix user name (if different) corresponding to the given Windows user name.

   An alias or Unix user name can be omitted by just putting a single "`-`" sign.

   An alias *must* be supplied if the host name is given as an internet address.
3. *Flags*, which are further described below.
4. A numeric time-out value in seconds. The default if this is omitted is 1000. This is most important for Windows clients, as it also denotes a time after which the connection becomes "stale" and must be refreshed, possibly by re-entering the password.

The flags field is one or more of the following separated by commas.

`probe`	Denotes that the scheduler should check that the specified host is active before attempting a connection.
`manual`	Denotes that no connection is attempted until the operator invokes one with gbch-conn.
`external`	Denotes that the named host is some external system. Currently this has no meaning in GNUbatch.
`dos(user)`	For a Microsoft Windows client PC. Requests are allocated by default to the username given.
`client(user)`	Is a synonym for `dos(user)`.
`clientuser`	Denotes that the first and second fields are *`user`*, not machine names, for DHCP clients. As a special case, if the first field is default and the second field is a user name on the Unix host, then a default user name is thereby supplied for all unknown Windows users.
`clientuser(machine)`	As `clientuser`, but denotes that the default client machine is given, otherwise a password is required.

`trusted`	Exchange information with this Unix host about Windows client users.
`pwchk`	Demand Unix password from Windows clients in all cases.

For example:

```
mach19          red    probe
mach20          green  probe,manual
192.112.238.7   yellow probe
WS21            blue   dos(jmc)         30
john            jmc    clientuser,pwchk
default         guest  clientuser
```

This provides for 4 machines, where host names are `mach19`, `mach20`, `WS21` an IP address and also a user name for DHCP clients. These are given aliases of `red`, `green`, `yellow` and `blue`.

In the first and third case any connection will be tested first before continuing.

In the `green` case no connection is attempted until the user types,

```
gbch-conn green
```

or

```
gbch-conn mach20
```

The `blue` machine is a Microsoft Windows workstation. Requests will be assumed to come from user `jmc`. Time-outs of 30 seconds apply to requests.

Next the Windows user name of `john` on any Windows PC is translated to a Unix user name of `jmc`, after checking the password.

Finally, any unrecognised Windows user name is treated as the Unix user name of guest.

The utility program gbch-hostedit (or the GUI version gbch-xhostedit) may be used to create or edit this file with appropriate checks.

Note that the mapping of UNIX names to Windows names is deprecated – this is now done in the user mapping file.

3.1.5.1.1 Multiple IP addresses

It is sometimes unclear what the local address is, i.e. the IP address corresponding to the host on which it is running. It is important for the software to know this, as other hosts will use this to identify jobs and variables belonging to the host. It is possible to specify this in the hosts file thus:

```
localaddress 193.112.238.250
```

The `localaddress` statement must be the first item (other than comments or blank lines) in the host file. The address given can be either a host name or an IP address.

The address can also be obtained each time it is started by connecting to another host and running `getsockname()` on the result. To signify this, the following format is used.

```
Localaddress GSN(www.gnu.org,80)
```

The integer gives a port number to use.

The host name can be given as above, or an IP address can be used.

3.1.5.2 GNUbatch Master Configuration File

In order to work properly, the scheduler process and all the other programs must be started with the same environment variables. For convenience, the environment may be initialised for each program by creating a master configuration file `/usr/local/etc/gnubatch.conf`.

This file contains a list of environment variable assignments. Any environment variables not defined on entry to any of the programs are initialised from this file. Any environment variables used by GNUbatch may be included in this file, not just those shown in the example.

For example:

```
SPOOLDIR:/usr1/spool/batch
SPROGDIR:/usr1/spool/bin
MAILER=/usr/lib/sendmail
```

An environment variable declared using the equality sign `=` will be included in the environment of all batch jobs that are submitted. This may not be wanted for all variables, in particular the scheduler directories pointed to by `SPOOLDIR`, `SPROGDIR` and `SPHELPDIR`. To avoid jobs inheriting environment variables from the configuration file declare them using the colon, `:` , instead of the equals sign, `=`.

Please note that the text to the right of the colon or `=` sign is taken literally; there is no recursive expansion of `$name` constructs except for the message file names `BTQCONF`, `BTUSERCONF` and `BTRESTCONF` (where it is limited to 10 recursive expansions).

3.1.5.3 User Mapping file

The user map file provides a mapping between external names, usually Windows user names, and UNIX names.

The file is in `/usr/local/etc/gbuser.map`, and consists (apart from comments introduced by the # character) of lines of the format

```
unix-user:windows-user
```

For example:

```
# User mapping file
jmc:john collins
sec:sue collins
guest:default
```

The final entry gives a default user if a named user is not found in the file.

UNIX users not found on the host are silently ignored.

3.1.5.4 GNUbatch Static Environment File

To avoid every job having to have all the environment variables in it, thus saving space, the static environment file, `/usr/local/etc/gnubatch-env`, is provided. The commands that submit jobs will only store "differences" from this file in each job. This is provided to avoid saving large amounts of environment information with each job.

Alternative files to `/usr/local/etc/gnubatch-env` can be specified by including the following line in `/usr/local/etc/gnubatch.conf`:

```
BATCHENV:file1,file2....
```

These files are read in sequence and constitute the new environment.

3.2 Job and Variable Modes

Each job and variable is given a *protection mode*. This consists of a set of permissions dictating how various users may, or may not, access the job or variable. The modes are like those on Unix files, providing *user*, *group* and *other* access. An expanded set of permissions has been devised to enable the permissions to control separate operations.

The permissions are as follows, one set for each of *user*, *group* and *others*:

Permission	Function
Read	Job or variable may be read
Write	Job or variable may be written
Reveal	Job or variable is 'visible' to user
Read modes	Modes may be displayed
Set modes	Modes may be set
Give away Owner	Ownership may be given away
Give away Group	Group may be given away
Assume Ownership	Ownership may be assumed
Assume Group	Group may be assumed
Delete	Job or variable may be deleted
Kill (jobs only)	Job may be killed

Only the primary group of a user is considered when evaluating group access permissions.

The visibility of variables and jobs can be set to the local machine only or all networked GNUbatch machines.

3.2.1 Change of owner and group

Changes of owner and group take place in 2 stages for security.

1. The existing owner, or someone with *give away* permission gives the job or variable away to a designated owner or group. This designated owner or group is noted, but the change has no effect at this stage.
2. The designated owner or a user with that group will have to explicitly *assume ownership* of the job or variable. This owner or group must also have the appropriate permission.

This 2-stage process is to prevent the security violations of unauthorised assumption of ownership, and also to prevent jobs from being run masquerading as unauthorised users.

A user with *write administration file* privilege does not have to go through this procedure. Changes to owner or group of a job or variable by such users are immediate and complete.

3.2.2 Initialisation of modes

The modes of jobs and variables are set when they are created, however users authorised by the mode may reset them subsequently.

In the case of *jobs*, the modes set by the option `-M` to gbch-r are used, in default of which a set of default modes for the given user are set.

In the case of *variables*, the mode is set from the default modes for the given user.

A user may be permitted to reset his own default modes with the *change default modes* privilege as described in a later chapter, using gbch-user. A system-wide 'default default mode' is given to each new user, along with a default set of privileges.

As distributed, GNUbatch will assign the following default modes to jobs and variables:

Permission	**Jobs**			**Variables**		
	User	**Group**	**Other**	**User**	**Group**	**Other**
Read	Yes	Yes	No	Yes	Yes	No
Write	Yes	No	No	Yes	No	No
Reveal	Yes	Yes	Yes	Yes	Yes	Yes
Read Mode	Yes	Yes	Yes	Yes	Yes	Yes
Set Mode	Yes	No	No	Yes	No	No
Give away owner	Yes	No	No	Yes	No	No
Give away group	Yes	Yes	No	Yes	Yes	No
Assume owner	No	No	No	No	No	No
Assume group	No	No	No	No	No	No
Delete	Yes	No	No	Yes	No	No
Kill	Yes	No	No	N/A	N/A	N/A

3.3 Standard Exit Codes

The command line programs are often run from within other programs or shell scripts. To allow convenient error diagnosis, there is a set of standard exit codes which are used by the GNUbatch programs. These are:

Exit	**Description of Probable Cause**
0	Return true, i.e. program ran correctly
1	Return false, returned only by gbch-var and gbch-jstat for test operations which fail.
2	Bad arguments to program
3	Invalid permissions on job or variable for operation

Exit	Description of Probable Cause
4	gbch-var only - lost race competing with someone else
5	Could not cd to spool directory (probable set-up error)
6	Scheduler, i.e. btsched, not running
7	Unknown host: gbch-conn, gbch-rr, etc
8	TCP error
11	btsched shutting down
13	Unknown job: gbch-jdel or gbch-jchange
14	gbch-cichange name clashes with an existing command interpreter
16	No privilege for requested operation
19	File not found (actually not used anywhere)
20	Variable not found
30	User not set up, run gbch-uchange -R
31	Unknown user: gbch-charge, gbch-uchange, etc
32	Cannot perform operation because the job is running
50	Cannot create file in spool directory, disc probably full
100	Corrupted help or configuration file
101	Terminal input error (in curses library)
150	Internal error for jobdump, file not found
151	Internal error for jobdump, directory not found
152	Internal error for jobdump, cannot create file
153	Internal error for jobdump, job not found
154	Internal error for jobdump, cannot delete job
155	Internal error for jobdump, cannot save options
200	User program received unexpected signal
201	Cannot create pipe, check for disc full
202	Cannot fork, process table probably full
203	Cannot access shared memory for jobs, set-up probably scrambled
204	Cannot access shared memory for variables, set-up probably scrambled
240	scheduling process btsched has failed
246	Cannot access working directory for job (in last stages of starting job)
247	Job not found (in last stages of starting job)
248	Unknown command interpreter (in last stages of starting job)
249	Could not create/open file in redirections
250	Something strange, probable set up error
251	Cannot create pipe to execute job, disc probably full
252	Cannot fork to execute job, process table full
253	Ran out of string space in job for environment variables etc.
254	Process ran out of memory
255	Cannot find help message file.

4 User Administration

GNUbatch maintains a list of users which is generated from the password system (whether using the `/etc/passwd` file or NIS). Hence, each user must first have a Unix account in order to have a GNUbatch account.

User permissions are now held as a default set together with differences for specific users, so no special action need be taken when users are added or deleted, unless they are distinguished in some way.

There are 4 aspects to the GNUbatch user account:

Privileges
Control access to usage and administration functions of the system. For example, the privilege to submit jobs to the queue.

Charges
Provide an accounting mechanism for each users batch work.

Priorities
When there are more jobs ready to run than are allowed these position the "ready" jobs with respect to each other. Facilities exist to specify what priorities each GNUbatch user may specify for their batch jobs.

Modes
Specify the default modes that are placed on jobs and variables for the user who creates them. Modes are described in detail in the Variables chapter and the Jobs chapter.

4.1 Privileges

In addition to the ability to access jobs and variables in the manners described by the *modes*, each user has a number of *privileges*, as follows. The privileges may be individually set for each user using gbch-user, and a default established for new users. The privileges are:

Privilege	**Description**
Read admin file	User may display contents of administration file showing users, charges and privileges.
Write admin. file	User has full write access to administration file.
Create entry	User may create jobs and variables. This permission is granted by default.
Special Create	User may update command interpreter file or adjust load levels.
Stop scheduler	User may stop GNUbatch (i.e. run gbch-quit)
Change default modes	User may change his/her default modes. This permission is granted by default.
Combine user and group permissions	If the user has this privilege, then any job or variable in the user's primary group will have the permissions of "owner" and "group" combined.
Combine user and other permissions	If the user has this privilege, then any job or variable in not in the user's primary group will have the permissions of "owner" and "others" combined.

Privilege	Description
Combine group and other permissions	If the user has this privilege, then any job or variable will have the permissions of "group" and "others" combined, effectively "turning off" any difference between "group" and "other" permissions.

Unless otherwise stated in the above table the privileges are turned off by default. The default privileges are those which by default are applied to new users. They may be changed by a system administrator using gbch-user.

A *system administrator* is any user with all privileges enabled, especially the *write administration file* privilege. Initially the super-user, `root`, and `batch` are designated as system administrators. A particular feature of this privilege is that changes to owner or group of jobs or variables are immediate and complete.

To save screen space abbreviations are often used in GNUbatch to represent these permissions. An example of these may be seen on the main screen of the user administration tools, gbch-user, gbch-xuser and gbch-xmuser.

The set of abbreviations used is:

Mnemonic	Privilege
`RA`	Read admin File
`WA`	Write admin File
`CR`	Create entry
`SPC`	Special Create
`ST`	Stop scheduler
`Cdft`	Change default modes
`UG`	Combine user and group permissions
`UO`	Combine user and other permissions
`GO`	Combine group and other permissions

4.2 Priorities

A batch job may have a priority in the range of 1 to 255. Users will usually be restricted to a smaller range between their individual minimum and maximum priorities, but which are normally the system defaults, initially 100 to 200. A default priority for each user may be set; again there is a system default, initially 150.

When a job is queued using gbch-r, it is given the user's default priority unless overridden with the `-p` option. It is possible to set a users minimum, maximum and default priorities to apparently useless values, but, in fact, these combinations provide possibly useful restrictions:

Combination	Meaning
min <= default <= max	Normal usage; user may set priority anywhere between *min* and *max*.
min = default = max	Special case; user can only use *default* priority.
min > max	"Impossible"; user may not use gbch-r will complain.

default > *max* or *default* < *min*	Default priority is effectively undefined. User **must** explicitly give a priority to gbch-r, or the job will be rejected with an error message.

Jobs belonging to remote machines may appear in different places on the queue than on their machines when they initially come on line, but this situation, which is harmless, should in any case rapidly adjust itself.

4.3 Charging

Charging is deprecated and has been removed from GNUbatch.

4.4 Modes

The modes of jobs and variables are set when they are created. Unless otherwise specified, they are set to the default modes in force for the user who created the job or variable.

See Job and Variable Modes for an introduction to Modes. Additional information follows about jobs and variables in the relevant sections.

A user may be permitted to reset his/her own default modes with the *change default modes* privilege, using gbch-user. A system-wide 'default default mode' is given to each new user, along with a default set of privileges.

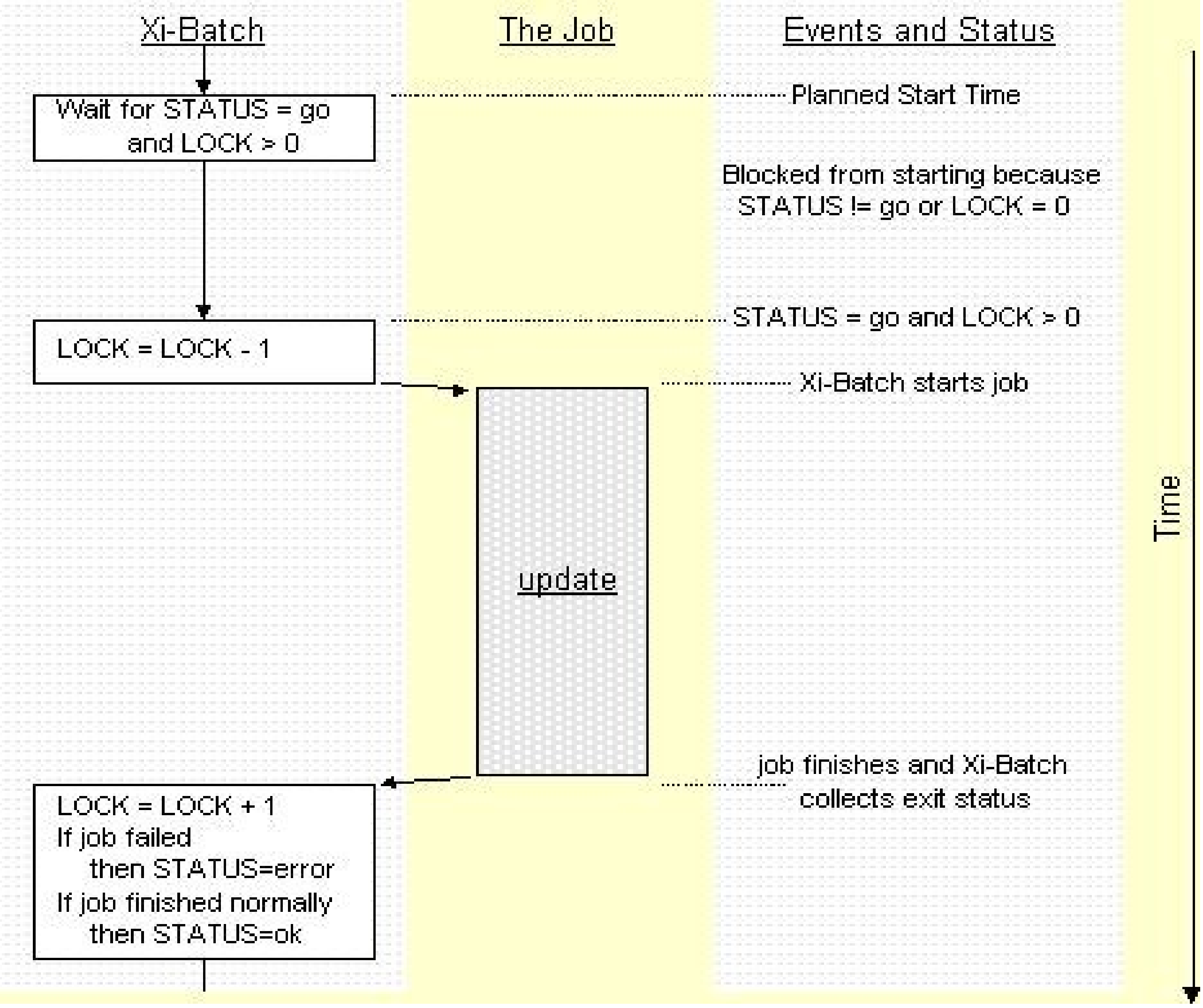
As distributed, GNUbatch will assign default modes to users for the jobs and variables they create as described previously.

5 Job control variables

GNUbatch provides Job control variables for handling all types of dependencies. In this manual, when it is not likely to be ambiguous, Job control variables are usually just referred to as variables.

There are some variables defined as standard, which are used for controlling the overall operation of GNUbatch. Other variables may be created, modified, queried and deleted by any suitably authorised users, via interactive and command line tools.

The specifications for batch jobs include options for interacting with variables before jobs may start, when jobs are starting and when they finish. For example a job called update may use two variables, called `STATUS` and `COUNT`, for dependencies with other jobs and the outside world.


Xi-Batch
The Job
Events and Status
Wait for STATUS = go and LOCK > 0
Planned Start Time
Blocked from starting because STATUS != go or LOCK = 0
LOCK = LOCK - 1
STATUS = go and LOCK > 0
Xi-Batch starts job
update
Time
job finishes and Xi-Batch collects exit status
LOCK = LOCK + 1
If job failed
then STATUS=error
If job finished normally
then STATUS=ok

In this example the job `update` may not start until the variable `STATUS` contains the string "`go`" and the variable `COUNT` contains an integer value less than 10. In GNUbatch terminology these tests are called *conditions*. The variables can have their values set by any combination of the following:

- GNUbatch running jobs which specify variable operations for starting.

- GNUbatch collecting the exit status of jobs which have finished and specifying variable operations for normal exit, error and/or abort.
- Users changing the value of variables manually.
- Batch jobs changing the value of variables themselves.
- The file monitor program gbch-filemon setting a variable or variables to indicate the arrival or change of a file.
- Other processes unconnected with GNUbatch changing the values of variables via the API or using the command-line tools.

Returning to our example, either one or both of the variables do not have the required values, so GNUbatch waits until both variables meet the conditions on job update. Once the conditions are met GNUbatch starts job update running.

When the job finishes, GNUbatch gets the exit status and performs any specified operations. In this case it will increment `COUNT` by one (however the job finished) and set `STATUS` to "`ok`" if the job worked or "error" if it did not. In GNUbatch all operations on variables that are specified in the options for a batch job are called *assignments*.

Operations on variables are "atomic". No other jobs, processes or users can access a variable or variables whilst they are being tested or changed. This is best seen looking at the assignment of variables `STATUS` and `COUNT` when job update finished. Both variables were protected from the time GNUbatch started incrementing `COUNT` until it finished assigning the appropriate string to `STATUS`.

5.1 Dependency on files

Implementation of dependencies based on files is implemented via the *File Monitoring Option*, gbch-filemon, described in Chapter 8.7.

It is not integrated with the main part of GNUbatch as it is necessary to "poll" (repeatedly interrogate at fixed intervals) the relevant files and directories to monitor for changes.

Typically, gbch-filemon will be set up to modify a variable when the requested file event occurs (although it could do something completely different).

5.2 What's in a Variable?

Apart from having a Name and some Contents, variables also have other pieces of information associated with them. The set of fields which make up a variable are:

Name

The unique identifier by which the variable is referred to. It is an alphanumeric string which must start with an alphabetical character. GNUbatch will make sure that there are no duplicate or invalid names on any machine.

Value

The information contained by the variable, which can be either an integer value or a text string.

Comment

A free text string which should be used to hold a brief description of the variable.

User

The name of the user who owns the variable. This is set to the person who created the variable unless it has been transferred to another user.

Group

Like the user field this shows which group the variable belongs to. It is set to the primary group of the user who created the variable, unless it has subsequently been transferred to another.

Mode

Like the modes on a Unix file, these specify who may see and modify the variable. There are, however, far more modes than those associated with files. Refer to the next section for "More about Modes".

Export flag

Variables may be declared as purely local to the machine or accessible by any co-operating GNUbatch host. This is a binary flag having the value Local or Exported. Note that two or more hosts may have variables of the same name, the name is distinguished by the host name, thus host:name.

Cluster flag

If a variable is marked "clustered" in addition to exported, then a job running on the given host will use that local version of the variable of that name for conditions and assignments applied to the job.[1]

5.3 More about Modes

Access to variables is controlled by the Modes which are similar to Unix file permissions, but with greater functionality. Permission to each access mode is granted to the owner of the variable (User), users in the same primary group (Group) or everyone (Others).

Here is a screen, from the interactive batch queue management tool gbch-q, showing the modes for a variable called A_STATUS.

```
Modes for Variable `A_STATUS'
Variable owner wally group staff
                       User   Group Others
Read                   Yes    Yes   No
Write                  Yes    No    No
Reveal                 Yes    Yes   Yes
Display mode           Yes    Yes   Yes
Set mode               Yes    No    No
Assume ownership       No     No    No
Assume group ownership No     No    No
Give away owner        Yes    No    No
Give away group        Yes    Yes   No
Delete                 Yes    No    No
```

The various modes give the following type of access when permission is granted:

`Read`	The variable and its contents can be read.
`Write`	The data, name and comment of a variable may be modified.

---

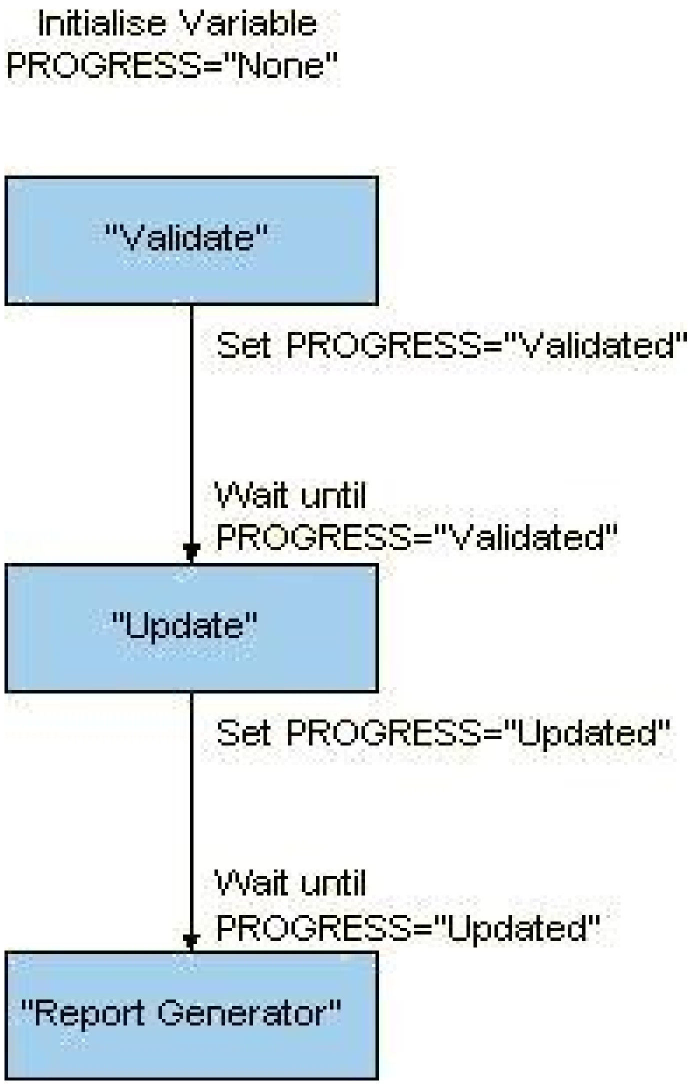
[1] GNUbatch2 will make that a feature of a condition rather than a variable.

`Reveal`	Variables will be completely invisible to users without reveal permission. If reveal permission is granted but not read permission then only the name, owner and group of a variable may be seen. The contents and comment will be hidden.
`Display mode`	Allows these modes to be viewed.
`Set mode`	Allows these modes to be changed.
`Assume ownership`	Dictates to whom ownership of a variable may be transferred relative to the current owner. Hence giving Assume ownership permission to just the owning User has no value, since the owner can only give the variable to themselves.
`Assume group ownership`	Dictates to which primary group a variable may be transferred relative to the current group. Only granting this privilege to Other is meaningful. User and Group are included to be orthogonal.
`Give away owner`	Grants permission to transfer the ownership of a variable to another user. The permission may be given to the owner, members of the same primary group or anyone.
`Give away group`	Grants permission to transfer the variable to another group. The permission may be given to the owner, members of the same primary group or anyone.
`Delete`	Permission to delete variable from GNUbatch

5.4 Examples of Dependencies Handled by Variables

Any job can have several conditions and make several assignments, hence the possibilities for handling dependencies are infinite. This section contains some examples to help understand the importance of variables.

5.4.1 Running Jobs in a Simple Chain


Initialise Variable
PROGRESS="None"
"Validate"
Set PROGRESS="Validated"
Wait until
PROGRESS="Validated"
"Update"
Set PROGRESS="Updated"
Wait until
PROGRESS="Updated"
"Report Generator"

The simplest example of job dependencies is a single threaded chain of jobs, which is controlled by one variable. Each job in the chain is required to start as soon as possible after the previous job has finished.

To prevent subsequent jobs firing off prematurely the variable, `PROGRESS`, must be initialised to some safe value before the first job starts. Words like "`None`", "`Ready`", "`Standby`" or the empty string, "", are good options.

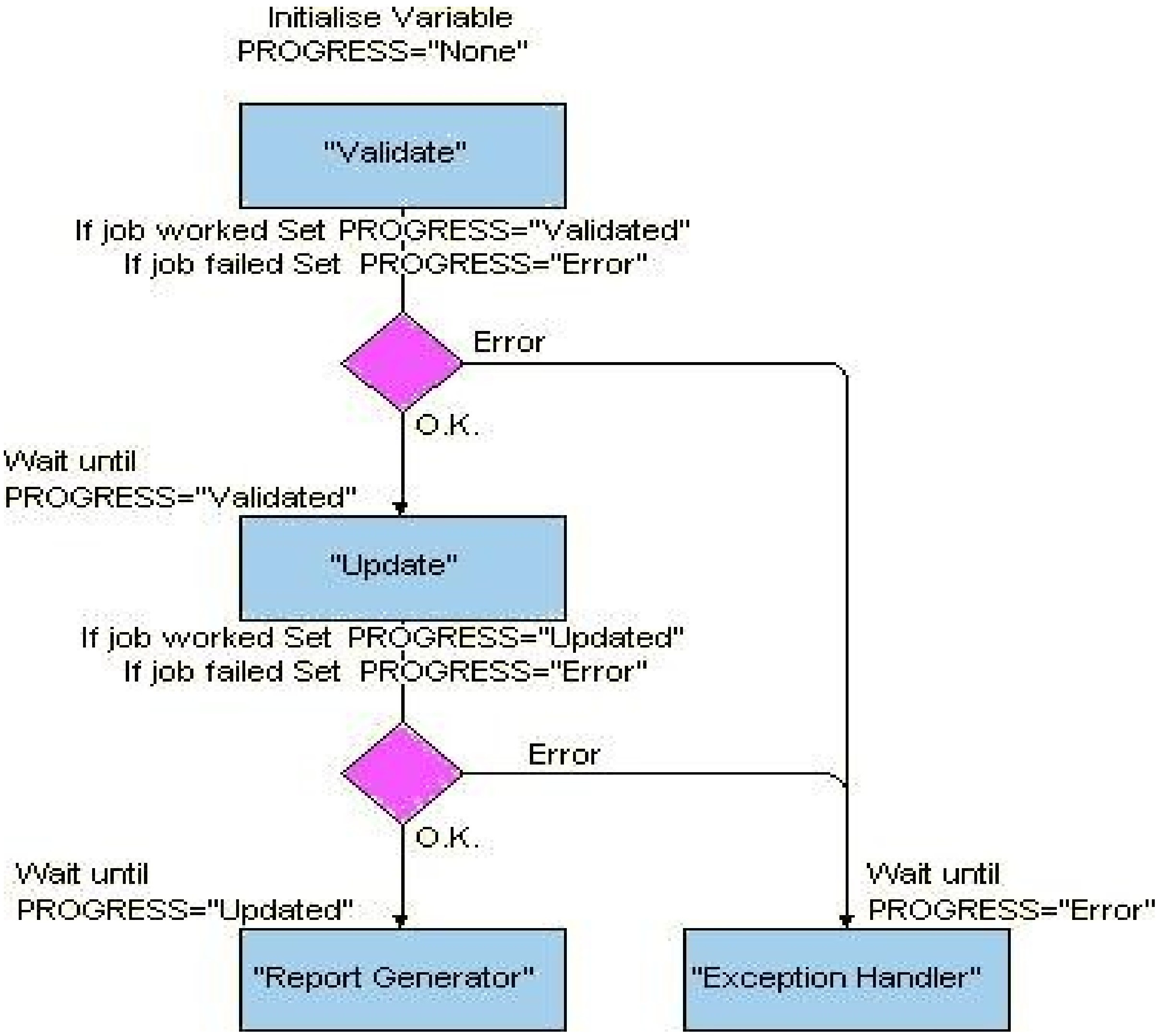
The most obvious values to use for `PROGRESS` are perhaps numeric, since each job could simply increment the value by one.

Using string values has the following advantages:

- By choosing meaningful names the progress through the chain of jobs can be seen by looking at the value in `PROGRESS`.
- Only two jobs need their options changing if new jobs are added or existing jobs are removed from the chain. One job will need a condition changing and the other an assignment.

5.4.2 Running jobs in a chain with exception handling

This is the same simple chain as used in the previous example but with alternative execution paths to run an exception handler if either of the first two jobs fail. In this case there is one exception handler, but it would be as easy to have a different handler for the Validate and Update jobs. The value of `PROGRESS` is set to a different string depending on the success or failure of the job.


Initialise Variable
PROGRESS="None"
"Validate"
If job worked Set PROGRESS="Validated"
If job failed Set PROGRESS="Error"
Error
O.K.
Wait until
PROGRESS="Validated"
"Update"
If job worked Set PROGRESS="Updated"
If job failed Set PROGRESS="Error"
Error
O.K.
Wait until
PROGRESS="Updated"
Wait until
PROGRESS="Error"
"Report Generator"
"Exception Handler"

This is a very simple scenario. The exception handler(s) could be programmed to perform recovery actions and resume processing of the chain at the appropriate job.

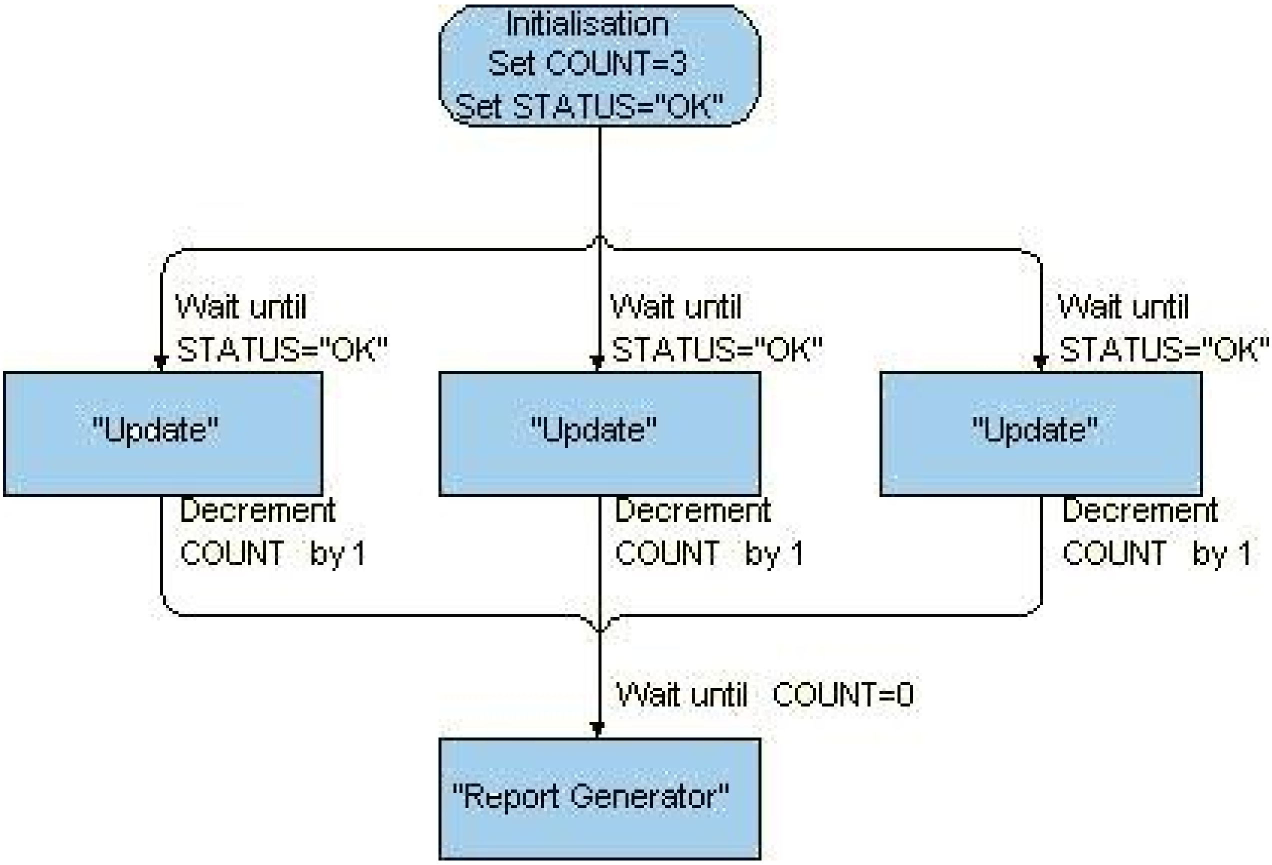
Two variables could be used in this example if it is desirable to keep the job exit status separate from the progress through the chain. `PROGRESS` could always be set to indicate the last job that finished. Another variable, for example `STATUS`, could be initialised to "`OK`" and set to "`Error`" by any job that failed.

5.4.3 Running Jobs in Parallel

A sequence of jobs can contain some that will be running in parallel, to be followed by one or more subsequent jobs. Only one variable is needed to indicate when all of the parallel jobs have completed.

The variable should be initialised to the number of jobs that will be running concurrently. Each job then decrements this variable by one on completion. When all of the jobs have finished the value of the variable will be 0.

For example if three jobs are due to run in parallel, using the variable `COUNT` then initialise `COUNT` to the value of 3.


Initialisation
Set COUNT=3
Set STATUS="OK"
Wait until STATUS="OK"
Wait until STATUS="OK"
Wait until STATUS="OK"
"Update"
"Update"
"Update"
Decrement COUNT by 1
Decrement COUNT by 1
Decrement COUNT by 1
Wait until COUNT=0
"Report Generator"

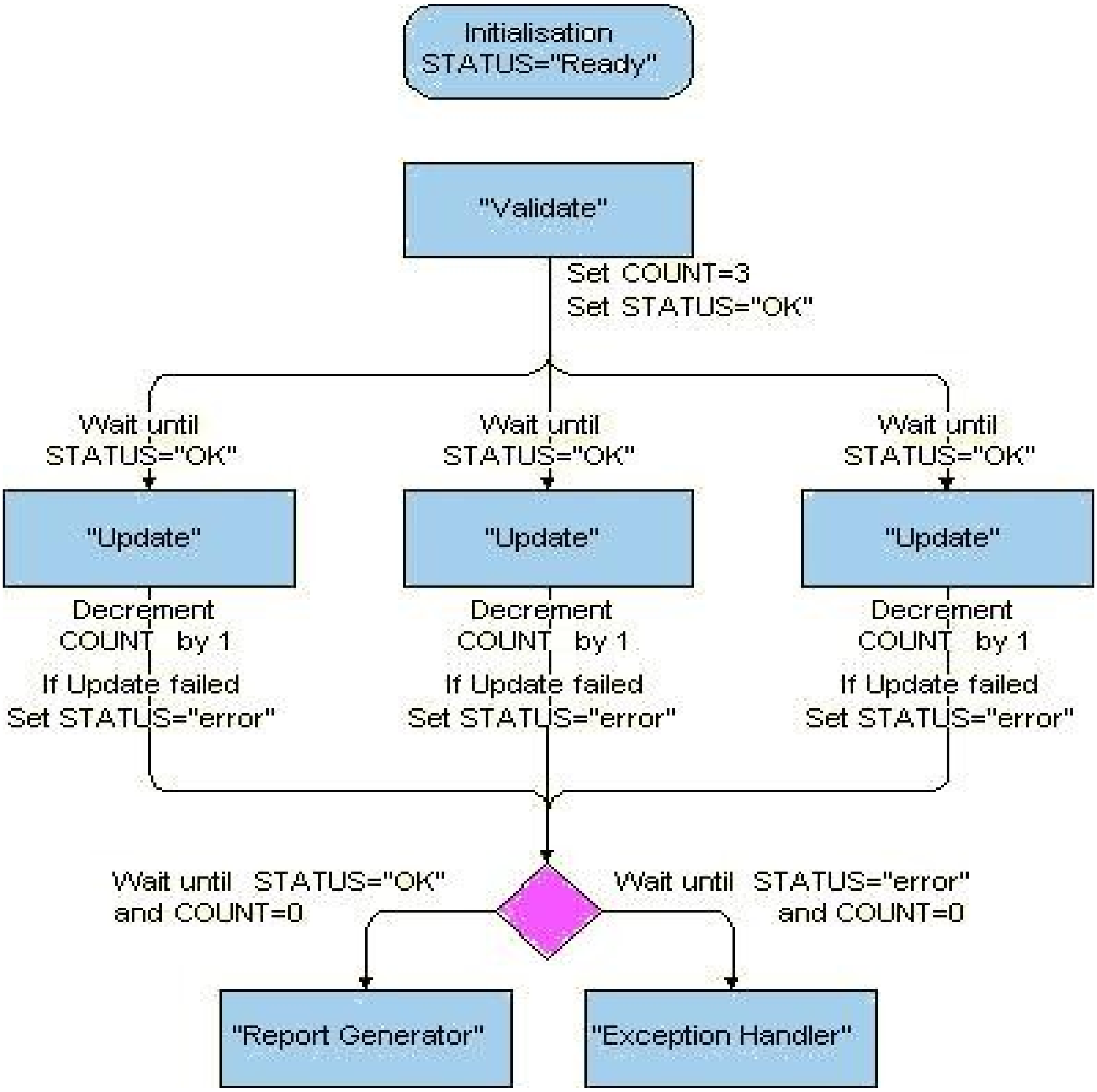
The diagram shows the three jobs waiting for a variable called `STATUS` before starting. This is easier to follow than using `COUNT` to control the job start as well.

The value of `COUNT` should not be used to indicate the success or failure of the jobs. If this is required then an additional variable will be needed, as shown in the next section.

5.4.4 The Parallel Example with an Exception Handler

Providing exception handling in a group of parallel jobs will require two variables. One variable will act as a counter to show when all jobs are completed, as in the previous section. Another variable will be required to show if an error occurred.

The error status variable must be initialised to a value indicating that no errors occurred. Any job which fails should assign an error status to the variable. All jobs which finish normally must leave the variable alone. For example here is the previous example expanded to use the variable `STATUS` to control whether a Report Generator or an Exception Handler job will run after the three concurrent Update jobs.


Initialisation
STATUS="Ready"
"Validate"
Set COUNT=3
Set STATUS="OK"
Wait until STATUS="OK"
Wait until STATUS="OK"
Wait until STATUS="OK"
"Update"
"Update"
"Update"
Decrement COUNT by 1
If Update failed Set STATUS="error"
Decrement COUNT by 1
If Update failed Set STATUS="error"
Decrement COUNT by 1
If Update failed Set STATUS="error"
Wait until STATUS="OK" and COUNT=0
Wait until STATUS="error" and COUNT=0
"Report Generator"
"Exception Handler"

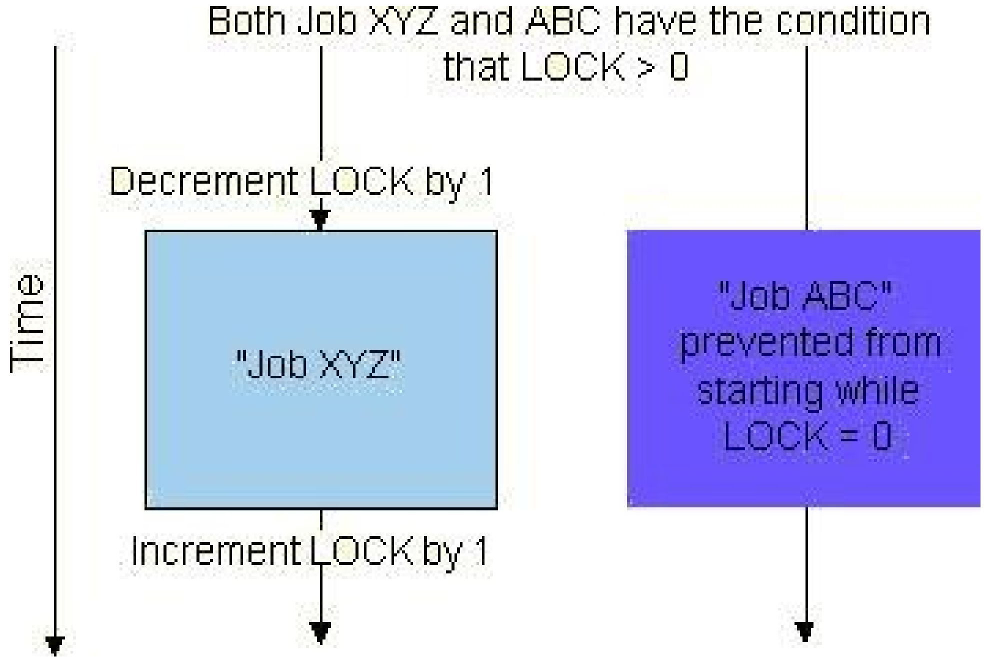
5.4.5 Mutual Exclusion & Semaphores

It is quite common to find two or more jobs that must never be allowed to run at the same time. This could be because they need the same piece of hardware, like a particular tape drive. Alternatively they may write to a file or update a database which could be corrupted by being opened by more than one job.

To enforce the required Mutual Exclusion amongst such a group of jobs, a variable can be used as a semaphore. All jobs in this group are given a condition that they may only start when the variable has a certain value. They also have an assignment that sets the variable to some other value on starting and returns it to the initial value on completion.

For example:

Two jobs XYZ and ABC can be controlled by one locking variable, called LOCK. Both jobs have the condition that LOCK must be greater than 0. If LOCK is initialised to 1 then either job may start when its scheduled start time arrives. The jobs decrement LOCK on starting and increment it on finishing, hence LOCK can only have the values 0 and 1.


Both Job XYZ and ABC have the condition that LOCK > 0
Time
Decrement LOCK by 1
"Job XYZ"
"Job ABC" prevented from starting while LOCK = 0
Increment LOCK by 1

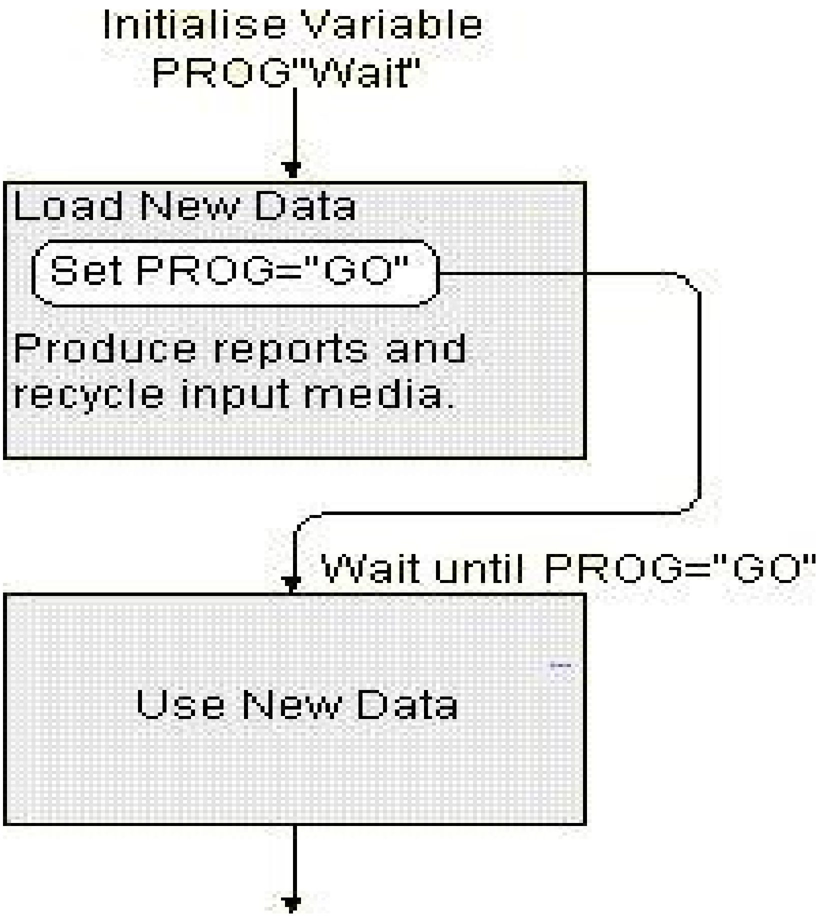
By using the decrement and increment operators in the example we support the general case for limiting the number of concurrently running jobs to a specific value. In this case 1 enforces mutual exclusion.

If up to a given number of jobs may run at the same time then `LOCK` should be initialised to that number. So for a maximum of seven jobs initialise `LOCK = 7`.

5.4.6 Passing Data between Jobs

The values of variables can be queried and/or assigned from within a job using the appropriate command line programs (or functions of the API). This provides enormous flexibility to do the same things between jobs as ordinary variables may do inside them.

For example:


Initialise Variable
PROG"Wait"
Load New Data
Set PROG="GO"
Produce reports and recycle input media.
Wait until PROG="GO"
Use New Data

A simple use for setting a Job Control Variable inside a job would be to indicate that the next one in a chain may start before it has finished.

Other operations that can be performed include:

- Exchanging small amounts of data without using an intermediate file (or possibly the name of a file containing large amounts of data).
- Providing mutual exclusion between jobs during critical processing rather than for the whole execution of a job.
- Interrogating variables to see how other jobs have run and are running.
- Initialising variables to the required values for a particular schedule of jobs.
- Modifying execution of other jobs in a particular schedule.

5.5 System variables and logging

There are seven pre-defined "System" variables known to GNUbatch. They are initially set to be owned by batch with the default modes which may be reset if desired. These variables may not be deleted or set to an invalid value (e.g. string for numeric variable etc.). They may be included in job conditions or assignments provided that these do not attempt to perform an invalid operation on them.

The variables are:

`CLOAD`	GNUbatch updates `CLOAD` in real time to show the total load level of all currently-running jobs. This is a read-only variable.
`LOADLEVEL`	Controls the maximum load of batch jobs that may be running on the system. Jobs can only be started when `CLOAD` is less than `LOADLEVEL` and it will not put `CLOAD` over the limit set by `LOADLEVEL`. `LOADLEVEL` may only be set to a numeric value. It may be specified when GNUbatch is started using the `-l` option to gbch-start, usually to zero, to give maximum control. If the value is increased, then new jobs may start immediately. If the value is reduced, then it is possible that the total load level of running jobs may temporarily exceed it until some of them terminate, however no new jobs will start until the level is no longer exceeded.
`LOGJOBS`	Specifies where to send output from the job audit trail logging. If the variable holds null (an empty data field) then logging is turned off.
`LOGVARS`	Specifies where to send log output for the variable audit trail. If the variable holds null (an empty data field) then logging is turned off.
`MACHINE`	This is a read-only variable containing the current machine name, that can only be referenced as a local variable.
`STARTLIM`	This variable contains the maximum number of jobs that GNUbatch can start at once. The initial value, upon installation of GNUbatch, is 5.
`STARTWAIT`	This variable contains the waiting time in seconds for the next available job to start if the previous batch set by `STARTLIM` has not been started for some reason. The initial value upon installation of GNUbatch is set to be 30 seconds.

5.5.1 Using CLOAD & LOADLEVEL

`LOADLEVEL` and `CLOAD` may be used to control the batch workload and avoid conflicts with other activities in a variety of ways.

5.5.1.1 Running fewer batch jobs in office hours.

A batch job can be set up to reduce the value of `LOADLEVEL` at the start of office hours, to prevent batch jobs slowing down interactive users. Another job can run at the close of office hours to put `LOADLEVEL` back up to the overnight level.

5.5.1.2 Stopping GNUbatch gracefully

To stop GNUbatch, yet allow jobs to complete, perform the following steps:

1. Set `LOADLEVEL=0`
2. Wait until `CLOAD=0`
3. Stop GNUbatch

This can be done manually or incorporated in a shell script.

It may even be set up as a batch job. However we would recommend that such a batch job has a very low load level, such as 10, much lower than any other job, and is conditional on `LOADLEVEL` being equal to 10 and `CLOAD` being less than or equal to 10, so that to start it, all that is required is to set `LOADLEVEL` to 10, and it will automatically wait until other jobs have finished. At all other times, `LOADLEVEL` would never be set to exactly 10, GNUbatch always being restarted with the `-l` option to gbch-start.

Batch jobs which stop the scheduler must launch their script asynchronously to avoid killing themselves with the gbch-quit command. So the actual "script" of the job would be:

```
/usr/sbin/batchshutdown &
```

and `/usr/sbin/batchshutdown` would contain

```
#! /bin/sh
sleep 10
gbch-quit -y
```

This would give the shutdown job time to complete, before invoking the gbch-quit command.

5.5.1.3 Starting Administration activities, when Batch work completes.

Set the administration activities up as a batch job to start towards the end of the expected batch work schedule. Specify that `CLOAD=0` as a pre-condition for the administration job. If there are more than one administration jobs to be run, set them up as a chain of jobs, with only the first one dependant on `CLOAD`.

5.5.2 Controlling peak activity

The variables `STARTLIM` and `STARTWAIT` were created to prevent a large number of jobs swamping a machine or network by starting at the same instant. For example: if a user scheduled 400 network I/O intensive jobs to start at Midnight, it is likely that network problems would ensue.

Any process tends to use a large amount of system resources when starting up. If you observe any resource being swamped then lower the value of `STARTLIM` and/or increase the number of seconds delay specified by `STARTWAIT`.

On high performance machines `STARTLIM` may be increased and/or `STARTWAIT` reduced. The slowest components may well be any networked or I/O bound resources.

5.5.3 Job Logging via LOGJOBS

This variable may only be set to a string value. It should contain a file name, or a program or shell script name starting with a "`|`". However, it is vitally important to use "`|`" with great care so as to ensure the scheduler process cannot be held up by the receiving process.

If a file name is given, it will be taken relative to the spool directory, by default `/usr/local/var/gnubatch`. Thus a file name of `joblog` will be interpreted, if the spool directory hasn't been changed, as `/usr/local/var/gnubatch/joblog`.

The file access modes on the file will correspond to the *read/write* permissions on the variable, (execute permissions will not be set) and the owner and group will correspond to that of the variable, usually `batch`.

If a program or shell script is given, then the `PATH` variable which applied when the scheduler was started (this may be from the terminal of the user who ran gbch-start) will be used to find the program.

Lines written to the file or sent to the program will take the form

```
03/05/99|10:22:43|13741|date|completed|jmc|users|150|1000
```

Each field is separated from the previous one by a |, for ease of processing by awk etc. The fields are in the following order (new versions will add fields on the right):

Date	In the form *dd/mm/yy* or *mm/dd/yy* depending on the time zone.
Time	
Job Number	Prefixed with host and colon for external jobs
Job Title	or if no title <unnamed job>
Status code	(listed below) Prefixed by host and colon if from remote host
User	
Group	
Priority	
Load Level	

The status codes may be one of the following:

`Abort`	Job aborted
`Cancel`	Job cancelled
`Chgrp`	Group changed
`Chmod`	Mode changed
`Chown`	Owner changed
`Completed`	Job completed satisfactorily
`Create`	Job created (i.e. submitted to queue)
`Delete`	Job deleted
`Error`	Job completed with error exit
`force-run`	Job forced to start, without time advance
`force-start`	Job forced to start
`Jdetails`	Other details of job changed
`Started`	Job started

5.5.4 Variable Logging via LOGVARS

This variable may only be set to a string value. It should contain a file name, or a program or shell script name starting with a "`|`".

If a file name is given, it will be taken relative to the spool directory, by default `/usr/local/var/gnubatch`. Thus a file name of `varlog` will be interpreted as the file name `/usr/local/var/gnubatch/varlog`.

The file access modes on the file will correspond to the *read/write* permissions on the variable, (execute permissions will not be set) and the owner and group will correspond to that of the variable.

If a program or shell script is given, then the `PATH` variable which applied when the scheduler was started will be used to find the program, possibly that of the user who ran gbch-start. Lines written to the file or sent to the program will take the form:

```
03/05/99|09:52:43|cnt|assign|Job start|jmc|users|2011|86742|myjob
```

Each field is separated from the previous one by a "`|`", for ease of processing by awk etc.

The fields are in the order(new versions will add fields on the right):

Date	in the form *dd/mm/yy* or *mm/dd/yy*, depending on time zone.
Time	
Variable Name	
Status code	(listed below) Prefixed by machine: if from remote host
Event code	see below.
User	
Group	
Value	numeric or string
Job number	If done from job
Job title	if done from job

The status codes indicate what happened, and may be one of the following:

`assign`	Value assigned to variable
`chcomment`	Comment changed
`chgrp`	Group changed
`chown`	Owner changed
`create`	Variable created
`delete`	Variable deleted
`rename`	Variable renamed

The event code shows the circumstance in which the variable was changed, as follows:

`manual`	Set via user command or operation
`Job start`	Assigned at job start
`Job completed`	Assigned at job completion
`Job error`	Assigned at job error exit
`Job abort`	Assigned at job abort
`Job cancel`	Assigned at job cancellation

6 Jobs and related entities

To execute a job GNUbatch invokes the specified command interpreter and "pipes" the text of the job to the standard input of the command interpreter. The most common types of batch job are shell scripts. Any program which will read instructions from standard input may be set up as a command interpreter for use by GNUbatch.

The text of a job is often referred to as the job, job file or commands. To avoid confusion the use of the word *script* is now being encouraged. The script for each batch job may invoke other programs, compiled or shell script, as it would if it was run from the command line. The term job file is still used to describe the file of an unqueued job which holds the job script.

The set of parameters held by GNUbatch governing what it should do with the job is often called the command file. This is now being referred to as the job specification. The term command file is used to refer to the file of an unqueued job which holds the specification.

Apart from variables, which are described in their own chapter, jobs also have relationships with two other entities. These are:

- The command interpreter under which the job actually runs. All jobs have a specified command interpreter.
- A queue, which provides a grouping mechanism for jobs. All jobs belong to a queue. This is not always obvious since jobs which do not specify a queue are associated with the null queue, which has no name.

These entities are discussed at the end of this chapter, as well as in the sub-sections which describe how each job specifies relationships with them.

6.1 Time

Jobs can be specified without any scheduling time specifications. In this case they will run as soon as possible, just like jobs run under the standard Unix batch command. Once such a job has run it will be deleted from the queue.

If a job has a time specification it will always have a *Scheduled Run Start Time* and flags to indicate if the job is to be: deleted, retained or repeated after completion.

Repeating jobs have additional options, such as a specification of any days to avoid, which do not have any impact on the first *Scheduled Run Start Time*.

The intricacies of the time options are explained below:

6.1.1 Scheduled run start time

The time at which a job is scheduled to start can be specified by date and time to the nearest minute. GNUbatch starts jobs on the minute boundary, unless they are prevented from doing so by some condition. When a job has a time specified, it may be set up to be deleted, retained in a done state or repeated, automatically after the first run.

The user interface programs accept and understand years specified as 4 digit numbers, hence avoiding any problems over the year 2000.

If a job is due to run for the first time then it will always wait, if blocked by some condition, until that condition is satisfied. Once all conditions are satisfied the job will run immediately.

When a job is blocked from repeating by some condition, there are a range of behaviours that it can follow. The options for these behaviours are described in the sub-section on Repetition.

6.1.2 Retention

Jobs may be set up to run once at a specified start time and then be retained on the queue after execution. Once the job has run its progress state is set to `Done`.

A job that is in the `Done` state may be set running at any time by a suitably authorised user or program. Similarly the specification of the job may be changed. For example a new run could be scheduled and possibly some repetition specified.

6.1.2.1 Auto delete after execution

An automatic delete time may be specified if the job has been retained, either with the "retain on completion" or one of the repeat options. The job will be deleted automatically after the specified number of hours. The default is zero hours, which will retain the job indefinitely (this maintains backward compatibility with earlier versions of GNUbatch).

Each time the job runs, the timeout period will restart.

6.1.3 Repetition

Jobs can be specified to automatically repeat at regular intervals after the initial run. The interval is specified as an integer number of a particular unit of time. The available units are:

`Minutes`	Minutes (All repeating jobs attempt to start on the minute)
`Hours`	Hours (This is the default as distributed)
`Days`	Days
`Weeks`	Weeks
`Monthsb`	Months relative to the beginning. This specification requires two integer values: the first is the interval in months and the second is the day of the month on which to run.
`Monthse`	Months relative to the end of the month. This specification requires two integer values: the first is the interval in months and the other is the day on which to run.
`Years`	Years in the range 1 to 99.

With the exception of `Monthsb` and `Monthse` the first scheduled repetition is calculated by adding the repeat interval to the initial run time. The repeat specification is often represented in the format unit:number on the command line of programs like gbch-r. For example:

```
Minutes:10
```

run every ten minutes

`Weeks:2`

run at the same time and day, once a fortnight

`Days:5`

run at the same time of every fifth day

Repeating jobs have additional options which can be set to indicate what to do if a job fails and to specify any days to avoid. These are explained in detail later.

6.1.3.1 Monthly Repeat Intervals

For `Monthsb` and `Monthse` the first scheduled repetition will be at the same time of day as the initial run. The month for the repetition is calculated from that of the initial run, but the actual day of the month is specified as a separate parameter. If a *day of month* is not specified when the repetition is set up, GNUbatch takes the day of the initial run as the *day of month*.

Monthly repeat specifications are often represented in the form *`unit:number:day`* in the command line options of programs, like gbch-r. For example:

`Monthsb:1:5`

specifies that the job will be run on the 5th day of every month. The repeat specification may also include one or more days to avoid. If this is the case and the scheduled repeat falls on one of these, the job is put back until the same time on the first acceptable day.

The repeat option for days relative to the end of the month has to take into account the number of days in each month. To specify how many days from the end of the month is required, the month given in the next scheduled time is taken, for example if the month in the next scheduled time has 31 days, then to specify the last day of each month use:

`Monthse:1:31`

or to specify the next to last day of the month use:

`Monthse:1:30`

If the next scheduled start time is in February and not a in leap year, then these should be:

`Monthse:1:28`

or to specify the next to last day of the month use:

`Monthse:1:27`

to achieve the same effect.

With "months relative to the end", if "days to avoid" is set, then earlier days in each month are selected until an acceptable day is found, whereas with all other repetitions the next repeat is put back.

Specifying "months relative to the end" with a day of 5 or less, or "months relative to the beginning" with a day of 27 or more is probably a mistake.

Exceptionally, the web browser interface takes a number representing the day of the month in either case with 1 representing the last day of the month with "months relative to the end", 2 the next to last etc.

We would appreciate feedback as to which convention is preferable.

6.1.3.2 Days to Avoid

The repeat specification has options to specify one or more days of the week and holidays to be avoided when scheduling the next run of a job. The holidays are marked in the holiday table, which is described later in this chapter. Up to 6 days of the week can be set to be avoided.

When the next repetition of a job is calculated the scheduler will step past any days to avoid. For example a job that runs at 3 minutes past the hour every hour, but avoiding Saturdays and Sundays will run at 23:03 on Friday night and then next at 3 minutes past midnight on Monday morning.

The days to avoid parameter does not affect the initial run time. Hence a job can be submitted to run the first time on a Saturday, but avoid Saturday and Sunday thereafter.

Upon installation the default abbreviations for the days to avoid are: `Sun`, `Mon`, `Tue`, `Wed`, `Thu`, `Fri`, `Sat` and `Hday`. The `Hday` refers to days to avoid as specified in the scheduler's Holiday file.

6.1.3.3 Time adjustments on error

The time adjustment parameter specifies whether the job's scheduled start time should be left in the past or set to the next repetition in the event of the job failing.

Specifying that the start time is not to be advanced to the next repetition, allows errors to be corrected and the job restarted. Select the advance time on error option, when a problem can or need not be rectified until the next repetition is due.

6.2 Job Completion Messages

GNUbatch can send messages to the owner of a batch job, for example when it finishes or fails. These messages can be directed to e-mail, the users terminal session if logged in, or disabled as part of each jobs specification. The options are:

- Discard all messages.
- Write messages to the job owner's terminal, if they are logged in. Otherwise e-mail the messages back to them.
- E-mail messages to the job owner.

Do not confuse messages from the scheduler about a job with the output from the job. which is handled differently.

The message handling can be modified on a per user or system wide basis. The same flags are used but the scheduler can be configured to behave differently, when writing or e-mailing messages. This is not part of the job specification and is described under the chapters on Configurability and Extensibility.

6.3 Redirection of Input and Output

By default any standard output and standard error produced by a job is spooled to a temporary file, then e-mailed back to the job owner on completion. If no standard input is specified for a job it will just hang.

The job specification includes redirection options to specify Unix files for input and output. These work in the same way as redirections in a shell, and have a similar syntax to the popular shell programs.

A redirection can have:

- *File number*, the file descriptor to be used. If omitted, standard input is assumed for input, standard output for output.
- `<` indicating input, or `>` indicating output or `>>` append to output or `|` output to pipe, standard input for following shell command.
- `<>` indicating that the following file would be opened in read/write mode, or `<>>` for read/write/append mode.
- `<|` for input from the standard output of the following shell command.
- `&File_number` indicating dup from that file number, or `&-` meaning close the file number first specified.

Examples of redirections are

```
<Cfile
>>Output
2>Error
2>&1
|lp -d laser
2|grep error >Errs
```

Note that in the last two cases there are further redirections involving a pipe or output file which are interpreted by the shell. The last example causes the standard error to be passed to the standard input of grep.

Symbols for meta-data can also be included in the path/file names used in redirections. See the section on Meta-Data for a list of the available data.

6.4 Arguments

Programs that are run from the command line are often given options or arguments on the command that invokes them. For example the file list program ls may be given the options -al to specify that all files should be included in a report of long format:

```
ls -al
```

A job script may also take arguments and options. These are held in an argument list as part of the job specification. For example a script called `update`, may be run from the command line with arguments like this:

```
update -C all sales
```

In a batch job these would be held in the argument list as separate arguments. How the arguments are separated is up to the owner of the job. For example `-C` and all could be treated as separate arguments or a single argument.

The gbch-r command

```
gbch-r -a '-C' -a 'all' -a 'sales' update
```

(none of the quotes are necessary in this instance, but help the reader to follow the difference between gbch-r arguments and arguments to the job) will produce an argument list of:

```
-C
all
sales
```

Symbols for meta-data can also be included in arguments. See the section on Meta-Data for a list of the available data.

6.5 Environment & process parameters

A copy of the environment in effect when jobs are submitted is saved as part of their specification. The job will be run using this environment. The various elements of this environment can be re-specified as required.

Jobs submitted from an alien platform, such as a PC running Windows, will be given a default environment. This default can be configured on the submitting machine.

6.5.1 Environment Variables

The job specification holds a list of environment variables that are set up in the job's environment each time it is run. At run time the scheduler first sets up any environment variables that are specified in the `/usr/local/etc/gnubatch-env` file. The variables from the job specification are then added to the environment.

Symbols for meta-data can also be included in the values of environment variables. See the section on Meta-Data for a list of the available data.

Note that if you have a large number of environment variables, the standard set should be placed in `/usr/local/etc/gnubatch-env`. The job will contain the differences between those variables and the variables set with the job.

6.5.2 ulimit and umask

The ulimit and umask parameters may be applied to a batch job. By default the values are taken from the values in force when the job is submitted.

ulimit specifies the maximum file size, in blocks, that can be written by the job.

umask affects the default permissions of files created by the job. It is usually represented as an octal number, the same as file permissions. For example if umask is `022` the write permission will be turned off for *Group* and *Other* on any files created by the job.

6.5.3 Working Directory

By default GNUbatch assumes that a job is to be run in the same directory as it was submitted from. This is held in the job specification and any alternative directory may be specified, either when the job is specified or later. Take care not to specify a directory which does not exist or for which the owner of the job has insufficient permissions.

The `~` notation for users' home directories, such as `~jmc` is expanded. Without a user name, `~` on its own represents the home directory of the job's owner.

6.5.4 Normal and Error Exit Codes

Jobs like all Unix processes will return an exit code on completion. Usually an exit code of zero indicates the process performed its tasks successfully and exited normally. Any other exit code usually indicates an error of some kind. Exit codes are integers in the range 0 to 255.

Not all programs follow this convention, however, so job specifications include two parameters to interpret exit codes.

The *normal exit* range parameter specifies a range of exit code values that the scheduler should interpret as a normal exit after a successful run. When specifying a *normal exit* range on the command line of a program like gbch-r the parameter would look like this:

```
N0:0
N0:9
```

The first example indicates that only 0 represents a normal exit. The second indicates that the exit codes 0 to 9 are normal.

The *error exit* range may be set using the other parameter, which on the gbch-r command line would look like:

```
E17:255
```

If an exit code does not fall inside either the *normal exit* or *error exit* ranges the job is considered to have been aborted.

If an exit code falls inside both ranges, then the smaller of the two ranges will apply. For example, if the ranges are:

```
N0:10
```

and

```
E1:255
```

an exit code of 1 to 10 will fall inside both ranges but will be treated as *normal* since the *normal* exit range is smaller.

6.5.5 Network Scope

When two or more machines are running GNUbatch in co-operation with each other the scope of jobs becomes relevant. There are three alternatives, which are:

Local
: Specifies that the job is visible and accessible only on the machine to which it was submitted.

Export
: Specifies that the job must run on the local host, but allows the job to be seen and managed from any networked GNUbatch host.

Full Export (remote runnable)
: Enables the job to run on any co-operating GNUbatch host as well as being visible and manageable by the remote hosts.

6.5.6 Time-out parameters for stopping Runaway Jobs

There are three parameters that specify how to identify and stop a runaway job. They are:

1. The maximum elapsed time since starting that a job may run for until it is terminated by the scheduler. This is specified as a number of seconds. The default is 0 seconds indicating that the job may run unchecked.

2. What signal to send an over-running job in order to terminate it. The job should trap anything other than a `SIGKILL` and respond by tidying up and exiting cleanly. The signal is specified by number rather than by name.

3. A grace time, again in seconds, within which the job should terminate after being sent a signal. If the job does not terminate itself within the specified grace time the scheduler will kill it with a `SIGKILL`.

6.6 Owners, Groups and Modes

Each job belongs to a user and a Unix group. Access to jobs is controlled by a set of permissions, called modes, similar to those on ordinary Unix files.

6.6.1 Owners and Groups

The job specification includes the *user* who owns the job and the Unix *group* that the job belongs to. The *owner* and the *group* are normally taken as those of the user who submitted the job. A different user and group may be specified when the job is submitted, but only if the submitting user has *write admin file* privilege.

Suitably authorised users may change the owner and group of a job when it is on the queue. An administrator may do this in one operation. Ordinary users, may be given sufficient privilege to change the specification. In this case the current owner has to specify who the job is to be given to and then the recipient must accept it.

These security features prevent un-authorised transfer of jobs to and from more privileged owners and groups such as the `root` user.

Only the primary groups of users are considered for evaluating access permissions to jobs.

6.6.2 Modes

Access to jobs is controlled by the Modes which are similar to Unix file permissions, but with greater functionality. Permission to each access mode is granted to the owner of the variable (User), users in the same primary group (Group) or everyone (Others).

Here is a screen, from the interactive batch queue management tool gbch-q, showing the modes for a job called `update`.

```
Modes for Job `update'
Job owner wally group staff

                       User   Group Others
Read                   Yes    Yes   No
Write                  Yes    No    No
Reveal                 Yes    Yes   Yes
Display mode           Yes    Yes   Yes
Set mode               Yes    No    No
Assume ownership       No     No    No
Assume group ownership No     No    No
Give away owner        Yes    No    No
Give away group        Yes    Yes   No
Delete                 Yes    No    No
kill (jobs only)       Yes    No    No
```

The various modes give the following type of access when permission is granted:

`Read`	The job specification and script contents can be read.
`Write`	The job specification may be modified, and read permission is conferred automatically.
`Reveal`	Jobs are completely hidden from users without reveal permission. If reveal permission is granted but not write permission then only the job id number, owner and group may be seen.
`Display mode`	Allows these modes to be viewed.
`Set mode`	Allows these modes to be changed.
`Assume ownership`	Dictates to whom ownership of a job may be transferred relative to the current owner.
`Assume group ownership`	Dictates to which group a job may be transferred relative to the current group.
`Give away owner`	Grants permission to transfer the ownership of a job to another user.
`Give away group`	Grants permission to transfer the job to another group.
`Delete`	Permission to delete job from the queue.
`Kill`	Allows jobs to be killed, or sent some other signal.

6.7 Job Identifiers - Queues, Titles and Job ID numbers

Each job has a unique job id number, also called the job number or jobno. This is an unsigned integer, generated by the scheduler when the job is submitted. The job number is used whenever jobs have to be identified unambiguously.

The job specification also includes a title, providing a more user friendly means of identifying jobs on the queue. This title is specified and editable by users and so can not be guaranteed to be unique. It should be used to help users recognise jobs.

As part of the title specification, a job can be associated with a queue. Each job may only belong to one queue and a queue may hold many jobs. Queues and their uses are described later in this chapter.

6.8 Priority and Command Interpreter

Priority, load level and command interpreter are loosely related in that they indicate the importance and impact on the system of a job.

All jobs are run under a command interpreter, which is referred to by name in the job specification. Command interpreters are separate entities which specify a default load level for jobs submitted to run under them. See the section on command interpreters later in this chapter for more information.

The load level is held as an unsigned 16 bit integer. It specifies the relative impact that a job is likely to have on the loading of the host machine.

Priority is specified as an integer in the range 1 to 255, and controls how likely a job is to be run ahead of other jobs in the queue. If there were no conditions on jobs then they would all run as soon as their start time arrived.

In reality, due to conditions set by the variable `LOADLEVEL`, there may be more jobs ready to start than the system will allow. In this case jobs with the higher priority get started ahead of lower priority ones. When the maximum number of jobs are running, those that did not get in have to wait until one or more of the higher priority jobs finish before being started.

6.9 Job control variables - Conditions and Assignments

Dependencies between jobs, and other parts of the system, can be implemented using variables. The job specification holds two lists of relationships between a job and the variables. One list specifies conditions which must be true before GNUbatch will allow the job to start. The other list specifies assignments that GNUbatch will perform on the data held in variables when a job starts, stops or fails.

6.9.1 Conditions

A condition is a simple expression that compares the value of a variable with a literal string or integer constant. The scheduler will not start a job unless all of the conditions are satisfied, i.e. the expressions return a value of true. Up to 10 Conditions may be specified for each job. The expression has the following four components:

1. A *variable name*, which may be any variable readable by the user, including variables on remote machines represented as machine:variable.
2. A *comparator*, which may be any of `=`, `!=`, `<`, `<=`, `>` or `>=`.

   Remember to enclose these sequences in quotes when using them in a shell command.
3. A *constant*, which can be a string or an integer (negative or positive).

4. A *critical flag*, which determines whether the condition should be ignored if it involves an inaccessible remote variable.

For example,

```
Update_Count<17
BackUp_State!=Done
voyager:Update_Count>2
```

Where a condition refers to a variable on a remote machine, there is always the possibility that the remote machines copy of the scheduler is not running or disconnected. To handle this the condition has the option to specify whether the condition is critical or not.

If the condition is specified as *critical* the job has to wait until the machine is available, and the variable satisfies the expression, before running.

Alternatively if the condition is specified as *non-critical* and the machine is not available, the condition will be ignored.

6.9.2 Assignments

Up to 8 assignments may be specified for a job. Each assignment specifies what operation to perform on a variable and under what circumstances to perform the operation. The operation is specified as a simple programming language like assignment statement, hence the name assignment. The circumstances are defined by a set of flags; all, one or more of which may be set.

There are two special cases of the assignment. One performs a straight assignment of the exit code with which the job terminated, to a variable. The other does the same thing with the signal number, if the job was terminated on a signal, either by a kill command, a signal from gbch-q etc, or a program fault.

6.9.2.1 Flag Options

There are six flags to specify when an assignment should be performed. At least one flag must be set. They can be used in combination or all set as required. The flags are usually represented by a single letter, as follows:

Letter	Operation specified
`S`	Start, the scheduler performs the assignment when it starts the job
`N`	Normal exit, performs the assignment on normal exit
`E`	Error exit, performs the assignment on error exit
`A`	Aborted, performs assignment if job aborted (signal)
`C`	Cancellation, performs assignment if job cancelled
`R`	Reverse the specified assignment for everything except S.

The `R` flag, is only relevant one or more of the exit flags is set. It undoes whatever assignment was (or would have been if S is not specified) performed at the start of the job, when the job finishes.

If all the flags, `SNEACR`, are set then the assignment is performed on start up, and reset when the job finishes however it exited. If only the flags `SNR` are set then the assignment is only reversed when the job finishes normally.

Remember that you can adjust what exit codes constitute "normal" and "error" exits, as described earlier.

6.9.2.2 Assignment Operation

Each assignment statement has five components, which are:

- A *variable name*, which must already exist and be *writeable* by the user. To access an exported variable from a remote machine, prefix it with the machine name and a colon.
- An *assignment operator*, which can be as follows:

`=`	Assign constant to variable
`+=`	Increment value of variable by constant
`-=`	Decrement value of variable by constant
`*=`	Multiply value of variable by constant
`/=`	Divide value of variable by constant
`%=`	Take modulus of variable (i.e. remainder when divided by the constant)

- A *constant*, which may be a string or numeric value. Only the = operator is valid for assignments with strings.
- A *critical flag* to determine whether the assignment should be ignored if the host is offline.
- Assignment flags for the start and end of jobs.

Here are some examples of assignments:

```
count+=1
status=error
mach2:log+=3
```

In the case of assignments from an exit code or signal number only the plan assignment is provided. The keyword exitcode or signal is used in the statement instead of a constant. For example:

```
status=exitcode
killed_by=signal
```

If the R flag is set, to reverse a start assignment, the assignments performed are:

Operator	**Reverse implies**
`=`	Assign zero to an integer value or an empty string to a string value.
`+=`	`-=` decrement value by same constant used for increment.
`-=`	`+=` increment value by same constant used for decrement.

Operator	**Reverse implies**
`*=`	`/=` divide value by same constant used for multiplication (ignoring any remainder).
`/=`	`*=` multiply value by same constant used for division.
`%=`	Unchanged, since this operation does not have a meaningful complement.

Note that reverse assignment may still be applied if the job has no start assignment flag to be reversed. The operation is still applied in the "reversed" state as above as if the start condition had applied. However it is recommended that you avoid this.

Where an assignment operates upon a variable on a remote machine, there is a possibility that the remote machine's copy of the scheduler is not running or disconnected. To handle this the assignment has the option to specify whether the operation is critical or not.

If the assignment is specified as *critical* the job has to wait until the machine is available, for the operation to be performed, before running.

Alternatively if the assignment is specified as *non-critical* and the machine is not available, the job will be run without performing the specified operation.

Once a job is running the critical specification has no effect. If a remote variable becomes unavailable during execution of a job, any critical job completion assignments to that variable are ignored.

6.10 Meta-Data

There are several useful parameters from the job specification that can be substituted into arguments, environment variables and I/O redirections. These are:

`%s`	Command Interpreter name
`%t`	Job title
`%U`	User name
`%G`	Group name
`%N0`	Host name where job originated
`%d1`	Job number, in decimal
`%d2`	Priority, in decimal
`%d3`	Load Level, in decimal
`%x1`	Job number, in hexadecimal
`%x2`	Priority, in hexadecimal
`%x3`	Load Level, in hexadecimal
`%%`	To insert a single % character.
`` `cmd` ``	Insert first line of output of cmd.

The substitution is performed at run time, making sure that the information is up to date.

For example to output a standard banner containing the Title and Job ID number, each parameter could be set up in an environment variable. If the environment variables are:

```
JOBNUM=%d1
JOBNAME=%t
```

then a simple piece of shell script to use them might look like this:

```
cat <<endbanner
********************************************
Output from Job: $JOBNAME
Job ID number:   $JOBNUM
********************************************
endbanner
```

The output can be re-directed to a unique file by using the job number like this:

```
1>/joblogs/jobnum%d1
```

6.11 Command Interpreters

The command interpreters are separate entities which are referred to in the job specifications. Each command interpreter specifies the following set of parameters:

Name	A unique identifier which is used, both internally and by user programs, to refer to the command interpreter.
Program	Holds the full path name of the command interpreter program. This can be any program, such as the Bourne shell, usually `/bin/sh`, or the Korn shell, usually `/bin/ksh`, that will read commands from standard input.
Arguments	Specifies any "predefined" arguments that are to be passed to the command interpreter when it is invoked, preceding any arguments which are given to the job. This is very commonly set to -s for shells, which directs the shells not to interpret the first actual argument as a file name.
Load Level	Sets the default Load level to be given to all jobs running under the command interpreter. Only users with the *special create privilege may* override this default for a job.
Nice	Sets the Unix nice value for processing batch jobs under this command interpreter. The default is 24. Remember that increased priority is denoted by a low nice value.
Argument 0	When getting a list of processes using a command like ps, the batch jobs will normally have the name of the command interpreter program they are running under. Setting the Argument 0 option causes the job title to be used as the process name instead. This may confuse some programs, hence it is made an option.
Expand args	Arguments with $ in are expanded by GNUbatch, rather than by the command interpreter. This may be desirable in cases where the command interpreter is not a shell, or where the semantics of $ signs in arguments is different than that of the shell.

The same program can be used by more than one command interpreter, for example with low nice values (and hence a high priority) and a high load level or vice versa.

Be careful about using the **Expand Args** flag with shells and in conjunction with arguments specified in the job and environment variables with quotes etc in. This is because most shells identify syntax before expanding variables. So for example if argument 1 contained a single quote and argument 2 contained `My String`, then

```
echo $1$2$1
```

would print

```
'My String'
```

having identified the syntax first and decided that there are no quotes, but if the arguments were expanded the shell would never "see" the $1 and $2 constructs and the output would be

```
$2
```

As the quotes would "protect" the $ from expansion.

6.12 Queues

All of the jobs on a GNUbatch host run in the same physical queue. There is however a mechanism of virtual queues, referred to simply as queues, that can be used for grouping jobs together. GNUbatch does not impose any structure or operations on these queues. It provides mechanisms to restrict the view of and selectively query the physical queue by virtual queue names.

This is enough, combined with the configurability and extensibility options of GNUbatch to provide sophisticated management of queues of jobs.

Each job is associated with just one queue. If no queue is specified then the job is said to be in the null queue. A view or query can be restricted to a set which contains one or more queues, and which may include or exclude the null queue.

The set of queues may be one name, a list of names or a list of patterns for matching queues. It may be advisable to use quotation marks around the queue specification when invoked from a shell command. They may be given as a comma-separated list of alternatives, including the use of shell-style wild-cards. For example

```
test
```

would restrict the view to just the queue test.

```
"dev_a,test1,test2"
```

would select the three separately named queues: dev_a, test1 and test2.

```
"dev*,test[3-7]"
```

select jobs in any queue whose name starts with the string "`dev`" and jobs in queues `test3`, `test4`, `test5`, `test6` and `test7`.

The wild-card options are:

`*`	Matches anything
`?`	Matches one character
`[a-mp]`	Matches one character in list or range
`[!n-zq]`	Matches one character not in list or range

6.12.1 Examples

It is important to devise a naming convention for queues that reflects the structure of the batch processing. It is also a good idea to use the same naming conventions for job control variables related to particular queues. The queue name could be incorporated into the variable name.

For example variables associated with jobs in queue `abc` could be named as follows:

```
abc_progress
```

could be used to control the progress of a chain of jobs in queue `abc`.

```
abc_count
```

to indicate how many times the chain has run.

The queue name `abc` is not very meaningful. It is best to choose names which describe the family of jobs in each queue. A queue of jobs that handle wages could be called `Payroll`. Since the word `Payroll` has been used for the queue name it can be left out of individual job titles. A job originally called

```
Check run for Payroll
```

could be put in queue `Payroll` and titled

```
Payroll:Check run
```

Queue names can indicate more than just a simple functional grouping. By taking into account how queues can be selected using wild card characters, much more complex groupings can be implemented.

6.12.1.1 Naming Conventions for Overlapping Sets

Queue names can reflect the business area which jobs relate to with names like: `sales`, `lease`, and `returns`. Alternatively they can indicate the developmental state of jobs with names like: `dev`, `test`, and `live`.

The selection mechanisms enable queues to represent jobs as belonging to overlapping groups of activity. For example: a suite of jobs currently being developed might be in the queue `dev_sales`. When ready to hand over to operations the queue name would be changed to `test_sales`. Once the jobs are ready for production work they would be moved again to `live_sales`.

Typical selections would be:

```
*sales
```

for all jobs in a queue relating to `sales`,

```
live*
```

forall jobs in queues for production jobs,

```
dev*,test*
```

for jobs in the development and test queues,

```
*sales,*lease
```

for all job relating to sales and leasing.

6.12.1.2 Naming Conventions for Sub-Queues or Hierarchies

Jobs may be related in a structure that resembles a family tree. This structure can be reflected in the queue naming conventions adopted for these jobs.

For example all jobs related to customer accounts might be in a queue called `cust`. Within that queue the jobs could be broken down into smaller units indicating what type of task each does, by a suffix. Jobs that update customer records may have a suffix of `_ch` and those that generate reports could have `_rep`.

Even finer resolution could be obtained by an additional suffix, perhaps indicating what customer information is being used: Addresses could be indicated by `_addr` and credit limits by `_cred`.

Used with meaningful job titles a list of jobs in the queue cust* might look like this:

```
cust_ch_addr:Fred Smith
cust_rep_mktg:MailshotList
cust_new:Bloggs Builders
cust_rm:Inactive Accounts
cust_ch_cred:Xi Software Ltd
cust_rep_addr:Scotland
cust_rep_bal:OverCreditLimit
```

Restricting the selection to `cust_rep*` would give only:

```
cust_rep_mktg:MailshotList
cust_rep_addr:Scotland
cust_rep_bal:OverCreditLimit
```

6.13 Holidays

Holidays are an 8th type of *day to avoid* for repetitions. It relates to a single table of holidays for the current, past and present years. A system administrator can set up, and any user can view, the table of holidays for the year using the programs gbch-q, gbch-xq and gbch-xmq.

The repeat time for a job is calculated at the previous run time. Any changes made to the table of holidays are not automatically taken into consideration by pending jobs. The changes only take effect after the job is next run.

7 Internal Programs and file formats

The following lists the internal programs and file formats used by GNUbatch. With a few exceptions these programs should not be invoked or accessed by a user including an administrator.

The internal programs are all held in the same directory, `/usr/local/libexec/gnubatch` by default. If the default directory is not used it will be pointed to by the `SPROGDIR` environment variable set up in the `/usr/local/etc/gnubatch.conf` file.

7.1 Core Programs

7.1.1 btsched

```
/usr/local/libexec/gnubatch/btsched [-options]
```

btsched is the scheduler process for the **GNUbatch** batch processing system.

It is normally invoked by the system startup routines, or otherwise by gbch-start.

It may take certain options from the command line, but these are normally passed to it by gbch-start and are not documented here as they are part of the internal interfaces of **GNUbatch** and are subject to change.

Information, either in respect of other machines to connect to, or pre-existing jobs and variables on the current machine, are read respectively from the files `/usr/local/etc/gnubatch.hosts` and the directory `/usr/local/var/gnubatch` (unless changed via the *master configuration file* `/usr/local/etc/gnubatch.conf`).

A "slave" btsched process is spawned to control running jobs, and if a networked version of **GNUbatch** is being run, then an additional "slave" btsched process is spawned to monitor and process incoming network messages.

Incoming remotely-submitted jobs and API interfaces are handled via a separate process (also invoked by gbch-start), xbnetserv.

7.1.1.1 Files used

`/usr/local/etc/gnubatch.hosts`	Host names and descriptions
`/usr/local/etc/gnubatch.conf`	Master configuration file
`/usr/local/share/gnubatch/help/btint-config`	Message file
`/usr/local/var/gnubatch`	Spool directory
`/usr/local/var/gnubatch/btsched_jfile`	Job file
`/usr/local/var/gnubatch/btsched_vfile`	Variables file
`/usr/local/var/gnubatch/btsched_reps`	Error log file
`/usr/local/var/gnubatch/btufile`	User data
`/usr/local/var/gnubatch/btmm_jobs`	Job memory-mapped file
`/usr/local/var/gnubatch/btmm_vars`	Variables memory-mapped file
`/usr/local/var/gnubatch/btmm_xfer`	Communication buffer memory-mapped file

7.1.1.2 IPC Facilities used

An IPC message queue, with key `0x5869b000` and owned by user `batch` is created by btsched and used to receive messages from user processes and pass instructions to and internal messages from the slave btsched processes to the master.

Two shared memory segments are created to hold details of jobs and variables. As the shared memory facility provides no facilities for growth, then additional shared memory segments may be created if the job and variable lists expand sufficiently and the original ones deallocated.

A further shared memory segment, with key `0x5869b100` is created to hold details of pending jobs before transfer to the main shared memory segment.

The keys given to the shared memory segments start at `0x5869b002` and ascend upwards to `0x5869b064` before wrapping around.

If built to use memory-mapped files rather than shared memory, the files are held in the spool directory, by default `/usr/local/var/gnubatch`, and have the names `btmm_jobs`, `btmm_vars` and `btmm_xfer`.

A set of at least 10 semaphores, with the key `0x5869b001` is created to interlock access to the shared memory segments, and a further set of 3 semaphores with the key `0x5869b003` is created to interlock network processes.

The presence or absence of these IPC facilities is used by btsched and other programs to determine whether a previous copy of itself is running. If btsched is abnormally terminated, it may be necessary to delete these IPC facilities before btsched can be restarted.

The utility gbch-ripc may be used to delete the IPC facilities quickly.

7.1.1.3 Internet ports used

Btsched accepts and sends interconnections from other machines on TCP port, passes the contents of batch jobs on a further TCP port, and undertakes "probes" on a UDP port.

The port numbers are set up in the `/etc/services` file when **GNUbatch** is first installed.

7.1.2 xbnetserv

`/usr/local/libexec/gnubatch/xbnetserv`

Xbnetserv is the remote server process for the **GNUbatch** batch scheduler system.

It serves 3 purposes

1. It accepts jobs from other hosts submitted by gbch-rr.
2. It accepts jobs and administration requests from DOS and Windows machines.
3. It supports API operations.

It is normally invoked by the system startup routines, or otherwise by gbch-start.

It takes no arguments from the command line (and ignores any which are supplied). Information, in respect of other machines to connect to is read from the file `/usr/local/etc/gnubatch.hosts`.

7.1.2.1 Internet ports

xbnetserv uses 2 ports

```
gbatch-netsrv
```

(N.B. no second "`e`") to accept incoming jobs on TCP from gbch-rr and on UDP from the DOS/Windows interface, and 2 further ports

```
gbatch-api
```

to communicate with the API library.

The port numbers are set up in the `/etc/services` file when **GNUbatch** is first installed.

7.1.2.2 Diagnostics

xbnetserv runs as a "daemon process" and diagnostics, apart from those detected when it is first started, are not usually written to any terminal but to the file `/usr/local/var/gnubatch/btsched_reps`.

In the event of any problems this file should be examined.

7.2 btexec

```
/usr/local/libexec/gnubatch/btexec options
```

Btexec runs commands for macros under the identity of the invoking user. This is required because gbch-q, gbch-xq and gbch-xmq are set-user programs (to other than `root`) and there is an inherent security breach in many versions of Unix in that such programs cannot divest themselves of traces of the set-user user id.

This program is only intended for internal use and is not further documented.

7.3 Bgtksave

```
/usr/local/libexec/gnubatch/bgtksave options
```

Bgtksave is invoked by the GTK programs gbch-xq gbch-xr and gbch-xuser to save options as required by the user. This is required because they are are set-user programs (to other than `root`) and it is needed to assume the identity of the original user in cases where it is needed to update files in the user's home directory.

This program is only intended for internal use and is not further documented.

7.4 Bgtkldsv

```
/usr/local/libexec/gnubatch/bgtkldsv options
```

Bgtkldsv is invoked by the GTK program gbch-xr to load and save job files as required by the user. This is required because gbch-xr is a set-user programs (to other than `root`) and it is needed to assume the identity of the original user in cases where it is needed to update files in the identity of the user (which GTK forbids being different from the effective user).

This program is only intended for internal use and is not further documented.

7.5 Message Handlers

7.5.1 btmdisp

```
/usr/local/libexec/gnubatch/btmdisp options
```

Btmdisp generates messages as required by btsched in response to the mail or write completion options of batch jobs.

By default, it uses the system basic mailer to dispose of mail options, btwrite to send messages to users' terminals and dosbtwrite to send messages to Windows PCs.

The messages are generated by btmdisp from the system message file, which by default is `/usr/local/share/gnubatch/help/btint-config`.

The program to be used in each case may be overridden by assignments to the environment variables `MAILER`, `WRITER` and `DOSWRITER`, most conveniently in the master configuration file `/usr/local/etc/gnubatch.conf`. The program (or shell script) to be run in each case should take data on standard input and the relevant user name as the first argument, and will run under the identity `batch`.

These variables may also be set on a per-user basis by assignment in a `.gnubatch` file located in a user's home directory. The user may also specify an alternative message file by assignment to the variable `SYSMESG`. These programs or scripts will be run under the identity of the user, typically the owner of the job to be run.

The interface (options etc) are internal to **GNUbatch** and are not documented here.

7.5.1.1 Notes

Btmdisp is identical to GNUspool's spmdisp, apart from using different message and configuration files.

7.5.2 btwrite

```
/usr/local/libexec/gnubatch/btwrite user [ ... ]
```

Btwrite sends messages to users' terminals in response to the `-w` option of gbch-r and equivalent. It is used in preference to the write(1) command as this picks just one (and usually the wrong one!) of the terminals at which the user may be logged in, and does not display a suitable name for the originator of messages.

Btwrite takes a list of one or more users as arguments. It sends the text on standard input to each user's terminal. The message is mailed to users who cannot be reached. This facility is available for use in your own software if you wish.

7.5.2.1 Notes

Btwrite is identical to GNUspool's spwrite(8), apart from using a different message file.

7.5.3 dosbtwrite

```
/usr/local/libexec/gnubatch/dosbtwrite options
```

Dosbtwrite sends messages to Windows PCs similar to btwrite does for user's terminals in response to the equivalents of the `-w` options of gbch-r and equivalents. This is only done for jobs which originated on Windows PCs.

The Windows PC must be running btqw for this to be effectual.

If the job was submitted by a user working from a client with a DHCP-allocated IP address, a message may be received on all clients currently logged-in with that user name.

7.5.3.1 Notes

Dosbtwrite is identical to GNUspool's dosspwrite, apart from using a different message file.

7.5.4 Jobdump

```
jobdump options
```

jobdump is invoked by gbch-q, gbch-xmq and gbch-jdel to unqueue jobs when required.

It is not intended for general use and is not documented further.

7.6 File Formats

7.6.1 /usr/local/etc/gnubatch.conf

`/usr/local/etc/gnubatch.conf` is an optional file for reconfiguring the **GNUbatch** batch management system.

This may be useful for relocating standard files and directories, such as `SPOOLDIR` which defaults to `/usr/local/var/gnubatch` so that a different spool directory is used. However completely arbitrary environment variable assignments may be made so as to pass the resulting environment on to various subprocesses invoked by the scheduler.

Note that as the environment is assigned early within gbch-r, then any jobs created will have these environment variables assigned. However there is an alternative syntax (see below) to avoid this.

The format of the file consists of two different forms of assignment.

`SPOOLDIR=/usr2/spooldir`	Sets up the given environment variable in all programs and jobs invoked by **GNUbatch**.

`SPOOLDIR:/usr2/spooldir`	Denotes environment variables which should not be passed on to subprocesses, or to jobs created by gbch-r.

Comment lines may be included, introduced by a `#` sign, and blank lines are ignored.

The latter environment assignment format is used by default in the installation process for GNUbatch.

Every component program of GNUbatch examines this file and resets its environment from this file as the first step of execution.

7.6.2 /usr/local/etc/gnubatch.hosts

`/usr/local/etc/gnubatch.hosts` is used to inform the GNUbatch batch scheduling system, and in particular btsched and xbnetserv, which other host machines are to be attached.

The host machines should in general be provided for in the standard file `/etc/hosts`.

The file consists of comment lines introduced by the `#` character, and of lines consisting of up to 4 fields, of which only the first is mandatory. These fields are as follows:

7.6.2.1 Host name

This is the name of the host as given in the `/etc/hosts` file.

Alternatively an internet address of the form `193.112.238.10` may be given if necessary and an alias is provided on the next field, but this is not recommended.

7.6.2.2 Alias name

This is the name of an alias to be used in preference to the host name to refer to the machine. To be particularly beneficial, this should be shorter than the host name.

If this field is not required, but subsequent fields are required, then the alias name may be replaced by a single `-` sign.

7.6.2.3 Flags

This is a comma-separated list of markers to denote information about the connection. The currently-supported markers are as follows:

`probe`	Indicates that a datagram should be sent, and a reply awaited, from the host, before a full-blown connection is attempted. This is recommended wherever possible, or it is not sure in which order machines are booted.
`manual`	indicates that no connection at all is attempted. To connect to the machine in question, then gbch-conn should be invoked.
`trusted`	indicates that the host is "trusted" by the current machine, which transmits information about Windows clients and their password validations, so the other host need not make such enquiries.

`Client (username)`	indicates that no connection is attempted; the current machine is acting as a server for Windows clients. The specified username is to be considered as the owner of any jobs submitted, and the user to whom charges should be applied and to which privileges apply; see gbch-user. If *`(username)`* is omitted, then the Windows user is assumed, which should correspond to a user name on the host system.
`Clientuser (machine)`	Indicates that the whole entry identifies a "roaming user" who might be using one of several Windows clients, possibly with IP addresses assigned via DHCP. The host name in this case is replaced by the Windows user name, and the alias gives the Unix user name if different. If *`(machine)`* is specified, then a password is demanded at the Windows client if the client's IP address does not match that of machine.
`dos(username)`	Is a synonym for `client(username)` kept for historical reasons.
`dosuser(machine)`	Is a synonym for `clientuser(machine)` kept for historical reasons.
`external`	Is a synonym for `client` (no username) kept for future extensions.
`pwchk`	Always demand the user's Unix password when first starting up.

7.6.2.4 Timeout

This gives a timeout value in seconds after which the interface is to be considered closed following a connection or alternatively to await a connection after a probe request.

A default of 1000 seconds applies if none is specified.

In the case of Windows clients, the "login" is considered to be dropped after this time, and the user may be asked for a password again.

7.6.2.5 Local address

On some machines, the "local" host address may be different from that obtained by looking at the result of gethostname(3). To specify a different address for "this" machine, a line of the form:

```
localaddress 193.112.238.112
```

may be specified, but this must precede all other host names in the file.

8 User Programs

Users have a wide variety of Unix programs which may be used to submit batch jobs, and manage scheduling. This includes a set of standard command line and interactive programs, plus optional Motif GUI applications.

The following are the user programs available, listed by function, including some intended only for set-up and installation. Some of the descriptions which follow are merged together to save repetition.

More detailed descriptions of the interactive interfaces to gbch-q, gbch-user, gbch-xq, gbch-xr, gbch_xuser, gbch-xmq, gbch-xmr and gbch-xmuser follow in the next two chapters, the descriptions here concentrating on the command line options to these programs.

8.1 Syntax of batch commands

All of the options referred to in the descriptions of the shell-level programs for GNUbatch below may be supplied in a *configuration file* (q.v.), or in an environment variable whose name is the same as the calling program, except that it is in upper case with – signs changed to underscore.

This may enable defaults to be supplied according to the application from which the program is invoked. However any options and arguments supplied on the command line usually take priority.

Additionally by editing the appropriate *message file* (q.v.) it is possible to change the option letters and keywords from those described.

8.1.1 Option types

In nearly all cases there are two alternative ways of supplying options:

- Via a traditional Unix-style -letter option, for example as -z. In some cases, such as in gbch-r and gbch-jchange we ran out of letters and had to use other a few other characters, such as digits.
- Via a keyword-style option, such as `+zero-charge`. Keywords are case-insensitive.

8.1.2 Option syntax

The syntax of options is intended to be as flexible as possible. Options which do not take arguments may be grouped together as in

```
-Nwm
```

or they may be given separately as in

```
-N -w -m
```

White space is optional in the case of options which do take arguments, thus both

```
-p150
```

and

```
-p 150
```

are acceptable and have the same effect.

If the keyword version of an option is given, then it must be separated from its argument by white space thus

```
+priority 150
```

8.1.3 Configuration files

To save the user from having to specify commonly-used combinations of options, there are mechanisms enabling these to be supplied to each program automatically.

One mechanism is the use of a *configuration file*, `.gnubatch`, in the current or user's home directory. The other is the use of an environment variable.

These files may also be used to specify alternative *message files*.

The format of configuration files is akin to a set of environment variable assignments, with empty lines and lines beginning with `#` being ignored.

The name assigned to is the same as that of the calling program but in upper case, for example that corresponding to gbch-r is `GBCH_R` etc. This is the same as for the corresponding environment variable.

Usually options are taken from the following places in order, so that later-processed ones override earlier ones:

Standard defaults	Each program has a set of standard defaults which are used to initialise the parameters when the program is invoked.
User profile	In some cases, for example default priority, the user's profile as displayed by gbch-user is used to initialise the defaults.
Home directory	The file `~/.gnubatch` is read, and any options specified therein (i.e. with an assignment to the appropriate name) are interpreted.
Environment	Any options specified in the appropriate environment variable (you will almost certainly have to use quotes when setting it via the shell in order to preserve the white space) are read and interpreted.
Current Directory	The file `.gnubatch` is read, and any options specified therein (i.e. with an assignment to the appropriate name) are interpreted. Note that this may mean that the `.gnubatch` file is read twice if the command is run from the current directory[2]. Beware therefore of options which are cumulative, such as arguments and redirections, and use the "cancel existing" options.
Command line	Any options specified on the command line are interpreted last.

Most options have inverses so that it is possible to reset anything which may have been set by previously-read options. Extra care should be taken with cumulative options, notably arguments and redirections, so that these are not doubled, especially in the case where the home directory is also the current directory.

---

[2] GNUbatch2 overcomes this problem by using a file in directory `~/.gbch` instead for the "home directory".

8.1.4 Option path

The above description of the order of selection of configuration files, environment and command-line options is the default.

It may be desirable to change the order of selection of options, in to eliminate some alternative locations or to include others.

The environment variable GB_CONFIGPATH may be set to a colon-separated list of directories (environment variables and `~user` constructs are appropriately interpreted).

The symbol `!` is used to represent the relevant environment variable, and `-` is used to represent option arguments.

The default value of `GB_CONFIGPATH` is

```
~:!:.:-
```

This provides the interpretation of options in various configuration files and the environment which is documented above.

Note that it is possible to eliminate or override the interpretation of options on the command line by removing or relocating the `-`. This may have very surprising effects especially where configuration files wipe out the effects of options which may have been set on the command line. Where the interpretation of options has been removed altogether, then any options supplied will probably be objected to or misinterpreted as file names or similar.

The commands

```
+freeze-home
```

and

```
+freeze-current
```

and equivalents do not take into account the value of `GB_CONFIGPATH` in any way.

Finally please note that any non-existent or inaccessible directories and files will (usually) be silently ignored. If a configuration file appears to exist but is inaccessible, a diagnostic may be given; however in some cases this may be misleading due to the fact that various versions of Unix are misleading or inconsistent with regard to the error codes reported from an attempt to open a non-existent or inaccessible file in a non-existent or inaccessible directory.

8.1.5 Message files

As well as providing help and error messages, screen key assignments etc, message files also provide the option letters and keyword names used to specify the options.

For each command, there is a default message file. For most of the shell-based commands, this is `/usr/local/share/gnubatch/help/btrest.help`. Alternative message files may be specified using an environment variable or configuration file assigning values to a name. For most of the shell-based commands, this is `BTRESTCONF`.

Within the message file itself, the option letters and keywords are set up using sequences of the form

```
A300:?,explain
```

Comments and the context should make clear which commands these options relate to.

These sequences define

A state number	The state number, in the above example `300`, which is used internally to denote the argument.
option letters	A single character, often a letter, but in the above example `?`, which is the single-character variant of the option, thus `-?`. Several option letters, each separated by commas may be defined. To define "`,`" itself as an option "letter", use `\,`.[3]
option keywords	A string of alphanumerics, possibly including hyphens and underscores, is used to denote an option keyword, in the above example `+explain`. Several such keywords may be defined, each separated by commas. Note that the case of letters in the keywords is discarded.

8.1.6 Location of message files

It is possible to specify alternative locations for message files so that alternatives are selected according to the application being run etc.

The location may be specified using configuration files in a similar fashion to the search for options, except that the search runs the other way.

The search is in the following order:

Current Directory	If a configuration file in the current directory specifies a location for the message file, by means of an assignment to the relevant variable (for most shell-based commands this is `BTRESTCONF`), then this file is taken. Environment variables in the form `$ABC` and users' home directories in the form ~user are appropriately expanded. The sequence `$0` is replaced by the name of the program being invoked. (This process may run recursively up to a level of 10).
Environment	If the relevant environment variable (for most shell-based commands this is `BTRESTCONF`) specifies a location, then this is taken.
Home Directory	A configuration file in the home directory may specify a location for the message file.
Default Location	If none of the above specify a replacement message file then the default location is taken.

If a file is specified but does not exist, then this is a fatal error.

However there is an additional step to assist the user to set up some alternative files with a default name.

---

[3] We intend removing this facility to respecify option letters (as opposed to keywords) in GNUbatch2 as it over-complicates this and gives rise to too many potential conflicts. Please advise us if you think this is a mistake.

Should the file not exist, then the search falls back to a name generated by taking the last part of the default file name (for example `btrest.conf`) and substituting this for the last part of the file name specified.

For example if the normal message file for the command were

```
/usr/local/share/gnubatch/help/btrest.help
```

and the user had specified in a configuration file

```
BTRESTCONF=~/$0.help
```

then if he or she were to run, say, gbch-r, then the file

```
~/gbch-r.help
```

would be searched for. If this did not exist, then a search would be made for

```
~/btrest.help
```

8.1.7 Path to locate message files

The above search path may be modified using the environment variable `GB_HELPPATH`. The interpretation is very similar to the description above for `GB_CONFIGPATH`, except that `-` fields are ignored.

The default value of `GB_HELPPATH` is `.:!:~` giving the interpretation described above. Note that this is in the opposite order to `GB_CONFIGPATH`.

8.2 Submitting Batch Jobs

8.2.1 gbch-r and gbch-rr

```
gbch-r [-options] [ files ]
gbch-rr [-options] [ files ]
```

gbch-r creates a GNUbatch batch job from each of the supplied files or the standard input if no file names are given.

gbch-rr operates similarly, but creates the jobs on a remote host without the necessity of having to have GNUbatch running on the submitting host.

8.2.1.1 Options

Except for the `-Q` option, which must be specified for gbch-rr, and the options keyword used to pick up default arguments and to save with the `+freeze-current` and `+freeze-home` options, the (standard - it would be possible to make them different by editing the option definitions in `/usr/local/share/gnubatch/help/btrest.help` but this would not be sensible) options to gbch-rr are identical in effect to those for gbch-r.

The environment variable on which options are supplied is `GBCH_R` for gbch-r, `GBCH_RR` for gbch-rr and the environment variable to specify the help file is `BTRESTCONF`.

We regret having run out of single letters for options to gbch-r and gbch-rr and having had to resort in three cases to non-alphabetic options. The bts command in GNUbatch2 introduces the new concept of templates carrying most of the information supplied as options to gbch-r.

Option		Arg	Description
`-?`	`+explain`		causes a summary of the other options to be displayed without taking further action.
`-2`	`+grace-time`	*time*	Sets the second stage time of handling over-running jobs to time, in seconds (the argument may be any number of seconds, or given as mm:ss for minutes and seconds). This only applies if a maximum elapsed time for a job is set with the `-Y` option. If a non-zero time is also given with this option, the job is first killed with the signal number given by the `-W` option and then, if it continues to run for the time given by this argument, killed with `SIGKILL` which cannot be caught or ignored.
`-9`	`+catch-up`		sets the "if not possible" action of the job or jobs to catch up - one run of a series of missed runs is done when it is possible without affecting future runs.
`-.`	`+done`		sets the job or jobs to "done" state (this is mainly intended for resubmitting jobs which have been "unqueued" and is not recommended for general use)

`-A`	`+avoiding-days`	*days*	signifies days to avoid when the job or jobs are to be repeated automatically. The days to avoid option supersedes any preceding or default option, unless a leading comma is given. Thus if the existing days to avoid are `Sat` and `Sun`, the default when installed,
			`gbch-r -A Wed`
			will change the days to avoid to be Wednesday only, whereas
			`gbch-r -A ,Wed`
			will change the days to avoid to be Saturday, Sunday and Wednesday.
			A single `-` argument cancels the days to avoid parameter altogether, thus `-A-`.
			Note that this parameter only affects automatic repetitions, so if the date given by the `-T` parameter falls on a day excluded by this argument, it will not be affected and the first run will be on the date specified.
			Upon installation the default abbreviations for the days are `Sun`, `Mon`, `Tue`, `Wed`, `Thu`, `Fri`, `Sat` and `Hday`, the last refers to holidays as specified in the holiday file. The days are interpreted case-insensitively, but on saving options with `+freeze-current` or `+freeze-home` will save the names in the initial capital format.

Option		Arg	Description
`-a`	`+argument`	*string*	Provide an argument string to the command interpreter. Successive `-a` options are cumulative and append additional arguments to the list of arguments for the job or jobs. To clear previously-specified options (maybe set in `.gnubatch` files) and start afresh, use the `-e` option first.
`-B`	`+assignment-not-critical`		Marks subsequently-specified assignments (with the `-s` option) as "not critical", meaning that the assignment will be ignored if it contains a reference to a variable on a remote host which is offline or inaccessible. This must precede the `-s` options to which it is to be applied.
`-b`	`+assignment-critical`		Marks subsequently-specified assignments (with the `-s` option) as "critical", meaning that the job or jobs will not start if the assignment contains a reference to a variable on a remote host which is offline or inaccessible. This must precede the `-s` options to which is to be applied.
`-C`	`+cancelled`		Sets the job or jobs to be in "cancelled" state.
`-c`	`+condition`	*condition*	Sets a condition to be satisfied before the job or jobs may run. Successive `-c` options cause further conditions to be appended to the list, up to a maximum of 10 conditions. To start from scratch, deleting any previously-specified conditions (in a `.gnubatch` file perhaps), use the `-y` option first.
`-D`	`+directory`	*directory*	Sets the working directory for the job or jobs. This may include environment variable references preceded by `$` to be expanded and constructs such as *~user* to select the given user's home directory. (Remember, if using the shell, and using these constructs, to put quotes around the directory, otherwise the shell may expand the constructs and not GNUbatch). If omitted, then the current directory at the time of invoking gbch-r or gbch-rr is used.
`-d`	`+delete-at-end`		cancels any repeat option of the jobs so that they will be deleted at the end of the run rather than repeated or kept. This is the default if no arguments are specified.
`-E`	`+local-environment`		set the environment variables to be as per the local environment rather than the remote environment. This applies to gbch-rr only and is ignored in gbch-r.
`-e`	`+cancel-arguments`		deletes any arguments set up by previous options.
`-F`	`+export`		marks the job or jobs to be visible throughout the network, but only available to run on the machine which they are queued on.

Option		Arg	Description
`-f`	`+flags-for-set`	*letters*	Precede `-s` (set assignment) options with this and an argument consisting of some or all of `SNEACR` (for respectively Start, Normal exit, Error exit, Abort, Cancel and Reverse) to set the flags which determine when an assignment is performed.
`-G`	`+full-export`		marks the job or jobs to be visible throughout the network and potentially available to run on any machine.
`-g`	`+set-group`	*group*	set the group owner of the job or jobs to be group. The user must have write admin file permission to invoke this argument.
`-H`	`+hold-current`		sets the "if not possible" action of the job or jobs to hold current - the run is done when it is possible without affecting subsequent runs.
`-h`	`+title`	*title*	sets the title of the job or jobs to title. In the absence of this argument the title will be that of the last part of the file name, if any. The title may be a string of any length containing any printable characters.
`-I`	`+input-output`	*redir*	sets a redirection for the job or jobs. Successive `-I` options are cumulative and will append to the current list of redirections. To start the list of redirections from scratch, precede them with the `-Z` option.
`-i`	`+interpreter`	*name*	sets the command interpreter for the job or jobs to be name. The load level is also set to that for the interpreter, so if a `-l` argument is to be specified, it should follow the `-i` argument. The command interpreter will be rejected if its load level exceeds the maximum per job for a user.
`-J`	`+no-advance-time-error`		sets the flag so that if the job exits with an error, the next time to run is not advanced according to the repeat specification.
`-j`	`+advance-time-error`		sets the flag so that if the job exits with an error, the next time to run is still advanced if applicable. This is the default if no arguments are specified.
`-K`	`+condition-not-critical`		marks subsequently specified conditions set with the `-c` option as "not critical", i.e. a condition dependent on a variable on an offline or otherwise inaccessible remote host will be ignored in deciding whether a job may start. This is the default if no arguments are specified.
`-k`	`+condition-critical`		marks subsequently specified conditions set with the `-c` option as "critical", i.e. a condition dependent on a variable on an offline or otherwise inaccessible remote host will cause the job to be held up.
`-L`	`+ulimit`	*value*	sets the ulimit value of the job or jobs to the value given. Set zero to indicate unlimited.

Option		Arg	Description
-l	+loadlev	*number*	sets the load level of the job or jobs to be number. The user must have special create permission for this to differ from that of the command interpreter and further the load level must be less than the maximum per job for a user. The load level is also reset by the `-i` (set command interpreter) option, so this must be used before the `-l` option.
-M	+mode	*modes*	sets the permissions of the job or jobs to be modes.
-m	+mail-message		sets the flag whereby completion messages are mailed to the owner of the job. (They may anyway if the jobs output to standard output or standard error and these are not redirected).
-N	+normal		sets the job or jobs to normal "ready to run" state, as opposed to "cancelled" as set by the `-C` option. This is the default if no arguments are specified.
-n	+local-only		marks the job or jobs to be local only to the machines which they are queued on. They will not be visible or runnable on remote hosts.
-O	+remote-environment		initialise the environment variables to be those from the remote environment. This applies to gbch-rr only and is ignored in gbch-r. This is the default if no arguments are specified.
-o	+no-repeat		cancels any repeat option of the job or jobs, so that the they will be run and retained on the queue marked done at the end.
-P	+umask	*value*	sets the umask value of the job or jobs to the octal value given. The value should be up to 3 octal digits as per the shell.
-p	+priority	*number*	sets the priority of the job or jobs to be *number*, which must be in the range given by the user's minimum and maximum priority.
-Q	+host	*hostname*	send the job or jobs to the given *hostname*. Note that hostname must be in `/usr/local/etc/gnubatch.hosts` on the submitting machine and the submitting machine's hostname must be in `/usr/local/etc/gnubatch.hosts` on the receiving machine. If specified with gbch-r, the effect is to invoke gbch-rr with the same command-line options as were given to gbch-r. Note that this does not include any options for gbch-r extracted from the environment or `.gnubatch` files. This option is mandatory for gbch-rr and it will fail if it is not specified. gbch-r is not invoked as otherwise there might be an endless loop of calls (with `-Q` specified in a `.gnubatch` file for gbch-r but not for gbch-rr for example).
-q	+job-queue	*queuename*	sets a job queue name as specified on the job or jobs. This may be any sequence of printable characters.

Option		Arg	Description
`-R`	`+reschedule-all`		sets the "not possible" action of the job or jobs to reschedule all - the run is done when it is possible and subsequent runs are rescheduled by the amount delayed.
`-r`	`+repeat`	*repeat_spec*	sets the repeat option of the jobs as specified.
`-S`	`+skip-if-held`		sets the ``not possible'' action of the job or jobs to skip - the run is skipped if it could not be done at the specified time.
`-s`	`+set`		sets an assignment on the job or jobs to be performed at the start and/or finish of the job or jobs as selected by a previously-specified `-f` option. This option is cumulative, and will add to the list of assignments specified by previous `-s` options. To start from scratch, precede the assignments with the `-z` option.
`-T`	`+time`	*time*	sets the next run time or time and date of the job or jobs as specified.
`-t`	`+delete-time`	*time*	sets a delete time for the specified job or jobs as a time in hours, after which it will be automatically deleted if this time has elapsed since it was queued or last ran. Set to zero to retain the job or jobs indefinitely.
`-U`	`+no-time`		cancels any time setting on the job or jobs set with `-T`, `-r` or `-o` options.
`-u`	`+set-owner`	*user*	set the owner of the job or jobs to be user. The user must have write admin file permission to invoke this argument.
`-V`	`+no-verbose`		cancel the effect of the `-v` option, so that a message is not output giving the job number of each batch job successfully created
`-v`	`+verbose`		output a message on standard error containing the job number of each batch job successfully created.
`-W`	`+which-signal`	*sig*	used in conjunction with `-Y` and `-2` options sets the signal number, e.g. 1, 2, 15 to kill the job or jobs after the maximum run time has been exceeded.
`-w`	`+write-message`		sets the flag whereby completion messages are written to the owner's terminal if available.

Option		Arg	Description

`-X`	`+exit-code`	*`range`*	sets the normal or error exit code ranges for the job or jobs. The format of the range argument is `N` or `E` followed by a range in the form *nn:nn*, thus
			`-X N0:9`
			and
			`-X E10:255`
			Note that an exit code which falls inside both ranges will be handled by the setting of the smaller range, so
			`-X N0:10`
			`-X E1:255`
			will mean that exit codes 1 to 10 inclusive are treated as normal as that is the smaller range. Unhandled exit codes are treated as abort. The default is `N0:0` and `E1:255`.

`-x`	`+no-message`		resets both flags as set by `-m` and `-w`.
`-Y`	`+run-time`	*`time`*	sets a maximum elapsed run time for the specified job or jobs. The argument time is in seconds, which may be written as *mm:ss* or *hh:mm:ss*. The job will be killed with `SIGKILL` unless a different signal is specified with the `-W` option and a further "grace time" specified with the `-2` option.
`-y`	`+cancel-condition`		deletes any conditions set up by previous arguments.
`-z`	`+cancel-set`		deletes any assignments set up by previous arguments.
`-Z`	`+cancel-io`		deletes any redirections set up by previous arguments
`+freeze-current`			Save all the current options in a `.gnubatch` file in the current directory with keyword `GBCH_R` for gbch-r or `GBCH_RR` for gbch-rr.
`+freeze-home`			Save all the current options in a `.gnubatch` file in the user's home directory with keyword `GBCH_R` for gbch-r or `GBCH_RR` for gbch-rr.

8.2.1.2 Redirection format

The format of the argument to the `-I` option is similar to that for the shell with some extensions. The argument should always be enclosed in quotes to avoid the shell interpreting it rather than gbch-r or gbch-rr.

Environment variables and *`~user`* constructs are expanded at run time in the strings.

Parameter substitutions, or "meta data" may be included in the argument strings for redirections, see meta data.

`n<file`	For example
	`3<myfile`
	`7</tmp/data`
	`<input_file`
	Opens the specified file descriptor for input connected to the specified file. The file descriptor may be omitted in the common case of file descriptor 0 (standard input).

`n>file`	For example
	`4>outfile`
	`12>/tmp/out`
	`2>errors.%t`
	`>output_file`
	Opens the specified file descriptor for output, possibly creating the file, or truncating it to zero length first if it exists. The file descriptor may be be omitted in the common case of file descriptor 1 (standard output).

`n>>file`	For example
	`5>>Log`
	`7>>Log.%t`
	`>>output.%t`
	As with the shell, this likewise creates the output file if it does not exists but appends new data to any previous data if it exists, rather than truncating it.

`n<>file`	For example
	`8<>Data`
	`<>Myfile`
	Connect the file descriptor (or file descriptor 0 if not specified) for input and output, read-write mode.

`n<>>file`	For example
	`8<>>Data`
	`<>>Myfile`
	Select read-write mode, appending to existing data.

`n<|program`	For example `7<|uname` Run the specified program and take input from it on the given file descriptor (defaulting to standard input, file descriptor 0, if not specified).

`n|program`	For example
	`2|log_errors`
	`|log_output`
	Run the specified program and send output to it on the given file descriptor (defaulting to standard output, file descriptor 1, if not specified).

`n&n`	Duplicate the second file descriptor as the first file descriptor

`n&-`	Close the given file descriptor.

8.2.1.3 Repeat periods

The repeat period names for the -r option are as follows:

`Minutes`	Period in minutes
`Hours`	Period in hours
`Days`	Period in days
`Weeks`	Period in weeks
`Monthsb`	Months relative to the beginning
`Monthse`	Months relative to the end of the month
`Years`	Period in years

Each is followed by the number of the relevant periods after a colon. In the case of the month parameters, then this should be followed by a "target day" after a colon.

Examples:

```
-r Days:4
-r Monthsb:1:4
-r Monthse:1:31
-r Years:2
```

For `Monthsb` the "target day" is the day of the month to aim for, in this case the 4th of the month. If this would be a "day to avoid", then the following day is tried and so on.

For `Monthse` the "target day" is selected from the day of the month given in the `-T` option. So if the month in the `-T` option has 31 days, then `-r Monthse:1:31`

will select the last day of each month and

```
-r Monthse:1;30
```

will select the second last, but if the month in the `-T` option has 30 days, the first will be invalid and the second will select the last day of the month.

If the selected day cannot be met for any reason, typically because it does not meet the "days to avoid" criteria, then the previous day is tried until an acceptable day is found. In this way you can select the "last working day of the month" or "next to last working day" etc.

8.2.1.4 Conditions

A condition must be of the form

```
[machine:]<varname><condop><constant>.
```

where `varname` is the name of an existing variable for which the user has read permission.

`condop` is one of the following:

`=`	equal to
`!=`	not equal

`<`	less than
`<=`	less than or equal
`>`	greater than
`>=`	greater than or equal

constant is either a string or a numeric value. If the string starts with a number then it should be preceded with a colon.

N.B. When invoked from a shell, quotation marks should surround the entire argument as shown above, otherwise the shell may attach its own interpretation on the various characters.

Examples of conditions:

```
-c 'Count>3'
-c 'Lock=0'
-c 'Remote:Lock!=0'
-c 'Val=:3rd'
```

8.2.1.5 Assignments

Each assignment should normally be preceded by a `-f` option to denote when the assignment is applied, apart from exit code and signal assignments.

The argument to the `-f` option is one or more of the following:

`S`	Perform assignment on startup
`N`	Perform assignment on normal exit
`E`	Perform assignment on error exit
`A`	Perform assignment on abort
`C`	Perform assignment on cancellation
`R`	Reverse assignment for `N`, `E`, `A`, and `C`.

The default if no `-f` options are specified is

```
-f SNEAR
```

but the default for this may be changed by editing the message file.

The format of the argument to the `-s` option is in the format

```
[machine:]<varname><operator><constant>
```

`varname` is the name of a variable to which the user has read and write permission.

`operator` is one of the following:

`=`	Assign value which may be a string or numeric constant. To indicate that a string starting with a digit is intended to be a string, prefix it with a colon. Exceptionally, the variable assigned to may have write permission and not read permission for the user. The effect of the "reverse" flag is to assign zero or the null string. Previous values are not recalled.

`+=`	Increment variable by numeric constant. The effect of the "reverse" flag is to decrement the variable by that constant. Arithmetic is as 32-bit signed integer.
`-=`	Decrement variable by numeric constant. The effect of the "reverse" flag is to increment the variable by that constant. Arithmetic is as 32-bit signed integer.
`*=`	Multiply variable by numeric constant. The effect of the "reverse" flag is to divide the variable by that constant. Arithmetic is as 32-bit signed integer and overflow is ignored.
`/=`	Divide variable by numeric constant. The effect of the "reverse" flag is to multiply the variable by that constant. Arithmetic is as 32-bit signed integer. Note that the remainder from division is ignored. The handling of negative numbers may be dependent on the hardware and should probably not be relied upon.
`%=`	Take the remainder (modulus) from division by the numeric constant. There is no "reverse" of the operation. Arithmetic is as 32-bit signed integer. The handling of negative numbers may be dependent on the hardware and should probably not be relied upon.
`=exitcode`	Assign the exit code of the job to the given variable. Flags are ignored and the operation only occurs when the job exits.
`=signal`	Assign the signal number with which the job terminated to the given variable, or zero if the job did not exit with a signal. Flags are ignored and the operation only occurs when the job exits.

The following are examples of assignments:

```
-s 'myvar=7'
-s 'host2:hisvar+=1'
-s 'status=exitcode'
-s 'val=:3rd'
```

Note the colon in the last assignment indicating that the value is a string, the colon is not included in the string.

8.2.1.6 Mode arguments

The argument to the `-M` option provides for a wide variety of operations.

Each permission is represented by a letter, as follows:

`R`	read permission
`W`	write permission
`S`	reveal permission
`M`	read mode
`P`	set mode
`U`	give away owner
`V`	assume owner
`G`	give away group
`H`	assume group

`D`	delete
`K`	kill

Each section of the mode (user, group, others) is represented by the prefixes `U:`, `G:` and `O:` and separated by commas.

For example:

```
-M U:RWSMPDK,G:RWSDK,O:RS
```

would set the permissions for the user, group and others as given. If the prefixes are omitted, as in

```
-M RWSDK
```

then all of the user, group and other permissions are set to the same value.

8.2.2 atcover

```
gbch-atcover options
```

Gbch-atcover may be used instead of the standard at(1) command. It converts the options that most versions of the at(1) commands take to the equivalent gbch-r commands and then invokes gbch-r to submit the batch jobs.

The gbch-r program provides a much greater set of facilities than at(1). Although it is strongly recommended to switch to the gbch-r command to take advantage of these, they can be made available to users of our gbch-atcover command. This is done by setting up `gbch-r` in the application or user environment.

gbch-atcover is usually installed in place of at in `/usr/bin`, with the original binary moved to something like `old.at` in the same directory.

8.2.3 Gbch-xr and gbch-xmr

```
Gbch-xr &
gbch-xmr &
```

gbch-xr is a fully interactive GTK alternative and gbch-xmr a fully interactive Motif alternative to the standard tools for submitting batch jobs, gbch-r and gbch-rr. Jobs are submitted from saved job files, which may have been created via "unqueue" from gbch-q, gbch-xq, gbch-xmq or gbch-jdel, or created afresh within gbch-xr or gbch-xmr.

Unlike gbch-r etc there are no specific command line options to gbch-xr or gbch-xmr. The facility to change or specify resources settings for an X11 (and hence GTK or Motif) program on the command line can be used.

8.3 Managing the batch scheduler

8.3.1 gbch-start

```
gbch-start [-options]
```

gbch-start initiates the GNUbatch batch scheduler system, by starting the processes btsched and xbnetserv.

8.3.1.1 Options

The environment variable on which options are supplied is `gbch-start` and the environment variable to specify the help file is `BTRESTCONF`.

`-?`	`+explain`		causes a summary of the other options to be displayed without taking further action.
`-l`	`+initial-load-level`	*number*	set the initial value of the `LOADLEVEL` variable, which controls the total load level of running jobs to the specified number (usually zero). If this option is not specified, then the value is unchanged from its initial value saved by the scheduler when it was last shut down. If the GNUbatch scheduler is already running, this option has no effect.
`-j`	`+initial-job-size`	*number*	Allocate shared memory initially for the specified number of jobs. This is necessary in some situations where other applications use up the system limit of shared memory and it is not possible to allocate further after the system has been running for some time. If the GNUbatch scheduler is already running, this option has no effect.
`-v`	`+initial-var-size`	*number*	Allocate shared memory initially for the specified number of variables. This is necessary in some situations where other applications use up the system limit of shared memory and it is not possible to allocate further after the system has been running for some time. If the GNUbatch scheduler is already running, this option has no effect.
`+freeze-current`			Save all the current options in a `.gnubatch` file in the current directory. If this option is specified, there is no further action.
`+freeze-home`			Save all the current options in a `.gnubatch` file in the user's home directory. If this option is specified, there is no further action.

8.3.2 gbch-quit

```
gbch-quit [-y]
```

gbch-quit is a program which should be invoked to bring the GNUbatch batch system to an orderly halt prior to system shutdown. Any jobs and variables will be saved.

Only a user with the stop scheduler privilege may successfully invoke it, and confirmation is requested unless the `-y` option is given.

8.3.2.1 Diagnostics

Various diagnostics may be issued, read as required from the message file `/usr/local/share/gnubatch/help/btrest.help`.

The most important ones are that it is not running and that the user is not permitted to invoke the command.

8.3.3 gbch-conn

```
gbch-conn hostname
```

gbch-conn instructs the GNUbatch scheduler to attempt to raise a connection to the given host, which should be specified in the file `/usr/local/etc/gnubatch.hosts`, and not currently active.

8.3.4 gbch-disconn

```
gbch-disconn hostname
```

gbch-disconn instructs the GNUbatch scheduler to close a connection to the given host, which should be specified in the file `/usr/local/etc/gnubatch.hosts`, and currently active.

8.3.5 gbch-cichange

```
gbch-cichange [-options] name
```

gbch-cichange is a shell-level command to create, delete or change details of a command interpreter according to the options specified. Only one command interpreter may be operated upon at a time.

The command interpreter in question is that given by the final argument name to the command.

The user must have *special create* permission to operate this command - see gbch-user.

8.3.5.1 Options

The environment variable on which options are supplied is `GBCH_CICHANGE` and the environment variable to specify the help file is `BTRESTCONF`.

Option		Arg	Description
`-?`	`+explain`		causes a summary of the other options to be displayed without taking further action.
`-A`	`+add`		The command interpreter whose name and details are given with the other options is to be added.
`-a`	`+args`	*args*	Set the "predefined argument list" to be that given by *args*. This replaces any existing predefined arguments. Supply an empty string with "" (two double-quotes) to delete all arguments. Almost invariably with shells, the `-s` option should be supplied as a predefined argument. This will cause the "real" arguments supplied by the job, e.g. with the `-a` option to gbch-r, which follow the predefined arguments, to be treated as strings and not the names of files.

Option		**Arg**	**Description**
`-D`	`+delete`		The specified command interpreter is to be deleted. Note that the first entry on the list, which is initialised on installation to be the Bourne shell `sh`, cannot be deleted. N.B. This is not subject to extensive checking to ensure that no job currently uses the specified command interpreter, so please check first.
`-e`	`+expand-args`		Expand $-prefixed environment variables, *`~user`* and backquote constructs in job argument strings before invoking the command interpreter, rather than relying upon the command interpreter to do it.
`-i`	`+set-arg0-name`		Argument 0 of the job, often displayed as the process name, is the name of the command interpreter. This is the default.
`-L`	`+load-level`	*number*	Set the load level to number to be the default for new jobs created with this command interpreter. The default for new command interpreters if this option is not given is the special create load level given in the user's profile as displayed by gbch-user. Remember that this load level must be less than or equal to a user's maximum load level per job to make use of this.
`-N`	`+nice`	*number*	Set the nice value to number.
`-n`	`+new-name`	*name*	Supply a new name name for an existing command interpreter. N.B. Beware that existing jobs referring to the old name will not be checked for or changed.
`-p`	`+path`	*pathname*	Set the path pathname to be the process invoked as the command interpreter. Note that gbch-cichange does not attempt to verify the accuracy of this path name. Environment variables etc are not expanded here, the full path name (starting from `/`) should be given.
`-t`	`+set-arg0-title`		Set the flag whereby the process name (or 0th argument) to the command interpreter becomes the job title.
`-U`	`+update`		The specified command interpreter is to have details changed as specified. This is the default in the absence of other options.
`-u`	`+no-expand-args`		Turn off expansion of environment variables,~name constructs and backquote commands.
`+freeze-current`			Save all the current options in a `.gnubatch` file in the current directory with keyword `GBCH_CICHANGE`. The program will exit without error if no command interpreter argument is given.
`+freeze-home`			Save all the current options in a `.gnubatch` file in the user's home directory with keyword `GBCH_CICHANGE`. The program will exit without error if no command interpreter argument is given.

8.3.5.2 Examples

To change the nice value, load level and to specify that the job title will become the process name for jobs running under the sh command interpreter:

```
gbch-cichange -N 22 -L 500 -t sh
```

To add a new command interpreter using the Korn shell with the `-s` option:

```
gbch-cichange -A -N 25 -L 1500 -p /bin/ksh -a '-s' ksh
```

The quotes around `-s` are not necessary in this case, only if spaces are included.

To change the name to `korn`

```
gbch-cichange -n korn ksh
```

8.3.6 gbch-cilist

```
gbch-cilist [-options]
```

gbch-cilist causes a list of command interpreters, optionally for a remote host, to be output on standard output.

8.3.6.1 Options

The environment variable on which options are supplied is `GBCH_CILIST` and the environment variable to specify the help file is `BTRESTCONF`.

`-? +explain`		causes a summary of the other options to be displayed without taking further action.
`-Q +host`	*host*	specifies the host name, defaulting to the host being run, for which the listing is required. To cancel a previously-specified host name, use a single minus sign as an argument, or the local host name.
`+freeze-current`		Save all the current options in a `.gnubatch` file in the current directory using the keyword `GBCH_CILIST`. No output will take place is this is specified.
`+freeze-home`		Save all the current options in a `.gnubatch` file in the user's home directory using the keyword `GBCH_CILIST`. No output will take place is this is specified.

8.3.7 gbch-hols

```
gbch-hols [-C] [-d] [-r] [-s] year [file]
```

gbch-hols is a shell-level program to display or set the holidays file for the given year.

The holidays are displayed or interpreted in the following format (as for UK in 2004)

```
January: 1
April: 9 12
May; 3 31
August: 30
December: 27 28
```

The year is given as 4 digits, thus 2004. Output when displaying, the default, is to standard output.

If setting (with the `-s` option) the input is from standard input or the specified file name. Holidays are added to the existing list for the year unless the `-C` option is also given.

Month names may be given in abbreviated or full format, case-insensitive, but are displayed in full. The full and abbreviated names are extracted from the help file, by default `/usr/local/share/gnubatch/help/btrest.help`.

8.3.7.1 Options

The environment variable on which options are supplied is `GBCH_HOLS` and the environment variable to specify the help file is `BTRESTCONF`.

`-? +explain`	causes a summary of the other options to be displayed without taking further action.
`-C +clear`	Relevant only when the `-s` option is specified, clear the existing holidays for the given year before applying the new ones.
`-d +display`	(Default) display existing holidays.
`-s +set`	Set holidays.
`-r +no-clear`	Reset previously-specified `-C` option.
`+freeze-current`	Save all the current options in a `.gnubatch` file in the current directory using the keyword `GBCH_HOLS`. No output will take place is this is specified.
`+freeze-home`	Save all the current options in a `.gnubatch` file in the user's home directory using the keyword `GBCH_HOLS`. No output will take place is this is specified.

8.3.8 gbch-hostedit

```
gbch-hostedit [-o file] [-s arg] [-I] [ file ]
```

gbch-hostedit is a simple curses-based program to edit host tables for `/usr/local/etc/gnubatch.hosts`, the host table for GNUbatch.

It knows about local addresses (selecting the correct IP address for a machine with different IPs on different networks), Windows clients, DHCP, trusted hosts, manual connections, probes and timeouts.

Input is taken from standard input unless a file name is given, and output is to standard output unless the `-o` option is given.

Normally this would be run as follows:

```
gbch-hostedit -o /tmp/hostfile /usr/local/etc/gnubatch.hosts
cp /tmp/hostfile /usr/local/etc/gnubatch.hosts
rm /tmp/hostfile
```

You will usually have to stop and restart GNUbatch after you have done this so that all parts of the system "know" about the new hosts, however this may not be necessary in all cases, you may only have to "`kill -1`" the process id of the xbnetserv process.

8.3.8.1 Options

`-o file`	Output to the named file rather than Standard Output
`-s char`	Where char is `h` or `i`, sort display by host name or by IP address.
`-I`	Edit in place; a file argument must be given.

The file name may be given as `@` to denote the standard file name of `/usr/local/etc/gnubatch.hosts`.

8.3.8.2 Commands

The following command keys are used from within the screen displayed by gbch-hostedit. As with other GNUbatch commands, any commands which operate upon an existing item will do so with the item to which the cursor is moved.

`k` or cursor up	Move cursor up.
`j` or cursor down	Move cursor down.
`N` or next page	Scroll down a screenful.
`P` or previous page	Scroll up a screenful.
`q`	Quit and write hosts file.
`a`	Create a new hosts entry.
`c`	Edit the selected hosts entry.
`d`	Delete the selected hosts entry.
`l`	Edit the local hosts entry.
`u`	Set the default user name for DHCP clients.

8.4 Querying/managing batch jobs from the command line

8.4.1 gbch-jchange

```
gbch-jchange [-options] job number ...
```

gbch-jchange is a program to modify details of a job or jobs from a shell script or another program. Jobs are specified by using the job number, as displayed by gbch-r with the `-v` (verbose) option, or as in the output of the first column of the gbch-jlist command with default format.

Remote jobs should be specified by prefixing the job numbers with the host name thus:

```
host:1234
```

It is not necessary to specify any leading zeroes.

Several jobs may be specified at once to apply the same set of changes to all of them at once.

8.4.1.1 Options

As supplied, the options to gbch-jchange are more or less identical to those for gbch-r, except that existing jobs have their parameters changed from whatever they are to the specified parameters, and there is no "default", in that mentioning an option means that the user requires an existing parameter for the job or jobs changed. For details of the syntax and much of the meaning of the options, please see the documentation of gbch-r at the corresponding option.

It is a mistake not to specify any options at all.

The environment variable on which options are supplied is `GBCH_JCHANGE` and the environment variable to specify the help file is `BTRESTCONF`.

Option		Arg	Description
`-?`	`+explain`		causes a summary of the other options to be displayed without taking further action.
`-2`	`+grace-time`	*time*	change the secondary maximum run time to time seconds (time may be a number of seconds, or mm:ss for minutes and seconds).
`-9`	`+catch-up`		change the "if not possible" action of the job or jobs to "catch up".
`-A`	`+avoiding-days`	*days*	signifies days to avoid when the job or jobs are repeated. The interpretation of the option syntax, and default days is just the same as for gbch-r, but the existing "days to avoid" in each job is replaced completely with the result. For example, the default days are `Sat` and `Sun`, so `-A ,Wed` Will incorporate the default days to avoid taken from the help file, adding Wednesday and change the days to avoid in the job or jobs to Saturday, Sunday and Wednesday.
`-a`	`+argument`	*string*	Specify an argument for the job or jobs. The argument will be added to the end of the arguments already in the job or jobs or added by previous `-a` arguments to gbch-jchange unless the `-e` option is used, which will discard any pre-existing arguments in the job or jobs as well as any previously-specified `-a` arguments.
`-B`	`+assignment-not-critical`		Marks subsequently specified assignments as "not critical", i.e. an assignment of a variable on an inaccessible remote host will be ignored. Existing assignments are not affected.
`-b`	`+assignment-critical`		Marks subsequent assignments as "critical", i.e. an assignment of a variable on an inaccessible remote host will cause the job to be held up. Existing assignments are not affected.
`-C`	`+cancelled`		Sets the job or jobs to "cancelled" state.
`-c`	`+condition`	*condition*	Sets a condition on the job or jobs. This will add an extra condition to existing conditions in the job or jobs or preceding `-c` unless the `-y` option is given to discard any previously-specified conditions.

Option		Arg	Description
`-D`	`+directory`	*directory*	Sets the working directory for the job or jobs.
`-d`	`+delete-at-end`		Cancels any repeat option of the job or jobs so that they will be deleted at the end of the next run.
`-E`	`+reset-environment`		Resets the environment for the job or jobs to be that of the environment of the gbch-jchange command.
`-e`	`+cancel-arguments`		Deletes any existing arguments for the job or jobs.
`-F`	`+export`		Marks the job or jobs to be visible throughout the network, but only liable to run on the machine which they are on.
`-f`	`+flags-for-set`	*flags*	Sets the flags which determine when an assignment is performed for subsequent `-s` options. Will not affect any existing assignments.
`-G`	`+full-export`		Marks the job or jobs to be visible throughout the network and potentially available to run on any machine.
`-g`	`+set-group`	*group*	Resets the group owner of the job or jobs to group. Note that the setting of the group is done as a separate operation from any other changes. Depending upon whether the pre-existing and new modes and ownership permit the various operations, this may need to be done before, after or interleaved with other changes for it to succeed.
`-H`	`+hold-current`		sets the "if not possible" action of the job or jobs to hold current - the run is done when it is possible without affecting subsequent runs.
`-h`	`+title`	*title*	Sets the title of the job or jobs to title. Note that this may be done whilst the job or jobs are running.
`-I`	`+input-output`	*redirection*	Sets a redirection for the job or jobs. This will add the redirection to any existing redirections for the job or jobs unless they are all reset first using the `-Z` option.
`-i`	`+interpreter`	*name*	Sets the command interpreter for the job or jobs to be name. This will also reset the load level of the job or jobs to be that of the command interpreter. If the load level is to be different from this value, use the `-l` option after this option.
`-J`	`+no-advance-time-error`		Sets the flag so that if the job or jobs exit with an error, the next time to run is not advanced.
`-j`	`+advance-time-error`		Sets the flag so that if the job or jobs exit with an error, the next time to run is still advanced if applicable.
`-K`	`+condition-not-critical`		Marks subsequent conditions as "not critical", i.e. a condition dependent on a variable on an inaccessible remote host will be ignored. Does not affect any conditions already defined.

Option		Arg	Description
`-k`	`+condition-critical`		Marks subsequent conditions as "critical", i.e. a condition dependent on a variable on an inaccessible remote host will cause the job to be held up. Does not affect any conditions already defined.
`-L`	`+ulimit`	*value*	Sets the ulimit value of the job or jobs to the value given.
`-l`	`+loadlev`	*number*	Sets the load level of the job or jobs to be number. The user must have special create permission for this to differ from that of the command interpreter.
`-M`	`+mode`	*modes*	Sets the permissions of the job or jobs to be modes. The format of the modes argument is defined in more detail below. Note that the setting of the mode is done as a separate operation from any other changes. Depending upon whether the pre-existing and new modes and ownership permit the various operations, this may need to be done before, after or interleaved with other changes to succeed.
`-m`	`+mail-message`		Sets the flag whereby completion messages are mailed to the owner of the job or jobs.
`-N`	`+normal`		Sets the job or jobs to normal "ready to run" state.
`-n`	`+local-only`		Marks the job or jobs to be local only to the machines which they are on.
`-o`	`+no-repeat`		Cancels any repeat option of the job or jobs, so that the they will be run and retained on the queue marked "done" at the end.
`-P`	`+umask`	*value*	Sets the umask value of the job or jobs to the octal value given. The value should be up to 3 octal digits as per the shell.
`-p`	`+priority`	*number*	Sets the priority of the job or jobs to be number. Note that the specified priority must be in the range given by the user's minimum and maximum priority.
`-q`	`+job-queue`	*queuename*	Sets a job queue name as specified on the job or jobs.
`-R`	`+reschedule-all`		Sets the "if not possible" action of the job or jobs to reschedule all - the run is done when it is possible and subsequent runs are rescheduled.
`-r`	`+repeat`	*repeat*	Sets the repeat option of the jobs as specified.
`-S`	`+skip-if-held`		Sets the "if not possible" action of the job or jobs to skip - the run is skipped if it could not be done at the specified time.
`-s`	`+set`	*assignment*	Sets an assignment on the job or jobs. The assignment will be added to those already defined unless the existing assignments are cleared first with the `-z` option.
`-T`	`+time`	*time*	Sets the next run time of the job or jobs as specified.

Option		Arg	Description
`-t`	`+delete-time`	*`time`*	Sets a delete time for the specified job or jobs as a time in hours after which it will be automatically deleted.
`-U`	`+no-time`		Cancels any time setting on the job or jobs.
`-u`	`+set-owner`	*`user`*	Resets the owner of the job or jobs to user. Note that the setting of the user is done as a separate operation from any other changes. Depending upon whether the pre-existing and new modes and ownership permit the various operations, this may need to be done before, after or interleaved with other changes to succeed.
`-W`	`+which-signal`	*`sig`*	Sets the signal to kill the job or jobs after the maximum run time has been exceeded.
`-w`	`+write-message`		Sets the flag whereby completion messages are written to the owner's terminal if available.
`-X`	`+exit-code`	*`range`*	Sets the normal or error exit code ranges for the job or jobs. The format of the range argument is N or E followed by a range in the form nn:nn, thus for example `-X N0:9`
`-x`	`+no-message`		Resets both flags as set by `-m` and `-w`.
`-Y`	`+run-time`	*`time`*	Sets a maximum run time for the specified job or jobs. time is in seconds, which may be expressed as *`hh:mm:ss`*.
`-y`	`+cancel-condition`		Deletes any existing conditions in the job or jobs.
`-Z`	`+cancel-io`		Deletes any existing redirections in the job or jobs.
`-z`	`+cancel-set`		Deletes any existing assignments in the job or jobs.
`+freeze-current`			Save all the current options in a `.gnubatch` file in the current directory. If no jobs are specified, this will not be treated as an error and the program will exit after saving the options.
`+freeze-home`			Save all the current options in a `.gnubatch` file in the user's home directory. If no jobs are specified, this will not be treated as an error and the program will exit after saving the options.

8.4.1.2 Mode arguments

The argument to the `-M` option provides for a wide variety of operations.

Each permission is represented by a letter, as follows:

`R`	read permission
`W`	write permission
`S`	reveal permission
`M`	read mode

`P`	set mode
`U`	give away owner
`V`	assume owner
`G`	give away group
`H`	assume group
`D`	delete
`K`	kill

Each section of the mode (user, group, others) is represented by the prefixes `U:`, `G:` and `O:` and separated by commas.

For example:

```
-M U:RWSMPDK,G:RWSDK,O:RS
```

would set the permissions for the user, group and others as given. If the prefixes are omitted, as in

```
-M RWSDK
```

then all of the job, group and other permissions are set to the same value.

An alternative format allows permissions to be added to the existing permissions, thus

```
-M U:+WD,G:+D
```

will add the relevant permissions to whatever is currently set.

Similarly permissions may be cancelled individually by constructs of the form:

```
-M G:-W,O:-RS
```

If the same operation is to be done with two or more of `U`, `G` or `O`, the letters may be run together, for example

```
-M GO:+W
```

8.4.1.3 Note on mode and owner changes

Changing various parameters, the mode (permissions), the owner and the group are done as separate operations.

In some cases changing the mode may prevent the next operation from taking place. In other cases it may need to be done first.

Similar considerations apply to changes of the owner and the group.

gbch-jchange does not attempt to work out the appropriate order to perform the operations, the user should execute separate gbch-jchange commands in sequence to achieve the desired effect.

8.4.2 gbch-jdel

```
gbch-jdel [ -options ] job number ...
```

gbch-jdel provides a means of deleting batch jobs from the shell or a program, optionally killing running jobs if required.

Jobs are specified by using the job number, as displayed by gbch-r with the `-v` (verbose) option, or as in the output of the first column of the gbch-jlist command with default format.

Remote jobs should be specified by prefixing the job numbers with the host name thus:

```
host:1234
```

It is not necessary to specify any leading zeroes.

Appropriate error messages are displayed if the user attempts to delete a job which is either running or if the user does not have the necessary permissions.

8.4.2.1 Options

The environment variable on which options are supplied is `GBCH_JDEL` and the environment variable to specify the help file is `BTRESTCONF`.

Option		Arg	Description
`-?`	`+explain`		causes a summary of the other options to be displayed without taking further action.
`-C`	`+command-prefix`	*name*	specify the given name as the prefix for the command file, followed by the job number, to be used by the `-u` option rather than the default of `C` (which in turn may be changed by editing the message file). The command file is a shell script, typically containing a gbch-r command, which when run via the shell will attempt to create a job equivalent to the one "unqueued". Obviously the user is at liberty to amend this or the job file before doing so.
`-D`	`+directory`	*name*	Save unqueued jobs to name rather than the current directory when gbch-jdel is invoked.
`-d`	`+delete`		Cancel any previous `-k` option to be the default of deleting jobs.
`-e`	`+do-not-unqueue`		Cancel the effect of a previous `-u` option.
`-J`	`+job-prefix`	*name*	specify the given name as the prefix for the job file, followed by the job number, to be used by the `-u` option rather than the default of `J` (which in turn may be changed by editing the message file).
`-K`	`+signal-number`	*signal*	Apply signal number given to kill running job. Default is 15 (`SIGTERM`).
`-k`	`+do-not-delete`		Kill jobs only where applicable, do not delete.
`-N`	`+no-force`		Do not kill or delete running jobs (default).
`-S`	`+sleep-time`	*seconds*	Monitor process state for seconds seconds after killing (default 10 seconds).
`-u`	`+unqueue`		unqueue job(s) to the current directory. Do not delete if `-k` given.
`-Y`	`+force`		Kill and delete running jobs.

Option	Arg	Description
`+freeze-current`		Save all the current options in a `.gnubatch` file in the current directory.
`+freeze-home`		Save all the current options in a `.gnubatch` file in the user's home directory.

8.4.2.2 Examples

To delete jobs even if running:

```
gbch-jdel -y 1237 avon:9371
```

Kill a job without deleting it with signal 2 (`SIGINT`).

```
gbch-jdel -K 2 -k 9120
```

Take a copy of the job in a work directory without deleting it.

```
gbch-jdel -u -k -D ~/work -C spec -J script 9123
```

8.4.3 gbch-jlist

```
gbch-jlist [-options] [job numbers]
```

gbch-jlist is a program to display a summary of the jobs (or to be precise the jobs visible to the user) on the standard output.

Each line of the output corresponds to a single job, and by default the output is generally similar to the default format of the jobs screen of the gbch-q command. The first field on each line (unless varied as below) is the numeric job number of the job, prefixed by a machine name and colon if the job is on a machine other than the one gbch-jlist is run on, job thus:

```
3493
macha:9239
machb:19387
```

This is the required format of the job number which should be passed to gbch-jdel and gbch-jchange.

Various options allow the user to control the output in various ways as described below. The user can limit the output to specific jobs by giving the job numbers as additional arguments.

8.4.3.1 Options

The environment variable on which options are supplied is `GBCH_JLIST` and the environment variable to specify the help file is `BTRESTCONF`.

Option		Args	Description
`-?`	`+explain`		causes a summary of the other options to be displayed without taking further action.

Option		Args	Description
-B	+bypass-modes		Disregard all modes etc and print full details. This is provided for dump/restore scripts. It is only available to users with *Write Admin File* permission, otherwise it is silently ignored. This option is now deprecated as gbch-cjlist is now provided for the purpose for which this option was implemented.
-D	+default-format		Revert the output format to the default format.
-F	+format	*string*	Changes the output format to conform to the pattern given by the format string. This is further described below.
-g	+just-group	*group*	Restrict the output to jobs owned by the group specified, or groups specified with shell-like wild cards as described below. To cancel this argument, give a single - sign as a group name.
-H	+header		Generate a header for each column in the output.
-L	+local-only		Display only jobs local to the current host.
-l	+no-view-jobs		Cancel the -V option and view job parameters rather than job scripts.
-N	+no-header		Cancel the -H option. Do not print a header.
-n	+no-sort		Cancel the -s option. Do not sort the jobs into the order in which they will run.
-q	+job-queue	*name*	Restricts attention to jobs with the queue prefix name. The queue may be specified as a pattern with shell-like wild cards as described below. To cancel this argument, give a single - sign as a queue name. The queue prefix is deleted from the titles of jobs which are displayed.
-R	+include-all-remotes		displays jobs local to the current host and exported jobs on remote machines.
-r	+include-exec-remotes		displays jobs local to the current host and jobs on remote machines which are remote-executable, i.e. which might possibly be executed by the current machine.
-S	+short-times		Displays times and dates in abbreviated form, i.e. times within the next 24 hours as times, otherwise dates. This option is ignored if the -F option is specified.
-s	+sort		causes the output to be sorted so that the jobs whose next execution time is soonest comes at the top of the list.
-T	+full-times		Displays times and dates in full. This option is ignored if the -F option is specified.
-u	+just-user	*user*	Restrict the output to jobs owned by the user specified. The user name may be a pattern with shell-like wild cards as described below. To cancel this argument, give a single - sign as a user name.

Option		Args	Description
`-V`	`+view-jobs`		Do not display job details at all, output the scripts (input to the command interpreter) on standard output.
`-Z`	`+no-null-queues`		In conjunction with the `-q` parameter, do not include jobs with no queue prefix in the list.
`-z`	`+null-queues`		In conjunction with the `-q` parameter, include jobs with no queue prefix in the list.
`+freeze-current`			Save all the current options in a `.gnubatch` file in the current directory with keyword `GBCH_JLIST`. There is no output if this is specified, the options are just saved.
`+freeze-home`			Save all the current options in a `.gnubatch` file in the user's home directory with keyword `GBCH_JLIST`. There is no output if this is specified, the options are just saved.

8.4.3.2 Wild cards

Wild cards in queue, user and group name arguments take a format similar to the shell.

`*`	matches anything
`?`	matches a single character
`[a-mp-ru]`	matches any one character in the range of characters given
`[!n-z9]`	matches any one character not in the range of characters given

Alternatives may be included, separated by commas. For example

```
-q 'a*'
```

displays jobs with queue prefixes starting with `a`

```
-q '[p-t]*,*[!h-m]'
```

displays jobs with queue prefixes starting with `p` to `t` or ending with anything other than `h` to `m`.

```
-u jmc,tony
```

displays jobs owned by `jmc` or `tony`

```
-g 's*'
```

displays jobs owned by groups with names starting `s`.

You should always put quotes around arguments containing the wildcard characters, to avoid misinterpretation by the shell.

8.4.3.3 Format codes

The format string consists of a string containing the following character sequences, which are replaced by the corresponding job parameters. The string may contain various other printing characters or spaces as required.

Each column is padded on the right to the length of the longest entry. If a header is requested, the appropriate abbreviation is obtained from the message file and inserted.

`%%`	Insert a single `%` character.
`%A`	Insert the argument list for job separated by commas.
`%a`	Insert the "days to avoid" separated by commas.
`%b`	Display job start time or time job last started.
`%C`	Display conditions for job in full, showing operations and constants.
`%c`	Display conditions for job with variable names only.
`%D`	Working directory for job.
`%d`	Delete time for job (in hours).
`%E`	Environment variables for job. Note that this may make the output lines extremely long.
`%e`	`Export` or `Rem-runnable` for exported jobs.
`%f`	Last time job finished, or blank if it has not run yet.
`%G`	Group owner of job.
`%g`	Grace time for job (time after maximum run time to allow job to finish before final kill) in minutes and seconds.
`%H`	Title of job including queue name (unless queue name restricted with `-q` option).
`%h`	Title of job excluding queue name.
`%I`	Command interpreter.
`%i`	Process identifier if job running, otherwise blank. This is the process identifier on whichever processor is running the job.
`%k`	Kill signal number at end of maximum run time.
`%L`	Load level
`%l`	Maximum run time for job, blank if not set.
`%M`	Mode as a string of letters with `U:`, `G:` or `O:` prefixes as in `U:RWSMPUVGHDK,G:RSMG,O:SM`.
`%m`	Umask as 3 octal digits.
`%N`	Job number, prefixed by host name if remote.
`%O`	Originating host name, possibly different if submitted via gbch-rr or the API.
`%o`	Original date or time job submitted.
`%P`	Job progress code, `Run`, `Done` etc.
`%p`	Priority
`%q`	Job queue name
`%R`	Redirections
`%r`	Repeat specification
`%S`	Assignments in full with operator and constant
`%s`	Assignments (variable names only)
`%T`	Date and time of next execution
`%t`	Abbreviated date or time if in next 24 hours
`%U`	User name of owner

`%u`	Ulimit (hexadecimal)
`%W`	Start time if running, end time if just finished, otherwise next time to run
`%X`	Exit code ranges
`%x`	Last exit code for job
`%Y`	If "avoiding holidays" is set, display holiday dates for the next year
`%y`	Last signal number for job

Note that the various strings such as `export` etc are read from the message file also, so it is possible to modify them as required by the user.

Only the job number, user, group, originating host and progress fields will be non-blank if the user may not read the relevant job. The mode field will be blank if the user cannot read the modes.

The default format is

```
%N %U %H %I %p %L %t %c %P
```

with the (default) `-S` option and

```
%N %U %H %I %p %L %T %c %P
```

with the `-T` option.

8.4.3.4 Examples

The default output might look like this:

```
15367 jmc  Go-to-optician  memo 150 100  10/08
25874 uucp dba:Admin       sh   150 1000 11:48      Done
25890 uucp dba:Uuclean     sh   150 1000 23:45
25884 uucp dba:Half-hourly sh   150 1000 10:26 Lock
26874 adm
```

If the user does not have read permission on a job, then only limited information is displayed.

This might be limited to a different format with only jobs in queue dba as follows:

```
$ gbch-jlist -q dba -Z -H -F "%N %H %P"
Jobno Title       Progress
25874 Admin       Done
25890 Uuclean
25884 Half-hourly
```

8.4.4 gbch-jstat

```
gbch-jstat [-options] jobnumber
```

gbch-jstat is provided to enable shell scripts to determine the status of a single job.

The jobs is specified by using the job number, as displayed by gbch-r with the `-v` (verbose) option, or as in the output of the first column of the gbch-jlist command with default format.

A remote job should be specified by prefixing the job number with the host name thus:

```
host:1234
```

It is not necessary to include any leading zeroes.

By default, the job is checked to see if it is running, just starting or just finishing, but by means of the `-s` option, the user can specify which states to test for.

gbch-jstat returns an exit code of 0 (true to shells) if the job is in the given state, 1 if it is not, and some other exit code (and a diagnostic) if some other error occurs, e.g. the job does not exist.

8.4.4.1 Options

The environment variable on which options are supplied is `GBHC_JSTAT` and the environment variable to specify the help file is `BTRESTCONF`.

`-? +explain`		Causes a summary of the other options to be displayed without taking further action.
`-d +default-states`		Cancel a `-s` option and revert to checking whether the job is running, just starting or just finishing.
`-s +state`	*statecodes*	Specify statecodes as the states to be tested for. statecodes is a comma-separated list of states exactly as reported by gbch-jlist. (The strings are read from the message file, and can be altered if required). As distributed, they are

empty string		Ready to run, represented if necessary by two quotes, or an empty field preceding a comma, as in
		`gbch-jstat -s '' 1234`
		`gbch-jstat -s ,Done 1234`

`Done`		Normal exit
`Err`		Error exit
`Abrt`		Aborted
`Canc`		Cancelled
`Init`		Startup stage 1 (included in the default case)
`Strt`		Startup stage 2 (included in the default case)
`Run`		Running (included in the default case)
`Fin`		Terminating (included in the default case)

The state names are case insensitive. If one (typically the "ready to run" state) is a null string, then this can be tested for by using a null string or two consecutive commas, thus:

```
Done,,Err
```

8.4.4.2 Example

The following shell script displays a list of the titles of jobs ready to run or running

```
gbch-jlist -F '%N %H'|while read num title
do
    if gbch-jstat -s '' $num
    then
        echo $title is ready to run
    elif gbch-jstat $num
    then
        echo $title is running
    fi
done
```

8.4.5 gbch-jgo, gbch-jgoadv, gbch-jadv

```
gbch-jgo job number ...
gbch-jgoadv job number ...
gbch-jadv job number ...
```

gbch-jgo forces a job or jobs to run, ignoring the "next run time". Conditions and load level constraints are however still enforced. The "next run time" will not be affected when the job completes. This inserts an extra run of the job.

gbch-jgoadv forces a job or jobs to run, ignoring the "next run time". Conditions and load level constraints are however still enforced. The "next run time" is advanced to the next time. This brings forward the next run, thereafter resuming the sequence.

gbch-jadv advances the run time on each job specified to the next run time according to its repeat time without running the job or looking at conditions.

These programs are all hard links to the gbch-jdel binary.

Jobs are specified by using the job number, as displayed by gbch-r with the `-v` (verbose) option, or as in the output of the first column of the gbch-jlist command with default format.

Remote jobs should be specified by prefixing the job numbers with the host name thus:

```
host:1234
```

It is not necessary to specify any leading zeroes.

8.4.6 gbch-dst

```
gbch-dst [ -R ] startdate enddate adjustment
```

gbch-dst adjusts all jobs between the specified start and end dates and times by adding the specified (possibly signed) adjustment in seconds to it.

The dates and times may be specified in the forms

```
dd/mm
mm/dd
yy/mm/dd
```

Which of the first two forms is chosen is taken from the existing time zone. For time zones greater or equal to 4 West from GMT, the *mm/dd* form is chosen, otherwise *dd/mm*.

The dates may be followed by a comma and a time in the form *hh:mm*, otherwise midnight is assumed.

When working out what to do, remember that Unix internal time is based upon Greenwich Mean Time (GMT), it is the display which changes, so that the effect of moving the clocks forward is to make the times (held as GMT) appear later than they did before.

A negative adjustment is subtracted from the time, making jobs run sooner. This is therefore appropriate when the clocks go forward at the start of the summer time. Likewise a positive adjustment should be used at the end of summer time.

The optional argument `-R` tries to apply the option to all exported remote jobs, but this really is not recommended as the local jobs on those hosts will be unaffected probably leaving the users on those machines confused.

8.5 Querying/managing variables from the command line

8.5.1 gbch-vlist

```
gbch-vlist [ -options ] [ variable names ]
```

gbch-vlist is a program to display GNUbatch variables on the standard output. It can be used in both shell scripts and other programs. Each line of the output corresponds to a single variable, and by default the output is generally similar to the default format of the variables screen of the gbch-q command. The first field on each line is the variable name prefixed by a machine name and colon thus:

```
macha:v1
machb:xyz
```

if the variable is on a remote machine. This is the required format of the variable name which should be passed to gbch-var and other shell interface commands.

An example of the output of gbch-vlist is as follows:

```
CLOAD         0                # Current value of load level
Dell:CLOAD    0         Export # Current value of load level
arnie:CLOAD   1000      Export # Current value of load level
LOADLEVEL     20000            # Maximum value of load level
LOGJOBS                        # File to save job record in
LOGVARS                        # File to save variable record in
MACHINE       sisko            # Name of current host
Dell:Neterr   0         Export # Exit code from polling
STARTLIM      5                # Number of jobs to start at once
STARTWAIT     30               # Wait time in seconds for job start
Dell:Two      2         Export #
bar           1                #
foo           123       Export # Testing
```

If the user has reveal but not read permission on a variable, the name only is displayed.

Various options allow the user to control the output in various ways as described below. The user can limit the output to specific variables by giving the variable names as arguments following the options.

8.5.1.1 Options

The environment variable on which options are supplied is `GBCH_VLIST` and the environment variable to specify the help file is `BTRESTCONF`.

Option		Arg	Description
`-?`	`+explain`		causes a summary of the other options to be displayed without taking further action.
`-B`	`+bypass-modes`		Disregard all modes etc and print full details. This is provided for dump/restore scripts. It is only available to users with *Write Admin File* permission, otherwise it is silently ignored. This option is now deprecated as gbch-cvlist is now provided for the purpose for which this option was implemented.
`-D`	`+default-format`		Revert to the default display format, cancelling the `-F` option.
`-F`	`+format`	*string*	Change the display format to string as defined below.
`-g`	`+just-group`	*group*	Restrict the output to variables owned by the group specified. The group name may be a pattern with shell-like wild cards. To cancel this argument, give a single `-` sign as a group name.
`-H`	`+header`		generate a header for each column of output.
`-L`	`+local-only`		list only variables local to the current host.
`-N`	`+no-header`		cancel the `-H` option.
`-R`	`+include-remotes`		list all variables, including those on connected remote hosts.
`-u`	`+just-user`	*user*	Restrict the output to variables owned by the user specified. The user name may be a pattern with shell-like wild cards.
`+freeze-current`			Save all the current options in a `.gnubatch` file in the current directory with keyword `GBCH_VLIST`. There is no output if this is specified, the options are just saved.
`+freeze-home`			Save all the current options in a `.gnubatch` file in the user's home directory with keyword `GBCH_VLIST`. There is no output if this is specified, the options are just saved.

8.5.1.2 Format codes

The format string consists of a string containing the following character sequences, which are replaced by the following variable parameters. The string may contain various other printing characters or spaces as required.

Each column is padded on the right to the length of the longest entry.

If a header is requested, the appropriate abbreviation is obtained from the message file and inserted.

`%%`	Insert a single `%`.
`%C`	Comment field.

`%E`	Export if variable is exported
`%G`	Group owner of variable.
`%K`	Cluster if the variable is marked clustered
`%M`	Mode as a string of letters with `U:`, `G:` or `O:` prefixes as in `U:RWSMPUVGHD,G:RSMG,O:SM`.
`%N`	Name
`%U`	User name of owner.
`%V`	Value

Note that the various strings such as `export` etc are read from the message file also, so it is possible to modify them as required by the user.

Only the `name`, `user`, `group`, `export` and `cluster` fields will be non-blank if the user may not read the relevant variable. The mode field will be blank if the user cannot read the modes.

The default format is

```
%N %V %E # %C
```

8.5.2 gbch-var

```
gbch-var [-options] variable name
```

gbch-var is a shell level tool to display, create, delete, modify or test the values of GNUbatch variables. Testing may be "atomic", in the sense that if two or more users attempt to assign new values to the same variable conditional on a test, only one will ``win''.

8.5.2.1 Options

The environment variable on which options are supplied is `GBCH_VAR` and the environment variable to specify the help file is `BTRESTCONF`.

Option		Arg	Description
`-?`	`+explain`		causes a summary of the other options to be displayed without taking further action.
`-C`	`+create`		Create the variable if it doesn't exist. An initial value should be supplied using the `-s` option.
`-c`	`+comment`	*string*	Assign or update the given comment field of the variable to be string.
`-D`	`+delete`		Delete the variable.
`-E`	`+set-export`		Mark the variable as "exported", i.e. visible to other hosts.
`-G`	`+set-group`	*group*	Change the group ownership of the variable to *group*.
`-K`	`+cluster`		Set the "clustered" marker on the variable. When used in conditions or assignments, the local version is used.
`-k`	`+no-cluster`		Reset the "clustered" marker on the variable.

Option		Arg	Description
`-L`	`+set-local`		Mark the variable as local to the host only. This is the default for new variables, for existing variables it will turn off the export flag if it is specified. To leave existing variables unaffected, invoke the `-N` flag.
`-M`	`+set-mode`	*`mode`*	Set the mode (permissions) on the variable.
`-N`	`+reset-export`		Reset the `-L` and `-E` options. For new variables this will restore to the default of local only. For existing variables this will mean that the export flag is left unchanged.
`-o`	`+reset-cluster`		Reset the `-k` and `-K` options. For new variables this will restore to the default of not clustered. For existing variables this will mean that the cluster flag is left unchanged.
`-S`	`+force-string`		Force all set values to string even if they appear numeric.
`-s`	`+set-value`	*`value`*	Assign the given value to the variable.
`-U`	`+set-owner`	*`user`*	Change the ownership of the variable to *user*.
`-u`	`+undefined-value`	*`value`*	In the test operations, if the variable does not exist, treat it as if it did exist and had the given value.
`-X`	`+cancel`		cancel options `-S`, `-C`, `-D`, `-s` and `-u`.
`+freeze-current`			Save the current options in a `.gnubatch` file in the current directory with keyword `GBCH_VAR`. Comparison options are not included.
`+freeze-home`			Save all the current options in a `.gnubatch` file in the user's home directory with keyword `GBCH_VAR`. Comparison options are not included.

8.5.2.2 Conditions

The six conditions `+eq`, `+ne`, `+gt`, `+ge`, `+lt` `+le` followed by a constant compare the variable value with the constant specified. The constant is assumed to be on the right of the comparison, for example:

```
gbch-var +gt 4 myvar
```

Returns an exit code of zero ("true" to the shell) if `myvar` is greater than 4, or 1 ("false" to the shell) if it is less than or equal to 4. (Some other exit code would be returned if `myvar` did not exist).

This may be combined with other options, for example

```
gbch-var -D +gt 100 myvar
```

Would delete `myvar` only if its value was greater than 100.

```
gbch-var -s 1 +le 0 myvar
```

Would assign 1 to `myvar` only if its previous value was less than or equal to 0. Exit code 0 (shell "true") would be returned if the test succeeded and the other operation was completed successfully, exit code 1 (shell "false") would be returned if the test failed and nothing was done, or some other error if the variable did not exist or the operation was not permitted.

The test is "atomic" in the sense that a diagnostic will occur, and no assignment made, if some other process sets the value in between the test and the assignment (or other change).

The condition must follow all other options.

`+eq`, `+ne`, `+lt` and `+gt` may be represented as `-e`, `-n`, `-l` and `-g` but this is not particularly recommended, especially for the last two.

8.5.2.3 Use of options

With no options, then the current value of the variable is printed, for example:

```
gbch-var abc
```

prints out the value of variable `abc`.

To assign a value, the `-s` option should be used, thus

```
gbch-var -s 29 abc
```

assigns the numeric value 29 to `abc`.

Remote variables are referred to as follows:

```
gbch-var -s 32 host2:def
```

assigns 32 to variable `def` on `host2`.

The conditional options should be the last to be specified.

The `-u` option may be used to specify a value to substitute for a non-existent variable in a test rather than reporting an error, for example:

```
gbch-var -u 10 -gt 5 myvar
```

will compare `myvar` with 5 if it exists. If it does not exist, then it will compare the given value, in this case 10, with 5, and in this case return "true". There should not be a diagnostic unless there is a completely different error.

8.5.2.4 Note on mode and owner changes

Changing various parameters, the mode (permissions), the owner and the group are done as separate operations.

In some cases changing the mode may prevent the next operation from taking place. In other cases it may need to be done first.

Similar considerations apply to changes of the owner and the group.

gbch-var does not attempt to work out the appropriate order to perform the operations, the user should execute separate gbch-var commands in sequence to achieve the desired effect.

8.6 Interactive job and variable administration

8.6.1 gbch-q

```
gbch-q [ -options ]
```

gbch-q is an interactive program that allows the user to display in real-time state and details of GNUbatch jobs and variables on local or remote machines, refreshing the screen automatically as the queue changes or variables are updated, and allowing the status of jobs and variables on the queue to be altered according to each user's permissions and privileges.

Please see Chapter 9.1 for more details of the interactive commands. This section focuses on the command-line options which may be used to control the initial display.

8.6.1.1 Options

The environment variable on which options are supplied is `GBCH_Q` and the environment variable to specify the help file is `BTQCONF`.

Certain commands available on-screen enable many of these options to be changed and saved in configuration files.

Option		Arg	Description
`-?`	`+explain`		Causes a summary of the other options to be displayed without taking further action or entering the interactive display.
`-A`	`+no-confirm-delete`		Suppress confirmation request for delete operations.
`-a`	`+confirm-delete`		Ask confirmation of delete operations (this is the default).
`-B`	`+no-help-box`		Put help messages in inverse video rather than in a box (this is the default).
`-b`	`+help-box`		Put help messages in a box rather than displaying inverse video.
`-E`	`+no-error-box`		Put error messages in inverse video rather than in a box.
`-e`	`+error-box`		Put error messages in a box rather than displaying inverse video.
`-g`	`+just-group`	*group*	Restrict the output to jobs owned by the group specified. Cancel this argument by giving a single `-` sign as an argument. The group name may be a pattern with shell-like wild cards.
`-H`	`+keep-char-help`		When displaying a help screen, interpret the next key press as a command as well as clearing the help screen. This is the default.
`-h`	`+lose-char-help`		Discard whatever key press is made to clear a help screen.
`-j`	`+jobs-screen`		Commence display in jobs screen. This is the default unless there are no jobs to be viewed.
`-l`	`+local-only`		Only display jobs local to the machine.
`-N`	`+follow-job`		If a job or variable moves on the screen, try to follow it and do not try to retain relative screen positions.

Option		Arg	Description
`-q`	`+job-queue`	*name*	Restricts attention to jobs with the queue prefix name. Cancel this argument by giving a single `-` sign as an argument. The queue name may be a pattern with shell-like wild cards.
`-r`	`+network-wide`		Display jobs on all connected hosts.
`-s`	`+keep-cursor`		If a job or variable moves on the screen, try to preserve the relative position of the cursor on the screen rather than following the job or variable.
`-u`	`+just-user`	*user*	Restrict the output to jobs owned by the user specified. Cancel this argument by giving a `-` parameter. The user name may be a pattern with shell-like wild cards.
`-v`	`+vars-screen`		Commence display in variables screen.
`-Z`	`+no-null-queues`		In conjunction with the `-q` parameter, do not include jobs with no queue prefix in the list.
`-z`	`+null-queues`		In conjunction with the `-q` parameter, include jobs with no queue prefix in the list.

8.6.2 Gbch-xq and gbch-xmq

```
Gbch-xxq &
Gbch-xmq &
```

gbch-xq and gbch-xmq are fully interactive GTK and Motif alternatives to the standard queue manager gbch-q. As with gbch-q the format of the screen display, the help messages and the command keystrokes can be easily altered to suit your requirements.

Unlike gbch-q there are no specific command line options to gbch-xq and gbch-xmq. The facility to change or specify resources settings for an X11 (and hence Motif) program on the command line can be used.

8.7 File Monitoring

8.7.1 gbch-filemon

```
gbch-filemon [-options]
```

gbch-filemon executes a given program or script when specified files change in specified ways in a specified directory.

It is intentionally not integrated with the GNUbatch core product, as there is no automatic mechanism within Unix for signalling changes to files, and it is therefore necessary to "poll" or monitor the files at a given interval. gbch-filemon is made as small as possible so that the "polling" does not have a large impact on the system.

The rest of GNUbatch is made to be "event-driven", as this has minimal impact on the system when the product is inactive.

The "action" of gbch-filemon may be to run a GNUbatch job, set a variable, or perform some completely unrelated task.

gbch-filemon may optionally be used to list or terminate running copies of itself.

8.7.1.1 Options

The GTK program gbch-xfilemon and the X/Motif program gbch-xmfilemon may be used to set up the options to and run gbch-filemon rather than remembering them here.

The environment variable on which options are supplied is `GBCH_FILEMON` and the environment variable to specify the help file is `FILEMONCONF`.

Option		Arg	Description
`-?`	`+explain`		causes a summary of the other options to be displayed without taking further action.
`-A`	`+file-arrives`		Perform the required action when a new file is detected in the directory.
`-a`	`+any-file`		Perform the required action for any file name.
`-C`	`+continue-running`		Continue gbch-filemon after a matching file and condition has been found, looking for further files.
`-c`	`+script-command`		Specify command to execute when one of the monitored events occurs. This is an alternative to `-X`, which runs a named shell script. In the command the sequence `%f` is replaced by the name of the file whose activity has provoked the action, and `%d` by the directory. To use this option, be sure to enclose the whole shell command in quotes so that it is passed as one argument, thus: `-c "xmessage -bg red 'Found %f'"`
`-D`	`+directory`	*dir*	Specify the given *dir* as the directory to monitor rather than the current directory.
`-d`	`+daemon-process`		Detach a further gbch-filemon as a daemon process, and return to the user.
`-e`	`+include-existing`		Include existing files in the scan, and report changes etc to those. If the `-A` option (watch for file arriving), this will have no effect unless an existing file is deleted and is recreated.
`-G`	`+file-stops-growing`	*secs*	Activate command when a file has appeared, and has not grown further for at least *secs*. Distinguish this from the `-M` option, which will check for any change, possibly in the middle of a file.
`-I`	`+file-stops-changing`	*secs*	Activate command when a file has appeared, and has not been changed for at least secs. This is more inclusive than `-M`, as it includes activities such as changing the ownership or mode of the file, or making hard links.
`-i`	`+ignore-existing`		Ignore existing files (default). However if an existing file is noted to have been deleted, and then re-created, the new version will be treated as a new file.

Option		Arg	Description
`-K`	`+kill-all`		Kill all gbch-filemon daemon processes belonging to the user, or all processes if invoked by `root`.
`-k`	`+kill-processes`	*dir*	Kill any gbch-filemon daemon processes left running which are scanning the given directory. Processes must belong to the invoking user, or gbch-filemon be invoked by `root`.
`-L`	`+follow-links`		Follow symbolic links to files (and subdirectories with the `-R` option).
`-l`	`+list-processes`		List running gbch-filemon processes and which directories they are accessing.
`-M`	`+file-stops-writing`	*secs*	Activate command when a file has appeared, and has not been written to for at least secs. This is more inclusive than `-G` as it includes writes other than to the end of the file. It is less inclusive than `-I` which also monitors for linking and permission-changing.
`-m`	`+run-monitor`		Run as a file monitor program (default) rather than `-l`, `-k` or `-K.`
`-n`	`+not-daemon`		Do not detach gbch-filemon as a daemon process (default), wait and only return to the user when a file event has been detected.
`-P`	`+poll-time`	*secs*	Poll directory every *secs* seconds. This should be sufficiently small not to "miss" events for a long time, but large enough to not load the system. The default if this is not specified is 20 seconds.
`-p`	`+pattern-file`	*pattern*	Perform action on a file name matching pattern. Pattern may take the form of wild-card matching given by the shell, with `* ? [a-z] [!a-z]` having the same meanings as with the shell, and possible alternative patterns separated by commas, for example: `-p '*.[chyl],*.obj'` Remember to enclose the argument in quotes so that it is interpreted by gbch-filemon and not the shell.
`-R`	`+recursive`		Recursively follow subdirectories of the starting directory.
`-r`	`+file-deleted`		Perform action when a file matching the criteria has been deleted.
`-S`	`+halt-when-found`		Halt gbch-filemon once a matching file and condition has been found.
`-s`	`+specific-file`	*file*	Perform action only with a specific named file, not a pattern.
`-u`	`+file-stops-use`	*secs*	Perform action when a file has appeared, and has not been read for at least *secs*.

Option		Arg	Description
`-X`	`+script-file`	*`file`*	Specify the given script as a shell script to execute when one of the monitored events occurs. This is an alternative to `-c` The existence of the shell script is checked, and gbch-filemon will fail with an error message if it does not exist. The shell script is passed the following arguments: 1. File name 2. Directory path 3. File size (or last file size if file deleted). 4. Date of file modification, change or access as YYYY/MM/DD, but only for those type of changes. 5. Time of file modification, change or access as HH:MM:SS, but only for those type of changes.
`+freeze-current`			Save all the current options in a `.gnubatch` file in the current directory with keyword `GBCH_FILEMON`. If this is specified, gbch-filemon will terminate after saving the file.
`+freeze-home`			Save all the current options in a `.gnubatch` file in the user's home directory. If this is specified, gbch-filemon will terminate after saving the file.

8.7.1.2 File matching

What to look for may be made to depend upon something happening to

Any file	With the `-a` option. Any file that meets the other criteria will trigger the event.
Specific file	With the `-s` option, gbch-filemon will watch for the specific file named.
Pattern	With the `-p` option, a file which matches the pattern and the other criteria will trigger the action.

8.7.1.3 Criteria

There are 6 criteria to watch for.

File arriving	This is probably the most common case. If you want to wait for a file being written and trigger an event, the `-A` option will look for this.
File removal	This will watch for files being deleted, for example some applications use a "lock file" to denote that they are being run, and you might wish to start something else when it has gone. Remember that you might want to include existing files in the scan with `-e` if the file in question existed when you started gbch-filemon.

File stopped growing	What this watches for is for a file being having been created, or with the `-e` option starting to "grow", and then apparently no longer grown for the given time. If files are arriving from FTP, for example, then when they are complete, they will cease to "grow" in size.
File no longer written	A file not used sequentially may be written to internally rather than have additional data appended. This often occurs with database files, where records are updated somewhere in the middle of the file. If a series of database transactions is made and then completed, the file will no longer be written to for some time, and gbch-filemon can be made to trigger an action after that time. You will often want to include the `-e` option if the file existed already on entry.
File no longer changed	This goes a stage further than "no longer written" as it includes any kind of change to the file, such as permissions, owner, hard links or change of access and write times.
File no longer used	This monitors the access time of the file, updated whenever the file is read, and proceeds when this has gone unchanged for the specified time. You will often want to include the `-e` option with this if the file existed already on entry.

8.7.1.4 Pre-existing files

If the `-i` (ignore existing) option is specified, which is the default, then no changes to existing files which would otherwise match the criteria will be considered, except where an existing file is deleted and then recreated and gbch-filemon "notices" this happen, in that the file is deleted before one "poll" of the directory and recreated before another. In other words, if the poll time is 20 seconds, then the deletion and recreation will have to be 20 seconds apart.

If the -e option to include existing files is specified, the `-G` `-u` `-M` `-I` and `-r` options will work as for new files but not `-A` as the file has already "arrived". However, if it is deleted, this is "noticed" and then recreated, it will be treated as a "new" file.

8.7.1.5 Recursive searches

If recursive searches are specified using the `-R` option, a separate gbch-filemon process will be invoked for each subdirectory, for each further subdirectory within each of those subdirectories, and for each new subdirectory created within one of those whilst each process is running, unless the `-r` option is being used to watch for file removal, whereupon only those subdirectories which existed to begin with will be considered.

If the `-S` option is specified to stop once a file has been found, each process will continue until a file is found in its particular subdirectory.

8.7.1.6 Examples

Monitor the FTP directories for new files which have finished arriving, sending a message to the user:

```
gbch-filemon -aRC -D /var/spool/ftp -G 30 -c "xmessage '%f in %d'"
```

Set a GNUbatch variable to an appropriate value when a file arrives in the current directory

```
gbch-filemon -aAC -c "gbch-var -s '%f arrived' file_var"
```

8.7.2 gbch-xfilemon and gbch-xmfilemon

```
gbch-xfilemon &
gbch-xmfilemon &
```

Gbch-xfilemon and gbch-xmfilemon are simple dialog interfaces to set up parameters for gbch-filemon.

8.8 User administration and charging

8.8.1 gbch-charge

```
gbch-charge [-options] [user] ...
```

gbch-charge displays and/or updates the charges recorded by GNUbatch against various users specified on the command line, or all users if none are specified.

It is only available to users with the *Write admin file* privilege.

Charges relate to jobs which have originated on the machine on which gbch-charge was invoked. The actual charging rate is determined by the machine on which the job is run.

8.8.1.1 Options

The environment variable on which options are supplied is `GBCH_CHARGE` and the environment variable to specify the help file is `BTRESTCONF`.

Option		Arg	Description
`-?`	`+explain`		causes a summary of the other options to be displayed without taking further action.
`-C`	`+consolidate`		Causes a consolidation record or "current balance" to be written in the file for each user, which summarises the charges to date for each user. The previous records remain in the file.
`-c`	`+add-charge`	*fee*	Causes the specified fee to be added to the charge for the specified user or users, or all users if none are given. If combined with a print option (`-p` or `-P`), the print is done after the charge is applied, so the charge will be included in the output.
`-K`	`+cancel-flags`		Cancels all previously-set options `-p` `-P` `-z` `-C` and `-R`. Useful if you want to start again from some indeterminate state in a configuration file.
`-P`	`+print-full`		causes the full log of charges for the specified user or users, or all users if none are given, to be displayed on standard output.

Option		Arg	Description
`-p`	`+print-users`		(the default if no other options are specified) causes the charge summary for the specified user or users, or all users if none are given, to be displayed on standard output.
`-R`	`+reset-file`		Causes all records to be deleted from the file except for consolidation records for each user with a non-zero charge. If the charge for each user is zero (as a result of applying the `-z` option perhaps), then the file will be left empty. Note that the batch scheduler btsched should not be running when this option is invoked, otherwise this option is ignored.
`-z`	`+zero-charge`		Clears to zero the charges for the specified user or users, or all users if none are given. This is done by appending an appropriate record or records to the charge file, the previous information (as displayed by `-P` will still be available. If combined with a print option (`-p` or `-P`), the print is done before the option is applied.
`+freeze-current`			Save all the current options in a `.gnubatch` file in the current directory with keyword `GBCH_CHARGE`. If this option is supplied, and no users are specified, then no further action is taken.
`+freeze-home`			Save all the current options in a `.gnubatch` file in the user's home directory with keyword `GBCH_CHARGE`. If this option is supplied, and no users are specified, then no further action is taken.

8.8.1.2 Examples

Print charges for all users

```
gbch-charge
```

Add a fee of 9.1 units to user sec and print the result

```
gbch-charge -p -c 9.1
```

Print charges for all users and zero them

```
gbch-charge -z -p
```

Note that it is intended to deprecate charging in future releases of GNUbatch.

8.8.2 gbch-uchange

```
gbch-uchange [-options] [users]
```

gbch-uchange is a shell tool that may be used to update the user permissions file giving the user profiles of various users and the operations which they may be permitted to perform within the GNUbatch system. Alternatively the "default permissions" may be updated. These are the permissions which are assigned by default to new GNUbatch users.

Further options allow for a "password dump" file to be maintained. This is for the benefit of NIS-type environments where reading through most of the password database can take an unacceptably long time, the user name and userid hash table is maintained in a file and updated as necessary.

The invoking user must have *write admin file* permission.

8.8.2.1 Options

The environment variable on which options are supplied is `GBCH_UCHANGE` and the environment variable to specify the help file is `BTRESTCONF`.

Option		Arg	Description
`-?`	`+explain`		Causes a summary of the other options to be displayed without taking further action.
`-A`	`+copy-defaults`		Copy the default profile to all users before setting other permissions on the named users (with the `-u` option) or after setting the defaults (with the `-D` option). The privileges of the invoking user are not changed by this operation.
`-D`	`+set-defaults`		Indicate that the other options are to apply to the default profile for new users.
`-d`	`+default-priority`	*num*	Set the default job priority to *num*, which must be between 1 and 255.
`-J`	`+job-mode`	*modes*	Set the default permissions on jobs according to the format of the modes argument.
`-l`	`+min-priority`	*num*	Set the minimum job priority to *num*, which must be between 1 and 255.
`-M`	`+max-load-level`	*num*	Set the maximum load level for any one job to *num*, which must be between 1 and 32767.
`-m`	`+max-priority`	*num*	Set the maximum job priority to *num*, which must be between 1 and 255.
`-N`	`+no-rebuild`		Cancel the `-R` option.
`-p`	`+privileges`	*privileges*	Set the privileges of the user(s) as specified by the argument.
`-R`	`+rebuild-file`		Rebuild the user permissions file `/usr/local/var/gnubatch/btufile` incorporating any changes in the password list.
`-S`	`+special-load-level`	*num*	Set the special load level for the user(s) to num, which must be between 1 and 32767.
`-s`	`+no-copy-defaults`		Cancel the effect of the `-A` option
`-T`	`+total-load-level`	*num*	Set the total load level for the user(s) to num, which must be between 1 and 32767.
`-u`	`+set-users`		Indicate that the other options are to apply to the users specified on the rest of the command line, resetting any previous `-D` option.
`-V`	`+var-mode`		Set the default permissions on variables according to the format of the modes argument.

Option		Arg	Description
`-X`	`+dump-passwd`		Dump out the hash table of the password file to avoid re-reading the password file within the other programs.
`-Y`	`+default-passwd`		Default handling of password hash file dump - rebuild if it is already present and `-R` specified, otherwise not.
`-Z`	`+kill-dump-passwd`		Delete any existing dumped password hash file.
`+freeze-current`			Save all the current options in a `.gnubatch` file in the current directory.
`+freeze-home`			Save all the current options in a `.gnubatch` file in the user's home directory.

8.8.2.2 Users or default

In one operation gbch-uchange either adjusts the default permissions, to be applied to new users, if `-D` is specified, or specified users, if nothing or `-u` is specified. So first set the required defaults:

```
gbch-uchange -D -n 20 -p CR,SPC,ST,Cdft -A
```

Then set named users

```
gbch-uchange -p ALL jmc root batch
```

8.8.2.3 Rebuilding the user control file

After adding new users to the system, you should rebuild the user control file by running

```
gbch-uchange -R
```

On a system with a large number of users, this can take a long time during all of which the user file is locked, so the previous method of adding new users as they were encountered meant that various hold-ups occurred in standard utilities or the scheduler, whichever was the first to "notice" the changes, which might, in the event, be half-complete.

We suggest that this command be added to the "add new user" procedure for your installation.

8.8.2.4 Dumping the password file

This has now been deprecated and the options are ignored.

8.8.2.5 Privileges

The following may be specified as the argument to `-p`, as one or more (comma-separated) of argument may be one or more of the following codes, optionally preceded by a minus to turn off the corresponding privilege.

RA	read admin file

`WA`	write admin file
`CR`	create
`SPC`	special create
`ST`	stop scheduler
`Cdft`	change default
`UG`	or user and group modes
`UO`	or user and other modes
`GO`	or group and other modes.

`ALL` may be used to denote all of the permissions, and then perhaps to cancel some. For example:

```
-p CR,ST,Cdft
-p ALL,-WA
```

A hexadecimal value is also accepted, but this is intended only for the benefit of the installation routines.

8.8.2.6 Mode arguments

The argument to the `-J` and `-V` options provides for a wide variety of operations.

Each permission is represented by a letter, as follows:

`R`	read permission
`W`	write permission
`S`	reveal permission
`M`	read mode
`P`	set mode
`U`	give away owner
`V`	assume owner
`G`	give away group
`H`	assume group
`D`	delete
`K`	kill (only valid for jobs)

Each section of the mode (job, group, others) is represented by the prefixes `U:`, `G:` and `O:` and separated by commas.

For example:

```
-J U:RWSMPDK,G:RWSDK,O:RS
```

would set the permissions for the user, group and others as given. If the prefixes are omitted, as in

```
-J RWSDK
```

then all of the user, group and other permissions are set to the same value. Alternatively two of the `J`, `G` or `O` may be run together as in

```
-J U:RWSKD,GO:RWS
```

if "group" or "other" (in this case) are to have the same permissions.

8.8.3 gbch-ulist

```
gbch-ulist [-options] [user ...]
```

gbch-ulist lists the permissions of users known to the GNUbatch batch scheduler system. All users are listed if no users are specified, otherwise the named users are listed. The report is similar to the main display of gbch-user.

The invoking user must have read admin file permission to use gbch-ulist.

8.8.3.1 Options

The environment variable on which options are supplied is `GBCH_ULIST` and the environment variable to specify the help file is `BTRESTCONF`.

Option		Arg	Description
`-?`	`+explain`		Causes a summary of the other options to be displayed without taking further action.
`-D`	`+default-format`		Cancel the `-F` option and revert to the default format.
`-d`	`+default-line`		Display an initial line giving the default options (included by default).
`-F`	`+format`	*format*	Format the output according to the format string given.
`-g`	`+group-name-sort`		Sort the list of users by the group name in ascending order, then by users within that group as primary group.
`-H`	`+header`		Generate a header for each column of the output.
`-N`	`+no-header`		Cancel the `-H` option.
`-n`	`+numeric-user-sort`		Sort the list of users by the numeric user id (default).
`-S`	`+no-user-lines`		Suppress the user lines. It is an error to invoke this and the `-s` option as well.
`-s`	`+no-default-line`		Suppress the initial line giving the default options.
`-U`	`+user-lines`		Display the user lines (default).
`-u`	`+user-name-sort`		Sort the list of users by the user name.
`+freeze-current`			Save all the current options in a `.gnubatch` file in the current directory.
`+freeze-home`			Save all the current options in a `.gnubatch` file in the user's home directory.

8.8.3.2 Format argument.

The format string consists of a string containing the following character sequences, which are replaced by various user permission parameters. The string may contain various other printing characters or spaces as required.

Each column is padded on the right to the length of the longest entry.

If a header is requested, the appropriate abbreviation is obtained from the message file and inserted.

`%%`	Insert a single `%` character.
`%d`	Default priority
`%g`	Group name
`%j`	Job mode
`%l`	Minimum priority
`%m`	Maximum priority
`%p`	Privileges
`%s`	Special create load level
`%t`	Total load level
`%u`	User name.
`%v`	Variable mode
`%x`	Maximum load level

The string `DEFAULT` replaces the user name in the default values line, or the group name if the user name is not printed. If the group name is not printed as well, then this will be omitted and will be indistinguishable from the rest of the output.

Note that the various strings are read from the message file, so it is possible to modify them as required by the user.

The default format is

```
%u %g %d %l %m %x %t %s %p
```

8.8.3.3 Privileges format

The following are output via the %p format. Note that the actual strings are read from the message file, and are the same ones as are used by gbch-uchange(1).

`RA`	read admin file
`WA`	write admin file
`CR`	create
`SPC`	special create
`ST`	stop scheduler
`Cdft`	change default
`UG`	or user and group modes
`UO`	or user and other modes
`GO`	or group and other modes.

`ALL` is printed if all privileges are set.

8.8.3.4 Modes

Modes printed by the %j and %v options are as follows:

`R`	read permission
`W`	write permission

`S`	reveal permission
`M`	read mode
`P`	set mode
`U`	give away owner
`V`	assume owner
`G`	give away group
`H`	assume group
`D`	delete
`K`	kill (only valid for jobs)

Each section of the mode (job, group, others) is represented by the prefixes `U:`, `G:` and `O:` and separated by commas.

For example:

```
U:RWSMPDK,G:RWSDK,O:RS
```

This is exactly the same format as is expected by gbch-uchange etc.

8.8.4 gbch-user

```
gbch-user [ -options ]
```

gbch-user provides 4 functions:

With no arguments it lists the permissions for the invoking user and exits.

With the `-m` option it enables the invoking user to edit his own default job and variable permissions. The user must have change default modes permission to do this.

With the `-v` option it enables the invoking user to interactively review the list of permissions for all users. The user must have read admin file permission to do this.

With the `-i` option it enables the invoking user to interactively review and update the list of permissions for all users. The user must have write admin file permission to do this.

Please see Chapter 9.2 for more details of the interactive commands. This section focuses on the command-line options which may be used to control the initial display.

8.8.4.1 Options

The environment variable on which options are supplied is `GBCH_USER` and the environment variable to specify the help file is `BTUSERCONF`.

Certain commands available on-screen enable many of these options to be changed and saved in configuration files.

Option		Description
`-?`	`+explain`	Causes a summary of the other options to be displayed without taking further action.
`-B`	`+no-help-box`	Put help messages in inverse video rather than in a box when operating in interactive mode. This is the default case.

Option		Description
`-b`	`+help-box`	Put help messages in a box rather than displaying inverse video when operating in interactive mode.
`-d`	`+display-only`	Is the default. A list of permissions is output to the standard output.
`-E`	`+no-error-box`	Put error messages in inverse video rather than in a box when operating in interactive mode. This is the default case.
`-e`	`+error-box`	Put error messages in a box rather than displaying inverse video when operating in interactive mode.
`-g`	`+group-name-sort`	Sort users by group name, then user name within primary group. This is only relevant to `-v` or `-i` options.
`-H`	`+keep-char-help`	When displaying a help screen in interactive mode, interpret the next key press as a command as well as clearing the help screen.
`-h`	`+lose-char-help`	Discard whatever key press is made to clear a help screen when in interactive mode.
`-i`	`+update-users`	View and update the list of users. This option requires write admin file privilege.
`-m`	`+set-default-modes`	Interactively set the default modes for the invoking user. This option requires change default modes privilege.
`-n`	`+numeric-user-sort`	Sort users by numeric userid (default).
`-u`	`+user-name-sort`	Sort users by user name.
`-v`	`+view-users`	View in read-only mode the list of users and permissions. This requires read admin file privilege.

8.8.5 gbch-xmuser

```
gbch-xuser &
gbch-xmuser &
```

gbch-xuser and gbch-xmuser are fully interactive Motif alternatives to the standard user control program gbch-user, but only in the fully interactive mode, so they cannot be started by a user without *Write admin file* privilege.

Unlike gbch-user there are no specific command line options to gbch_xuser and gbch-xmuser. The facility to change or specify resources settings for an X11 (and hence Motif) program on the command line can be used.

8.9 Web browser interface support

btjccgi	Operations on jobs CGI program
btjcgi	List jobs CGI program
btjcrcgi	Create jobs CGI program
btjdcgi	Delete jobs CGI program
btjvcgi	View jobs CGI program
btvccgi	Operations on variables CGI program
btvcgi	List variables CGI program

rbtjccgi	Operations on jobs CGI program
rbtjcgi	List jobs CGI program
rbtjcrcgi	Create jobs CGI program
rbtjdcgi	Delete jobs CGI program
rbtjvcgi	View jobs CGI program
rbtvccgi	Operations on variables CGI program
rbtvcgi	List variables CGI program

8.10 File management

8.10.1 gbch-uconv

```
gbch-uconv [-D dir] [-v n] [-e n] [-s] [-f] user file outfile
```

gbch-uconv converts the GNUbatch user file, which is usually `btufile` or `btufile` held in the batch spool directory to an executable shell script file outfile, which if executed, would recreate the GNUbatch users permissions with the same options and privileges.

gbch-uconv understands the format of the saved user file for versions of Xi-Batch going back to release 4, and when presented with a saved file, will attempt to work out from the format which release it relates to.

In addition to options, two arguments are always supplied to gbch-cvlist.

User file	This is the file containing the permissions of the users, `btufile` in the batch spool directory, by default `/usr/local/var/gnubatch`, or as relocated by re-specifying `SPOOLDIR`.
Output file	This file is created by gbch-uconv to contain the executable shell script, containing gbch-uchange commands, which may be used to recreate the user file. This file should be run before restarting the scheduler.

8.10.1.1 Options

Option	Description
`-D` *`directory`*	specifies the source directory for the users and user file. It can be specified as `$SPOOLDIR` or `${SPOOLDIR-/usr/local/var/gnubatch}` etc and the environment and/or `/usr/local/etc/gnubatch.conf` will be interrogated to interpolate the value of the environment variable given. If you use this, don't forget to put single quotes around it thus: `gbch-uconv -D '${SPOOLDIR-/usr/local/var/gnubatch}' ...` otherwise the shell will try to interpret the `$` construct and not gbch-uconv.
`-e` *`n`*	Tolerate *n* errors of the kinds denoted by the other options before giving up trying to convert the file.
`-f`	Ignore format errors in the saved user file, up to the limit of errors given by the `-e` option.

Option	Description
`-s`	Ignore file size errors in the saved user file (up to the number of total errors given by the `-e` option.
`-v` *`number`*	Tell gbch-uconv that the user file is for release number of Xi-Batch, where number is 4 to 6. This may be necessary where the user file is corrupted and gbch-uconv cannot guess what is meant.

8.10.2 gbch-cjlist

```
gbch-cjlist [-D dir] [-v n] [-e n] [-u] [-s] [-f] jobfile outfile
workdir
```

gbch-cjlist converts GNUbatch job files held in the batch spool directory to an executable shell script which may be used to recreate them. This may be useful for backup purposes or for one stage in upgrading from one release of GNUbatch to another.

The jobs are copied into the backup directory `workdir`, and the generated shell script file `outfile` refers to files in that directory.

gbch-cjlist understands the format of the saved job file for versions of Xi-Batch going back to release 5, and when presented with a saved file, will attempt to work out from the format which release it relates to. If required, it will skip apparent errors in the job file.

In addition to options, three arguments are always supplied to gbch-cjlist.

Job list file	This is the file containing the attributes of the jobs, `btsched_jfile` in the spool directory, by default `/usr/local/var/gnubatch`, or as relocated by re-specifying `SPOOLDIR`.
Output file	This file is created by gbch-cjlist to contain the executable shell script which may be used to create the jobs.
Backup directory	This directory is used to hold the job data.

Note that saved jobs make use of the `-U` option to gbch-r to set the ownership correctly.

8.10.2.1 Options

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td><code>-D</code> <i><code>directory</code></i></td><td>Specifies the source directory for the jobs and job file rather than the current directory. It can be specified as <code>$SPOOLDIR</code> or <code>${SPOOLDIR-/usr/local/var/gnubatch}</code> etc and the environment and/or <code>/usr/local/etc/gnubatch.conf</code> will be interrogated to interpolate the value of the environment variable given. If you use this, don't forget to put single quotes around it thus:
<pre>gbch-cjlist -D '${SPOOLDIR-/usr/local/var/gnubatch}' ...</pre>otherwise the shell will try to interpret the <code>$</code> construct and not gbch-cjlist.</td></tr>
</table>

Option	Description
`-e` *`n`*	Tolerate *n* errors of the kinds denoted by the other options before giving up trying to convert the file.
`-f`	Ignore format errors (invalid strings etc) in the saved job file up to the number of total errors given by the `-e` option. Offending jobs are skipped.
`-s`	Ignore file size errors in the saved job file (up to the number of total errors given by the `-e` option.
`-u`	Do not check user names in the saved job file.
`-v` *`number`*	Tell gbch-cjlist that the job file is for release number of Xi-Batch, where number is 5 to 6. This may be necessary where the job file is corrupted and gbch-cjlist cannot guess what is meant.

8.10.3 gbch-cvlist

```
gbch-cvlist [ -D dir ] [ -v n ] [ -e n ] [ -s ] [ -f ] var file
outfile
```

gbch-cvlist converts GNUbatch variables held in the batch spool directory to an executable shell script which may be used to re-install them. This may be useful for backup purposes or for one stage in upgrade from one release of GNUbatch to another.

gbch-cvlist understands the format of the saved variable file for versions of Xi-Batch going back to release 4, and when presented with a saved file, will attempt to work out from the format which release it relates to.

In addition to options, two arguments are always supplied to gbch-cvlist.

Variable list file	This is the file containing the attributes of the variables, `btsched_vfile` in the batch spool directory, by default `/usr/local/var/gnubatch`, or as relocated by re-specifying `SPOOLDIR`.
Output file	This file is created by gbch-cvlist to contain the executable shell script, containing gbch-var commands, which may be used to recreate the variables.

8.10.3.1 Options

<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td><code>-D</code> <i><code>directory</code></i></td><td>Specifies the source directory for the variables and var file if not the current directory. It can be specified as <code>$SPOOLDIR</code> or <code>${SPOOLDIR-/usr/local/var/gnubatch}</code> etc and the environment and/or <code>/usr/local/etc/gnubatch.conf</code> will be interrogated to interpolate the value of the environment variable given. If you use this, don't forget to put single quotes around it thus:
<pre>gbch-cvlist -D '${SPOOLDIR-/usr/local/var/gnubatch}' ...</pre>otherwise the shell will try to interpret the <code>$</code> construct and not gbch-cvlist.</td></tr>
<tr><td><code>-e</code> <i><code>n</code></i></td><td>Tolerate <i>n</i> errors of the kinds denoted by the other options before giving up trying to convert the file.</td></tr>
</table>

Option	Description
`-f`	Ignore format errors in the saved variable file, skip over invalid entries up to the limit given by the `-e` option.
`-s`	Ignore file size errors in the saved variable file (up to the number of total errors given by the `-e` option).
`-v` *`number`*	Tell gbch-cvlist that the variable file is for release number of Xi-Batch, where number is 4 to 6. This may be necessary where the variable file is corrupted and gbch-cvlist cannot guess what is meant.

8.10.4 gbch-ciconv

```
gbch-ciconv [-D dir] [-v n] [-e n] [-u] [-s] [-f] cifile outfile
```

gbch-ciconv converts GNUbatch command interpreters held in the batch spool directory to an executable shell script which may be used to re-install them. This may be useful for backup purposes or for one stage in upgrade from one release of GNUbatch to another.

gbch-ciconv understands the format of the saved job file for versions of Xi-Batch going back to release 4, and when presented with a saved file, will attempt to work out from the format which release it relates to.

In addition to options, two arguments are always supplied to gbch-cvlist.

Command interpreter list file	This is the file containing the attributes of the variables, `cifile` in the batch spool directory, by default `/usr/local/var/gnubatch`, or as relocated by re-specifying `SPOOLDIR`.
Output file	This file is created by gbch-cvlist to contain the executable shell script, containing gbch-cichange commands, which may be used to recreate the command interpreters.

8.10.4.1 Options

Option	Description
`-D` *`directory`*	specifies the source directory for the command interpreter file. It can be specified as `$SPOOLDIR` or `${SPOOLDIR-/usr/local/var/gnubatch}` etc and the environment and/or `/usr/local/etc/gnubatch.conf` will be interrogated to interpolate the value of the environment variable given. If you use this, don't forget to put single quotes around it thus: `gbch-ciconv -D '${SPOOLDIR-/usr/local/var/gnubatch}' ...` otherwise the shell will try to interpret the `$` construct and not `gbch-ciconv`.
`-e` *`n`*	Tolerate *n* errors of the kinds denoted by the other options before giving up trying to convert the file.
`-f`	Ignore format errors in the saved command interpreter file up to the limit of errors given in the `-e` option.
`-s`	Ignore file size errors in the saved command interpreter file (up to the number of total errors given by the `-e` option).

Option	Description
`-u`	Do not check user names (not used, but included for consistency with gbch-cjlist and gbch-cvlist).
`-v` *number*	Tell gbch-ciconv that the job file is for release number of Xi-Batch, where number is 4 to 6. This may be necessary where the command interpreter file is corrupted and gbch-ciconv cannot guess what is meant.

8.10.5 gbch-ripc

```
gbch-ripc [-d] [-r] [-F] [-A] [-D secs] [-P psarg] [-G] [-n] [-o file]
     [-S dir] [-x]  [-B n] [-N char]
```

gbch-ripc traces, and/or optionally monitors or deletes IPC facilities for GNUbatch. Many of the facilities are used for debugging, but it also serves as a quick method of deleting the IPC facilities, being easier to use than ipcs and ipcrm, in the event that the scheduler has crashed or been killed without deleting the IPC facilities.

To use this facility, just run gbch-ripc thus:

```
gbch-ripc -d >/dev/null
```

The diagnostic output may be useful as it reports any inconsistencies.

The monitoring option can be used to diagnose processes, possibly not GNUbatch ones, which are interfering with GNUbatch shared memory segments, in cases where a third-party application is suspected of damaging the shared memory.

gbch-ripc also checks for errors in memory-mapped files where the version of GNUbatch is using those rather than shared memory.

8.10.5.1 Options

Option	Description
`-A`	Display details of jobs and variables. This often generates a lot of output and is not really necessary.
`-D` *secs*	Monitor which process has last attached to the job shared memory segment and report apparent corruption, polling every secs seconds.
`-d`	Delete the IPC facilities after printing out contents. This saves messing with arguments to ipcrm(1).
`-f`	Display the free chains for jobs and variables. This generates a lot of output and isn't usually necessary.
`-n`	Suppress display from `-D` option if everything is OK.
`-o` *outfile*	Output to *outfile* rather than standard output. Set it to /dev/null if you don't want to see any output.
`-P` *psarg*	Specify argument to ps(1) to invoke if corruption detected when monitoring with `-D` option. The output is passed through fgrep(1) to find the line (if any) with the process id of the process which last attached to the shared memory.
`-G`	Used in conjunction with the `-P` option, the output from ps(1) is displayed in full, without passing it through fgrep(1).

Option	Description
`-r`	Read and display the entries on the message queue. This is normally suppressed because they can't be "peeked at" or "unread".
`-S` *`dir`*	This is only relevant for versions of GNUbatch which use memory-mapped files rather than shared memory. It specifies the location of the spool directory. If this is not specified, then the master configuration file `/usr/local/etc/gnubatch.conf` is consulted to find the spool directory location, or failing that, the directory `/usr/local/var/gnubatch` is used.
`-x`	Dump the contents of shared memory or memory-mapped files in hexadecimal and ASCII characters.
`-B` *`n`*	Where n may be 1 to 8, specify the width of the hexadecimal dump output as a number of 32-bit words.
`-N` *`<char>`*	Replace the character in the ASCII part of the hexadecimal dump to represent non-ASCII characters with the specified character (the first character of the argument). The default is `.`. To specify a space, you may need to use quotes thus: `-N ' '`

8.10.5.2 Example

To delete all IPC facilities after **GNUbatch** has crashed.

```
gbch-ripc -d -o /dev/null
```

To monitor the job shared memory segment for errors, printing out the ps(1) output (where the full listing is obtained with `-ef`) search for the process id of the last process to attach to the segment. Print out the contents of the segment including in hexadecimal after corruption is detected.

```
gbch-ripc -D 30 -P -ef -o joblog -A -x
```

8.10.6 gbpasswd

```
gbpasswd [-u user] [-p password] [-f] [-d] [-F file]
```

gbpasswd sets a password for the current user or a specified user if `-u` is given. This is separate and distinct from the user's login password. This password is used by the web interfaces, the Windows interfaces and the APIs.

The reason for doing this is because it is considered insecure to possibly repeatedly try login passwords from user programs.

If any users have a password set in this way, then all users will have to to have a password in the file to use any of the interfaces requiring a password.

Unlike the Unix passwd(1) routine, the old password is not prompted for and there is no confirmation.

8.10.6.1 Options

Option	Description
`-u` *`user`*	Set password for given user. This may only be for other than the current user if gbpasswd is invoked by `root`.
`-p` *`password`*	Specify the password to be set other than prompting for it.
`-f`	It is normally considered an error to include a password for `root` for the same reasons that the password file is separate. However this option may be set to insist upon it.
`-d`	Delete the user's password from the file.
`-F` *`file`*	Use file to hold the password. The default if no file is given is `/usr/local/share/gbpasswd`. Any number of `-F` options may be given to set up several password files at once.

9 Text screen-based programs

The following sections provide more information on the text screen-based programs gbch-q and gbch-user, with attention to the screen commands. See the previous chapter regarding command-line options to gbch-q and gbch-user.

9.1 gbch-q - interactive batch queue manager

The following screens and screen-based commands apply to gbch-q. Remember, however that all of the screen formats and the key bindings may be adjusted to suit your requirements, so the default commands listed here may have been changed on your system.

9.1.1 User Interface Settings on Entry

If gbch-q is given no arguments, then the jobs list is displayed, unless there are no jobs visible to the user, in which case the variables list is displayed. The jobs or variables display can be pre-selected by the `-j` and `-v` options respectively.

The `-A` option turns off the prompt for confirmation of deletions and `-a` turns it on.

The `-s` and `-N` options change the handling of the job queue display when the current job moves within the queue. The `-s` option tries to keep the job under the cursor at the same screen position whatever happens. The `-N` option tries to ensure that the job list appearance changes as little as possible.

The `-h` option causes the next keystroke after displaying a help message to just clear the message. The `-H` option causes the next keystroke to be taken as a command as well as clearing the message.

When no queue name is specified, all queues (including the un-named one) will be displayed by default. In this case, the corresponding queue name of each job can be seen as a prefix to its job *title* separated by a colon "`:`". The `-q` option allows the user to specify which job queues to display. Only jobs belonging to the specified queues will be displayed. No prefixes will be shown on the job title. The `-z` and `-Z` options allow the user to control whether or not jobs without a queue prefix are also displayed.

If the title is changed within gbch-q then the job queue name will be prefixed to it.

9.1.2 General gbch-q screen commands

A different set of interactive commands, and related on-line help messages, is available for each context or screen within gbch-q. For consistency there is a common set of commands to perform standard operations, such as forward and reverse scroll, cursor up and down, help and quit. These are then added to by the relevant commands for each context.

Not all of the common set of commands are available in some contexts. For example search commands are only available in the job and variable lists.

It is possible to re-assign any or all of the keystrokes corresponding to the commands, using the mechanisms described in the Configurability Chapter 11. The remainder of this section will describe the functionality of gbch-q using the default configuration.

The set of common keys for commands are:

Keys	Operation performed
`?`	Help (this screen)
ctrl-L	Refresh screen
`q Q`	Quit
`k`	Move cursor up one line or entry
`j`	Move cursor down one line or entry
`b`	Keyed once moves cursor to top of the screen. Keyed a second time, or if the cursor is already at the top of screen, moves to the beginning of the list.
`e`	Keyed once moves cursor to bottom of the screen. Keyed a second time, or if the cursor is already at the bottom of screen, moves to the end of the list.
Ctrl-B	Moves up list by one screen
Ctrl-F	Moves down list by one screen
Ctrl-U	Moves up list by half a screen
Ctrl-D	Moves down list by half a screen
`^`	Search forward for string
`\`	Search backward for string

The use of the arrow keys for cursor up and down, or Home or function keys are most helpful. The GNUbatch installation will attempt to set cursor movement keys up for terminals that support them. If cursor keys are available but not configured during installation they can be set up as described in the Configurability Chapter 11.

The search commands discard the case of letters, and take "`.`" as a wild-card character.

9.1.2.1 Context Sensitive Help

Pressing the "`?`" key will bring up help relevant to the current context. This ranges from a list of the commands available for a whole screen to simple prompts for entering data in a parameter field.

The help messages may be in inverse video or inside a box, according to user preference.

9.1.2.2 Macro Commands

Additional operations, known as macro commands, can be defined for use within gbch-q. These are not described in this manual because they are custom built as required. How to create and set up Macro commands is described in the Extensibility chapter 12 and especially the section 12.4 on macro definitions starting on page 232.

Up to nine macros can be set up for jobs, and another nine for variables, to be invoked by a single key press. Other macros can be used by pressing the general macro key and typing in the name of the command to be run.

The choice of keys to assign to each macro, including the general macro key, is left to the person who installs the macro(s). How to invoke each macro should be documented, in the on-line help, by the person who installed it.

9.1.3 The Job Screen and Commands

Assuming that there are plenty of jobs in the queue which are visible to the user; a typical screen for the, default configuration, job list might look like this:

```
Seq Jobno   User    Title         Shell   Pri Load Time  Cond      Prog
  0 340     wally   e-mail:dial u sh      150 1000 16:33
  1 734     tony    prog_a        sh      150 1000 06/02           Run
  2 1420    wally   Output Exampl sh      150 1000 29/01           Err
  3 735     tony    prog_b        sh      150 1000 08/02 A_STATUS
  4 736     tony    prog_c        sh      150 1000 08/02 A_STATUS
  5 439     wally   wally         sh      150 1000                 Canc
  6 588     wally   Also Sprach Z sh      150 1000 04/02           Done
  7 564     wally   Daily Update  sh      150 1000                 Run
  8 455     pior    Simple Job    sh      150 1000 11/03           Abrt
  9 309     wally   par:start     sh      150 1000 08/03           Canc
 10 310     wally   par:Process d sh      150 1000 08/03 **Cond**
 11 312     wally   par:Process d sh      150 1000 08/03 **Cond**
 12 313     wally   par:Process d sh      150 1000 08/03 **Cond**
 13 314     wally   par:Collect d sh      150 1000 08/03 **Cond**
 14 315     wally   par:Error han sh      150 1000 08/03 **Cond**
 15 316     wally   par:cleanup   sh      150 1000 08/03 **Cond**
                             -- 9 more jobs below --
=======================================================================
GNUbatch gbch-q (c) Free Software Foundation 2009   (? for help)
```

The job most likely to run next is the job at the top of the queue. As a job is completed, it will either be deleted altogether, or else it will be repositioned at the end of the queue.

Only jobs for which the user has at least "reveal" permission are displayed. If the user has reveal but not read permission only the first two columns of information are shown. All of the information is visible if the user has read permission for the job.

The time field is only displayed if a time constraint exists. It is displayed in 24-hour clock form if the job is scheduled to start within 24 hours. Otherwise the start day is displayed in *`dd/mm`* or *`mm/dd`* form, depending on the time zone being used. (*`mm/dd`* is displayed if the time zone is more than 4 hours West, otherwise *`dd/mm`* is used).

If the variable name or names will fit, then the names of the variables are displayed, otherwise `**Cond**` is shown.

Finally, the job's *Progress* is displayed. This may be one of

`Run`	when the job is running.
`Strt` or `Init`	if the job is being started.
`Fin`	if the job has just completed and GNUbatch is still performing assignments and updating related information.
`Done`	When the job has completed successfully and is being held on the queue, without a specified repetition.
`Err`	if the job terminated with an exit status indicating an error.

`Abrt`	if the job was terminated with a signal (either killed, or due to a program fault outside of GNUbatch control).
`Canc`	if the job was terminated before it started.

If the job is yet to run, or has been run and reset to run again, the progress field is blank. All commands available from within the job list are as follows. Where a specific job is involved, the cursor should be moved to the job to select that job first.

Key	**Function**
`D`	Delete the job. Confirmation may be requested (`y` or `n`).
`M`	Open window to View / Edit the job modes (access permissions)
`O`	Change the owner of the job
`G`	Change the group of the job
`"`	Change the title of the job
`p`	Change the priority of the job
`l`	Change the load level of the job (only with *special create* privilege).
`I`	View the script of the job on the screen
`t`	Edit the start time, retention and repeat specification for the job
`a`	Advance scheduled time to next repeat time without running job
`C`	Open window to Add, Delete and Edit job conditions
`S`	Open window to Add, Delete and Edit job assignments
`x`	Change the command interpreter for the job
`P`	Change the progress setting for the job
`r`	Set job runnable
`z`	Set job cancelled or held (does not invoke the cancel assignments)
`f`	Force job to run but do not advance time
`g`	Force job to run if possible, advancing to the next repeat if applicable
`K`	Kill or cancel the job
`F`	Open window to set the mail and write job completion flags.
`u`	Edit process parameters, i.e. directory, umask, ulimit and exit code conditions.
`A`	Open window to view / edit argument list for the job.
`E`	Open window to view / edit environment variables for the job.
`R`	Open window to add, delete and modify redirections for the job.
`U`	Unqueue the job
`X`	Open window to add, delete and edit command interpreters for the scheduler
`^`	Search forwards for job title
`\`	Search backwards for job title
`V`	Switch to variable list screen
`H`	Open window to set up holidays for the scheduler
`$`	Open window to change program options
`,`	Modify format and content of job list

If the user is not permitted to perform the requested operation an Error message appears and nothing further happens.

9.1.3.1 View job script

The `I` command causes the current job to be displayed on the screen. The commands to the job are assumed to be text. Any non-printing characters will appear in inverse-video representations, i.e. binary 1 will appear as an inverse-video `A`.

To page through the display, use the following commands:

Command	**Function**
`q`	Quit back to jobs screen.
SPACE j *down cursor*	Page down
`k` *up cursor*	Page up
`B b`	Top of document
`E e`	Bottom of document
`h` or *left cursor*	Shift window left
`l` or *right cursor*	Shift window right
`<`	Left margin
`>`	Right margin

9.1.3.2 Change title

To change the title for a job, type `"` and key in any string of characters directly into the field, terminated by pressing the RETURN or ENTER key.

It is possible to display a help message in free text fields like this one, but the help command will have to be bound to a key like a function key. The key chosen must be other than "`?`" or a printing character, because these are permitted as part of the title.

9.1.3.3 Time Specification

To edit the time and repetition controls press t, which opens this sub-window:

```
Seq Jobno   User    Title         Shell   Pri Load Time  Cond      Prog
+---------------------------------------------------------------------+
|    Set time for: e-mail:dial up Yes                                 |
|                                                                     |
|    23:11 Mon 22 Jan 2001                                            |
|                                                                     |
|    Repeat: Once (& delete)                                          |
|            Once (& retain)                                          |
|            Minutes                                                  |
|            Repeat every   2 Hours 01:11 Tue 23 Jan 2001             |
|            Days                                                     |
|            Weeks                                                    |
|            Months (rel beg)                                         |
|            Months (rel end)                                         |
|            Years                                                    |
|                                                                     |
|    Avoiding Sun  ---  ---  ---  --- ---  Sat  ---                   |
|                                                                     |
|    If not possible: Skip                                            |
|                     Delay current                                   |
|                     Delay all                                       |
|                     Catch up                                       |
|                                                                    |
+--------------------------------------------------------------------+
======================================================================
GNUbatch gbch-q (c) Free Software Foundation 2009   (? for help)
```

This screen displays all the start and repeat time parameters. If there are none, only the first line appears with `No` at the end. The Avoiding ... and If not possible ... parameters are only displayed if a repeat specification is set.

The second time and date displayed corresponds to the next time which would apply with the parameters given, and will be updated as the other parameters are changed.

The following key commands are available as appropriate:

Command	Function
`?`	Displays context sensitive help for the current field.
`Q`	Save changes and quit back to jobs screen
ESC	Leave unchanged and quit back to jobs screen.
TAB or *ENTER*	Move to the next field, or back to main screen as appropriate.
Back tab	Move to the previous field
`Y y T t S s`	Set field. i.e. Yes for time or Sun for first day field
`N n F f U u`	Un-set field. i.e. No for time or --- for first day field
`! ~`	Toggle setting.
`+`	Increment the value of the current field
`-`	Decrement the value of the current field
`h`	Move left a day in Avoiding days fields
`l`	Move right a day in Avoiding days fields
`0` to `9`	Type digits into a part of the time or date fields

9.1.3.3.1 Turn on or off time constraint

The first line of the time constraint display gives users the option to turn one on if it is off, or off if it is on.

9.1.3.3.2 Editing the time

If the job has not had any time constraint, then default parameters will be inserted. The defaults may be changed, by editing the gbch-q message file, see the chapter 11 on configurability for more information. As each part of the time is changed, the date and day of the week are altered also in step with the changes. Each part of the time may be edited separately.

9.1.3.3.3 Editing the repetition factor

The method of changing the repetition factor is to

1. Move the cursor to the required repetition factor (using j or *cursor down* and k or *cursor up*) and press ENTER. The selected repetition factor is underlined.
2. Type in the number of the selected units to be repeated by, and press ENTER,

or just press ENTER accept the default.

To change only the number of an existing repetition, move the cursor to it and press the `+` or `-` key. The value may now be typed in, incremented or decremented.

The *next time* field will be updated automatically.

The cases of *months (rel beg)* and *months (rel end)* are handled slightly differently. The day of the month is also required for months relative to the beginning, for example run every 2 months on the 5th. For months relative to the end the day back from the end of the month is required. In combination with *days to avoid* this allows for specifications like: the third working day from month end.

9.1.3.3.4 Days to avoid

The days displayed are those when repetition will not occur. It is treated as an error to avoid every day.

9.1.3.3.5 Reschedule options

To set the reschedule options, you move the cursor to the required option (using `j` or *cursor up* and `k` or *cursor down*) and press ENTER. The selected field will be highlighted. Press ENTER again or `q` to complete editing, or *BACKTAB* to return to the days to avoid.

To abort without saving changes, press ESC.

9.1.3.4 Change priority

To change the priority for the job, type `p` and key in a new priority for the job into the field, terminated by RETURN or ENTER.

Users are only able to change the priority within the range specified in the administration file. The default range, on installation of GNUbatch, is 100 to 200.

9.1.3.5 Change load level

Only users with *special create* privilege are allowed to alter load levels. Otherwise the load level is set to that of the command interpreter. The load level may be set to any value between 1 and 32767. To change the load level, press `l` and key in the new load level, terminated by RETURN or ENTER.

9.1.3.6 Progress Code

Providing a job is not running the *progress code* or state of the job can be changed. This can be done if the job halted with an error or abort and, after rectifying the problem, it needs to be reset so that it can run again.

The code may not be changed to `Run`. Changing it to `Cancelled` will not in this case invoke the assignments for job cancellation.

Press `P` which will give a prompt of the form

```
[Nil Done Error Abrt]? Nil
```

This shows the possible alternatives, and the default alternative. `Nil` is the ready to run or runnable state.

The following key commands are available:

Command	**Function**
ENTER	Accept current alternative.
TAB	Skip forward to next alternative.
BACKTAB	Skip back to previous alternative.
ERASE and explicit characters	Type in alternative directly.
ESC	Abort function.

An alternative may also be selected by typing in the first letter of the name, providing that it is unique in the list of alternatives.

The single-keystroke commands `r` and `z` are provided to enable the user to quickly set the job into the most-commonly required states of runnable or cancelled.

9.1.3.7 Force to run

The `g` command may be used to not only force a job into runnable state, but to bypass the time constraint for the current run and perform it immediately if possible

Conditions and load level limits are not bypassed and may still restrict the job from running. The job will not run until these pre-conditions are satisfied.

The `f` command is similar to the `g` command but does not advance the time after the job has run. The a command advances the time without running the job.

9.1.3.8 Kill or cancel job

To kill or cancel a job, move the cursor to it and type K, which gives the prompt:

```
[Int Quit Term Kill]? Term
```

Select the required alternative in the same way as for Progress Codes.

If the job is actually running, it will be killed with the selected signal, otherwise it will be set to cancelled state and any assignments flagged for *cancel* will be invoked.

If the process aborts with a signal, then it will be flagged as aborted. Note that if the process catches the signal and exits, then it will be considered to have terminated with an error, or if the exit code was zero, then it will be considered to have exited normally.

9.1.3.9 Process Parameters

Press the lower case u to open the window to view and edit the various process parameters: The screen displayed will look something like this:

```
Process environment for Job `Output Example' (1420)
--------------------------------------------------------------

Directory:      /users/wally/bin
```

```
Umask:          022

Ulimit:         3FFFFF

Normal exit:      0    0

Error exit:       1  255

Advance time

Export: Local only

Delete time         0
Maximum run time 0  0  0
Signal number    0  Run on time      0  0
```

The various components of this screen are:

Directory

specifies the directory which is set as the current working directory when the job is started. This can include environment variables, such as `$HOME` or refer to a user's home directory using expressions like `~/bin` to denote the directory `bin` from the home directory, or *`~tony`* to denote the given user's home directory.

Umask:

The Unix umask to be applied to the job

Ulimit:

Maximum size for files written by the job. Set to zero to avoid applying a limit.

Normal exit

Specifies the range of exit codes which the scheduler should interpret as indicating normal completion of the job.

Error exit

Specifies the range of exit codes which the scheduler should interpret as indicating the job terminated with some kind of error.

Advance time

Indicates that a repeating job should be advanced to the next scheduled run time even if it gave an error. If this is not set then instead of Advance time the string would appear as Do not advance time.

Export

Specifies the scope of the job across co-operating GNUbatch hosts. The options are:

Local only

Only visible and runnable on local machine

Export

Accessible on any machine but only runnable on the local machine.

Remote runnable

Can be accessed from and run on any machine.

Delete time

Specifies the time in hours from the last run of a job after which it will be automatically deleted. Zero means there is no delete time and the job will stay on the queue indefinitely.

Maximum ...

The Maximum elapsed run time (in hours minutes and seconds), before the job will be killed for over running. All zeroes indicates no time limit, hence the job may run unrestricted.

Signal number

Sets the signal which should be sent to an over running job to terminate it gracefully.

Run on time

Specifies how long, in minutes and seconds, that a job may run for after receiving the above signal before it is sent a `SIGKILL`.

Note that if the ranges for normal and error exits overlap, then the *smaller* range will be applied to any exit code which falls within both ranges.

If an exit code does not fall within either of the ranges, then it is treated as an abort case.

The following key commands are available:

Key	**Function**
`d`	Change working directory
`m`	Edit umask in octal
`l`	Edit ulimit in hexadecimal
`A`	Set or reset advance time on error
`N`	Set normal exit code range
`E`	Set error exit code range
`q`	Quit back to jobs screen
`X`	Set job export flags
`D`	Set delete time in hours. Zero lets the job stay on the queue indefinitely.
`R`	Specify maximum permitted elapsed run time.
`K`	Specify signal for killing job if it over runs
`g`	Set grace time from above signal until job sent a `SIGKILL`

Commands which prompt with a list of alternatives are handled in the same way as the Progress code described earlier.

The directory may contain environment variable names prefixed by `$` or constructs of the form `~`*`user`* to denote the home directory of the given user[4].

9.1.3.10 Change command interpreter

To change the command interpreter, press the lower case `x` which moves the cursor to that field for entry of the new command interpreter name.

Type in the new command interpreter as a string, e.g. `ksh`. Pressing the help key will prompt with a list of possible command interpreter names. If part of a name has already been given, the list is restricted to those starting with that string.

Pressing the space bar will cycle through possible command interpreter names. Likewise if part of a name has been given, the list of names cycled through is restricted to those starting with the character or characters already specified.

---

[4] This is probably most important for possibly remotely run commands as the directory structures may vary between machines.

Type RETURN or ENTER to accept the name offered. To abort the process, press ESC.

On successful completion, the command interpreter's load level will also replace the load level of the job.

9.1.3.11 Unqueue Job

Type an upper case U to unqueue a job, or just its specification, which prompts with:

```
[Unqueue Copy Save-home Options-current]? Unqueue
```

The four alternatives are:

`Unqueue`
: Remove the job from the queue and save it.

`Copy`
: Just make a copy of the job, do not delete it.

`Save-home`
: Create or edit a `.gnubatch` file in the home directory, using the options used for this job as default options for gbch-r commands.

`Options-current`
: As above but use the current directory.

For the first two alternatives, a sub-window is generated of the form

```
+------------------------------------------------------------+
|Unqueuing job `Output Example' (1420)                       |
|                                                            |
|Directory to write in /users/wally                          |
|Command file                                                |
|Job file                                                    |
+------------------------------------------------------------+
```

The default directory is the current directory when gbch-q was started, but this may be over-typed with any other directory name.

The two file names prompted for are a *command file*, into which a gbch-r command will be placed suitable for re-submitting the job with the same parameters, and a *job file*, containing the shell script or standard input for the job. The command file will be made executable (i.e. it will become a shell script), and it will name the job file as the source of the standard input.

For the *Save-home* and *Options-current* alternatives, the sub-window does not have fields for the Command and Job files, only the directory is prompted for, with the initial directory the home or the current directory respectively.

The files may be edited and the job re-submitted by executing the command file. On machines with some *read-only* environment variables this may fail. In this case invoke the program btresub.sh with the name of the command file to re-submit the job.

Note that to unqueue the job, a user must have read, read-mode and delete permission.

9.1.3.12 Set mail/write message on job completion flags

Press `F` to view and change the mail and write flags, which opens this sub-window:

```
+------------------------------------------------+
|Mail/Write Flags for job 'Output Example' (1420)|
|                                                |
|Mail user at end of job          No             |
|Write message to user's terminal No             |
+------------------------------------------------+
```

These options do not affect the output from the job. The output is always e-mailed back to the user unless it has been redirected. Only the job completion messages are affected.

The available commands for this sub-window are:

Command	**Function**
`Y T S`	Set current flag
`N F U`	Un-set current flag
`! ~`	Reverse current flag
`q`	Quit back to job screen
ESC	Abort edit and leave unchanged

9.1.3.13 Setting job arguments

Press `A` to open the job arguments screen, which will look like this:

```
Command arguments for Job `Accounts' (9309)
Numb Value

   1 'statements'
   2 'june'
```

Arguments can include job parameters, by using the `%` symbols in the same way as environment variables and I/O redirections, as described under Meta-Data (see page 49).

The following, context specific, key commands are available:

Key	**Function**
`i`	Insert new, copied or moved argument before the current one.
`a`	Insert new, copied or moved argument after the current one.
`d`	Delete current argument
`m`	Mark argument for moving. Pressing m again cancels.
`c`	Mark argument for copying. Pressing c again cancels.
`E`	Edit the text of the current argument.
`q`	Quit back to jobs screen

To move an argument, first place the cursor on the argument to be moved and press `m`. The (Moving) flag appears at the top right corner of the screen. Next move the cursor to the destination position, and type `i` or `a` to insert the argument before or after the line.

To copy an argument follow the same procedure, but press `c` to copy, instead of `m`.

9.1.3.14 Editing the environment

The procedure for setting the environment of a job is similar to that for the arguments, except that there is no copy facility. Press `E` to open the Environment list for the job, which will look something like this:

```
Environment variables for job `Audio:playback' (1082)
Numb     Name
    Value
--------------------------------------------------------------
1        MANPATH
    :/usr/share/man:/usr/motif/man:/usr/local/man
2        HZ
    100
3       PATH
    :/usr/bin:/usr/ccs/bin:/usr/local/bin:/usr/ucb:/usr/X386/bin:/usr/
motif/bin:/home/int/jmc/bin
4       CDPATH
    :..:/home/int/jmc:/home/int/work:/home/products
```

The following, context specific, key commands are available

Key	Function
`i`	Insert new or moved environment variable before the current one.
`a`	Insert new or moved environment variable after the current one.
`d`	Delete current environment variable
`m`	Mark environment variable for moving. Pressing m again cancels.
`N`	Rename the current environment variable.
`V`	Edit the value (i.e. contents) of the current environment variable.
`q`	Quit back to jobs screen

To change the name of an environment variable press `N`. To change the value, press `V`. Note that the value (or contents) of an environment variable may extend over several lines.

To move an environment variable, first place the cursor on the variable to be moved and press `m`. The (Moving) flag appears at the top right corner of the screen. Next move the cursor to the destination position, and type i or a to insert it before or after the line. The order of environment variables is not known to have an effect with any existing software.

Note that the `PWD` environment variable is not copied out when the job is unqueued. This is because many shells object to assignments to this variable.

`%` sequences are expanded in environment variables, in the same way as in arguments and I/O redirections.

9.1.3.15 Editing redirections

First move to the relevant job and press R. This opens a screen looking like:

```
I/O Redirections for job `Output Example' (1420)
Numb File  Type
    File/Process
1    1 (stdout)    Write
    /tmp/results_j%d1
2   2 (stderr)     Append
    /tmp/logfile_j$d1
```

The following, context specific, key commands are available:

Key	Function
i	Insert new or moved redirection before the current one.
a	Insert new or moved redirection after the current one.
d	Delete current redirection.
m	Mark redirection for moving. Pressing m again cancels.
N	Change the file number.
A	Change the Action (i.e. read, write, append, ...)
F	Edit the file / process.

The commands N, A and F respectively enable editing of the file descriptor number, *action* and file name (or number in the case of *dup descriptor* actions).

% sequences can be used to insert job parameters, as described under Meta-Data (see page 49).

9.1.3.16 Job Assignment Editing

Up to 8 assignments are allowed per job. Select the required job and press S, to open a window like this:

```
Assignments for job `prog_a' (451)

  Variable           Start  Reverse Normal  Error   Abort   Cancel
Oper     Value to set

  A_COUNT            --     --      Set     Set     Set     Set
Add      1
  A_STATUS           --     --      Set     --      --      --
Set      1
  A_STATUS           --     --      --      Set     Set     --
Set      999
```

The available key commands are:

Key	Function
D	Delete this assignment
I	Insert new assignment before current line.
a	Insert new assignment after current line

Key	Function
`V`	Select a different variable for assignment
`=`	Change assignment operation
`N`	Change value assigned
`S`	Edit **start** flag
`R`	Edit **reverse** flag
`O`	Edit **normal exit** (OK) flag
`E`	Edit **error exit** flag
`A`	Edit **abort exit** flag
`C`	Edit **cancel** flag
`T Y S`	Set flag
`F N U`	Clear flag
`~ !`	Toggle flag
`c`	Toggle *critical* mark for remote variables

When a new assignment is created, the fields for variable name, operation and value are prompted for. The flags are set from configurable defaults. As distributed these are: set *start*, *reverse*, *normal exit*, *error exit* and *abort*.

9.1.3.16.1 Choosing the Variable

Requesting help whilst entering or changing the variable name gives a list of available variables. If one or more characters of a variable name have already been entered the list will be restricted to just those variables starting with the character(s) typed.

Pressing the space bar whilst entering or changing the variable name will cycle the field through the list of possibilities. If part of the name has already been entered, the list of variables cycled through will be restricted to variables starting with those characters.

9.1.3.16.2 Specifying the Value to be Assigned

The value may be an integer number or a string. Numeric values are recognised by a leading digit or a "`-`" sign. Strings are recognised by the leading character not being numeric or a "`-`" sign. The first string character will be prefixed automatically by a " character, but this should not be typed at the end.

To enter a string which starts with a digit or "`-`", precede it by a double quote, `"`. This will be echoed, but will not form part of the string.

9.1.3.16.3 Specifying the Assignment Operation

Editing the assignment operation prompts with the set of alternatives, like this:

```
Set Add Subtract Multiply Divide Modulus Exit Signal]? Set
```

These are handled in the same way as described for the Progress Codes.

Only Set is permitted with string values.

9.1.3.16.4 Setting the flags

Press the appropriate key for the assignment flag that needs changing. Then press the key to set, clear or toggle the flag as required. Note that if the reverse flag is set, the effected exit conditions are highlighted.

To mark a variable assignment as critical press `c`. Press `c` again to clear the assignment critical flag.

9.1.3.17 Set job conditions

Type `C` to open the job pre-conditions sub-window, which

```
+------------------------------------------------------------+
| Conditions for job `Audio:playback' (1080)                 |
|                                                            |
| Variable          Cond Value                               |
|                                                            |
| Audio_Lock          >  0                                   |
| Audio_Status        =  "three"                             |
|                                                            |
|                                                            |
|                                                            |
|                                                            |
|                                                            |
|                                                            |
+------------------------------------------------------------+
```

New conditions are added to the end of the list. Up to a maximum of 10 conditions may be specified for each job.

The available key commands are:

Command	**Function**
`D`	Delete the selected condition
`a`	Add new condition
`V`	Change variable used in condition
`C = ! < >`	Change comparison operator
TAB	Cycles alternatives when changing comparison operator
`N`	Change value compared against
`c`	Toggle *critical* mark for remotes

9.1.3.17.1 Choosing the Variable

Asking for help whilst entering or changing the variable name, will give a list of available variables. If one or more characters of a variable name have already been entered, the list is restricted to names starting with those characters.

Pressing the space bar whilst entering or changing the variable name will cycle the field through the list of possibilities. If part of the name has already been entered, the list of variables cycled through will be restricted to variables starting with those characters.

9.1.3.17.2 Specifying the Comparison Operator

Editing the comparison operator prompts with the set of alternatives, like this:

```
[= != < <= > >=] !=
```

These are handled in the same way as described for the Progress Codes, plus the operator may be erased and typed in directly.

Only "`=`" and "`!=`" are guaranteed to work consistently with strings.

9.1.3.17.3 Specifying the Value to Compare Against

The value may be an integer number or a string. Numeric values are recognised by a leading digit or a "`-`" sign. Strings are recognised by the leading character not being numeric or a "`-`" sign. The first string character will be prefixed automatically by a `"` character, but this should not be typed at the end.

To enter a string which starts with a digit or "`-`", precede it by a double quote, `"`. This will be echoed, but will not form part of the string.

9.1.3.18 Change Owner

To change the ownership of a job, a suitable user must nominate the new owner, for which *give away* permission applies. Then the new owner must accept the transfer, for which *assume ownership* applies. Nothing effectively happens until these two stages have been completed, and the job list display will not change after the first step.

If the user has *write administration file* privilege, then these checks are bypassed and the change of owner is immediately effective.

Type `O` to change the owner, which will prompt with

```
New owner for job 'memo' currently jmc:
```

Enter the new owner as a string, e.g. `root`, or as a numeric user id[5]. Asking for help prompts with a list of possible user names. If the first one or more characters of the user name has been typed in, only names starting with those characters are listed.

To cycle through possible user names, press the space bar. The list of names cycled through can be restricted by entering the one or more characters.

To abort the process, press *ESC*.

9.1.3.19 Change Group

To change the group of a job it has to be nominated to the new group by a suitable user, for which *give away group* permission applies. It then has to be accepted by someone in the new group, for which *assume group ownership* applies. Nothing effectively happens until these two stages have been completed.

If the user has *write administration file* privilege, then these checks are bypassed and the change of group is immediately effective.

---

[5] Although we suggest you avoid this.

Pressing `G` to change group causes gbch-q to prompt with:

```
New group for job 'memo' currently users:
```

Type in the new group as a string, e.g. `other`, or as a numeric group id[6]. Pressing the help key will prompt with a list of possible group names. If part of a group name has already been given, the list is restricted to those starting with that string.

Pressing the space bar will cycle through possible group names. Likewise if part of a group name has been given, the list of groups cycled through is restricted to those starting with the character or characters already specified.

To abort the process, press *ESC*

9.1.3.20 Mode Editing

Press `M` to edit the job modes (or access permissions). This opens a sub-window that looks something like this:

```
+--------------------------------------------+
|Modes for Job 'update' (6943)               |
|Job owner jmc group users                   |
|                                            |
|                       User   Group  Others |
|Read                   Yes    Yes    No     |
|Write                  Yes    No     No     |
|Reveal                 Yes    Yes    Yes    |
|Display mode           Yes    Yes    Yes    |
|Set mode               Yes    No     No     |
|Assume ownership       No     No     No     |
|Assume group ownership No     No     No     |
|Give away owner        Yes    No     No     |
|Give away group        Yes    Yes    No     |
|Delete                 Yes    No     No     |
|Kill (jobs only)       Yes    No     No     |
+--------------------------------------------+
```

If the user has *Set mode* access, the following key commands are available:

Command	**Meaning**
`Y T`	Set corresponding permission, move right
`N F`	Unset corresponding permission, move right
`! ~`	Invert permission and move right

Note that some permissions, where it does not make sense to have one without the other, are coupled together. For example if turning on *read* permission, the *reveal* permission will be turned on at the same time.

Type `q` to quit back to the main job screen.

---

[6] Although we suggest you avoid this.

9.1.4 The Variables Screen and Commands

The variables screen may be displayed automatically on entry, if there are no jobs visible to the user or it has been set as the default. To switch to the variables screen from the job list type an upper case `V`.

A typical screen for the, default configuration, variable list might look like this:

```
 Name                  Value                                Exp/Loc
     Comment                                  User    Group
---------------------------------------------------------------------

 CLOAD                 0
     Current value of load level              batch   bin

 LOADLEVEL             20000
     Maximum value of load level              batch   bin

 LOGJOBS
     File to save job record in               batch   bin

 LOGVARS
     File to save variable record in          batch   bin

 MACHINE               voyager
     Name of current host                     batch   bin

 STARTLIM              5
     Number of jobs to start at once          batch   bin

 STARTWAIT             30
     Wait time in seconds for job start       batch   bin

=====================================================================
GNUbatch (variables)(c) Free Software Foundation 2009 (? for help)
```

In this example, only the system variables have been set up and the user has sufficient permissions to see all of the information for all of them. The variables `LOGVARS` and `LOGJOBS` contain the empty or null string.

The following key commands are available:

Command	Function
`D`	Delete the variable. Confirmation is requested - reply Y or N.
`M`	Change the variable modes (access permissions)
`O`	Change the owner of the variable
`G`	Change the group of the variable
`"`	Change the comment for the variable
`R`	Rename the variable
`C`	Create a new variable
`A`	Assign a new value to the variable
`=`	Reset constant for arithmetic

Command	Function
`+ - * / %`	Apply arithmetic operation (constant on *rhs*) to variable
`$`	Set program options
`J`	Switch to job list screen
`L`	Set variable as local
`E`	Set variable as export
`~`	Toggle export state of variable
`U`	Set variable clustered
`K`	Set variable not clustered
`&`	Toggle clustered setting
`^`	Search forwards for variable name or title
`\`	Search backwards for variable name or title
`,`	Set format and content of top line for each variable
`;`	Set format and content of bottom line for each variable

9.1.4.1 Assign new value

To assign a new value to the variable, press `A`. The cursor will move to the value field for entry of the new data, may be either an integer or a string. Numeric values are recognised by a leading digit or minus sign and strings are recognised by any other initial character. The first string character will automatically be prefixed by a `"` character.

To enter a string which happens to start with a digit or "`-`", precede it with a double quote `"` character. This will force the value to string and be echoed, but will not be taken as part of the string.

Type *ESC* to abort and leave the value unchanged.

9.1.4.2 Arithmetic operations

Arithmetic operations, like increment and decrement use a constant which is initially set to one. The constant maybe changed by pressing the "`=`" key. You will be prompted for a new value on the current screen line, and the header will be updated.

The constant may be applied to any variable by moving the cursor to it and typing `+ - * /` or `%` (the last gives modulus, i.e. remainder when divided by the constant). The variable must be numeric to start with.

9.1.4.3 Change comment

The comment field is a free text string, which GNUbatch maintains for documenting the purpose of variables. The comment field has no effect on the behaviour of variables or the scheduling of jobs.

To change the comment for a variable, type `"` and key in any string of characters, pressing ENTER or RETURN to complete the operation.

It is possible to display a help message whilst this is happening, but the help command will have to be bound to a non-printing character, as `?` is permitted as part of the comment.

9.1.4.4 Create new variable

Press `C` to create a new variable, which opens this sub-window:

```
+--------------------------------------------------------+
|                     Create new variable                |
|                                                        |
|Name:                                                   |
|                                                        |
|Value:                                                  |
|                                                        |
|Comment:                                                |
+--------------------------------------------------------+
```

gbch-q prompts through each field in turn. Press *ESC* at any stage to abort.

Variable names are restricted to alphanumeric characters and underscore starting with an upper or lower case letter.

The value may contain either an integer or a string. Numeric values are recognised as by a leading digit or "`-`" sign. Strings are recognised as by the entry of any printing character other than a digit or "`-`" sign. The first string character will be prefixed automatically by a `"` character (which will not form part of the string), but this should not be typed at the end.

To enter a string which happens to start with a digit or "`-`", type a double quote first. This will be echoed, but will not form part of the string.

The comment is free text string made up of spaces and any printable characters.

For example, creating a variable to control and show the progress of a chain of jobs relating to the pay roll might look like this:

```
+--------------------------------------------------------+
|                     Create new variable                |
|                                                        |
|Name:        PayRoll_Progress                           |
|                                                        |
|Value:       "Start_Wait                                |
|                                                        |
|Comment:     Progress through queue of pay roll jobs    |
+--------------------------------------------------------+
```

9.1.4.5 Rename variable

A variable may be renamed using the `R` command, which prompts with:

```
New name for variable cant:
```

The name must begin with an upper or lower case alphabetic character. The rest of the name may be any combination of alphanumeric and underscore characters.

Only users with *delete* permission on the variable may rename it.

The variable references in job assignments and conditions will be updated, including those on remote machines. Users cannot rename a remote variable.

9.1.4.6 Mode Editing

Move the cursor to the required variable and press M to open the Modes sub-window, which will look something like this:

```
                   +--------------------------------------------+
   LOADLEVEL       |Modes for Variable `LOADLEVEL'              |
       Maximum val |Variable owner batch group bin              |
                   |                                            |
   LOGJOBS         |                       User   Group  Others |
       File to sav |Read                   Yes    Yes    No     |
                   |Write                  Yes    No     No     |
   LOGVARS         |Reveal                 Yes    Yes    Yes    |
       File to sav |Display mode           Yes    Yes    Yes    |
                   |Set mode               Yes    No     No     |
   MACHINE         |Assume ownership       No     No     No     |
       Name of cur |Assume group ownership No     No     No     |
                   |Give away owner        Yes    No     No     |
   STARTLIM        |Give away group        Yes    Yes    No     |
       Number of j |Delete                 Yes    No     No     |
                   +--------------------------------------------+
```

The sub-window is brought up over the top of the variable screen, in the same way as sub-windows for jobs.

The context specific key commands available in the modes sub-window are:

Command	**Meaning**
`Y T`	Set corresponding permission
`N F`	Unset corresponding permission
`! ~`	Toggle permission

The cursor moves on to the next column or row after each command. The usual movement commands allow navigation forwards and backwards through the modes sub-window.

Note that some permissions, where it does not make sense to have one without the other, are coupled together. For example, turning on *read* permission also turns on the *reveal* permission at the same time.

Type `q` to quit back to the main variables screen saving the changes.

9.1.4.7 Change Owner

To change the ownership of a variable a user, with *give away* permission, must nominate the new owner. The nominated owner must then accept the transfer, for which *assume ownership* applies. Nothing effectively happens until these two stages have been completed. The job list display will not change after the first step.

If the user has *write administration file* privilege, then the checks are bypassed and the change of owner is effective immediately.

Pressing `O` to change the owner prompts with:

```
New owner for variable 'count' currently jmc:
```

Type in the new owner as a string, e.g. `tony`, or as a numeric user id[7]. Asking for help will be give a list of possible user names. If part of a user name has already been given, the help message will be restricted to user names beginning with those characters.

To cycle through possible user names press the space bar. If part of a user name has already been entered, the list of users cycled through is restricted to those starting with the characters already given.

To abort the process, press *ESC*.

9.1.4.8 Change of group

This is almost identical to change of owner. To change the group of a variable a user with *give away group* permission must nominate the new group. Then a user with *assume group ownership* must accept the transfer. Nothing effectively happens until these two stages have been completed.

If the user has *write administration file* privilege, then these checks are bypassed and the change of group is immediately effective.

Enter `G` to change the group, which prompts with:

```
New group for variable 'memo' currently users:
```

Type in the new group as a string, e.g. `other`, or as a numeric group id[8]. Asking for help will give a list of possible group names. If part of a group name has already been entered, the help message will be restricted to groups beginning with those characters.

To cycle through possible groups press the space bar. If part of a group name has already been entered, the list of names cycled through is restricted to those starting with the characters already given.

To abort the process, press *ESC*.

9.1.5 Command interpreter list

This facility is only available to users with *special create* privilege. From the Job screen, press `X` to review and edit the command interpreter list, which will look like this:

```
   Command Inter   Predef args        Load Level   Nice
       Path name
------------------------------------------------------------------

   sh              -s                       1000     24
       /bin/sh

   ksh             -s                       1000     24      Set A0
```

---

[7] Although we suggest you avoid this.

[8] Although we suggest you avoid this as well.

```
/bin/ksh
```

On a new installation of GNUbatch there will only be one Command Interpreter specified. This will probably be the Bourne shell, `sh`, and it will not have the "`Set A0`" flag set.

The following key commands are available:

Command	**Function**
`q`	Quit back to jobs screen
`A`	Add new entry
`D`	Delete current entry
`L`	Reset load level
`a`	Set pre-defined args
`n`	Reset nice value
`N`	Set name
`P`	Set path
`0` (zero)	Toggle the "arg 0" flag
`e`	Toggle the "expand args" flag.

9.1.5.1 Setting Up A Command Interpreter

To add a command interpreter press `A`, which moves the cursor to the "Command Inter" field. Enter a name by which the command interpreter will be known and press RETURN or ENTER. Next the "Path name" field is selected for entry of the full path and name of the command interpreter program. All the remaining fields are initialised to default values which should be edited as required.

The load level will be initialised to the user's special create load level.

The pre-defined arguments, if any, are prepended to any arguments supplied to the command interpreter by the job. This is particularly important with shells, and the default setting for the Bourne shell is to include the `-s` argument, which prevents it trying to interpret the first argument to the job as a shell script name. The `a` command causes these arguments to be reset.

These pre-defined arguments may include the `--` option. This causes subsequent - options to be passed to the job script and not the shell.

The nice value is an *absolute* nice value. The scheduler process runs at the highest priority and jobs are run with nice set to the given value. User processes from a login prompt usually run with a nice value of 20. Hence, values less than 20 represents a higher priority and values greater than 20 represents lower priorities than a login process. Command interpreters are created with an initial nice value of 24.

9.1.6 Edit holiday list

To edit the table of holidays used by the *days to avoid* setting, users must have *write administration file* privilege. Other users may view but not edit the list. The years up to 2099 are supported.

Press `H` to display a calendar for the current year with holidays highlighted. The calendar looks like this before any holidays have been specified:

```
                                      2001

Jan     1  2  3  4  5  6  Feb              1  2  3  Mar              1  2  3
     7  8  9 10 11 12 13       4  5  6  7  8  9 10       4  5  6  7  8  9 10
    14 15 16 17 18 19 20      11 12 13 14 15 16 17      11 12 13 14 15 16 17
    21 22 23 24 25 26 27      18 19 20 21 22 23 24      18 19 20 21 22 23 24
    28 29 30 31               25 26 27 28               25 26 27 28 29 30 31
Apr  1  2  3  4  5  6  7  May        1  2  3  4  5  Jun                 1  2
     8  9 10 11 12 13 14       6  7  8  9 10 11 12       3  4  5  6  7  8  9
    15 16 17 18 19 20 21      13 14 15 16 17 18 19      10 11 12 13 14 15 16
    22 23 24 25 26 27 28      20 21 22 23 24 25 26      17 18 19 20 21 22 23
    29 30                     27 28 29 30 31            24 25 26 27 28 29 30
Jul  1  2  3  4  5  6  7  Aug           1  2  3  4  Sep 30                 1
     8  9 10 11 12 13 14       5  6  7  8  9 10 11       2  3  4  5  6  7  8
    15 16 17 18 19 20 21      12 13 14 15 16 17 18       9 10 11 12 13 14 15
    22 23 24 25 26 27 28      19 20 21 22 23 24 25      16 17 18 19 20 21 22
    29 30 31                  26 27 28 29 30 31         23 24 25 26 27 28 29
Oct     1  2  3  4  5  6  Nov              1  2  3  Dec 30 31              1
     7  8  9 10 11 12 13       4  5  6  7  8  9 10       2  3  4  5  6  7  8
    14 15 16 17 18 19 20      11 12 13 14 15 16 17       9 10 11 12 13 14 15
    21 22 23 24 25 26 27      18 19 20 21 22 23 24      16 17 18 19 20 21 22
    28 29 30 31               25 26 27 28 29 30         23 24 25 26 27 28 29
```

Sundays and Saturdays may be rendered "dim" if the screen enhancements permit this.

The following key commands are available:

Key	Function
TAB	Move to next month
N P	Go to Next / Previous year
`y t s`	Set current day as holiday
`f u n`	Clear current day, i.e. set as NOT a holiday
`! ~`	Toggle holiday state of current day

9.1.7 Setting program options

To select a new set of program options and optionally to save them type `$`. A screen is displayed as follows:

```
Setting Program options for gbch-q

Job queues (pattern)        :
Include null queue names    : Yes
Display only user :
Display only group:
Confirm abort/delete jobs   : Always
If job moves      : Follow job
Local jobs/vars   : All jobs/vars
Clear help message: Use next command
Help messages     : Inverse video
Error messages    : Inverse video
```

```
Screen on entry   : Don't care
```

To cycle through the options for each parameter press the space bar. Parameters like the queue, user and group names allow the specification to be typed in.

Type `q` to exit from any parameter which does not accept free text. Alternatively just press return for each parameter until the cursor moves off the last parameter. On exiting from this screen, the option to save the parameters settings is prompted for:

```
Save parameters?
```

Opt to do so by typing `y`, which asks whether to save the settings in the current directory or the user's home directory. Type `n` to avoid saving the changes.

Options take effect immediately, apart from the *screen or entry* setting.

9.1.8 Setting Display Contents

Within gbch-q it is possible to alter the appearance of the screens. A user can change the size and ordering of the fields to suit their specific tastes. The user can also customise the heading title strings, perhaps to support a language other than English.

9.1.8.1 Display Format for the Jobs Screen

To bring up the display settings screen for the job screen, go in to the job list and press "`,`" the comma key. The default job list format will look like:

```
Job list formats
    Width  Code
        3  n   Sequence
" "
 <      7  N   Job number
" "
        7  U   User
" "
       13  H   Title (in full)
" "
       14  I   Command Interpreter
" "
        3  p   Priority
        5  L   Load Level
" "
        5  t   Time or date
" "
        9  c   Conditions (abbreviated)
" "
 <      4  P   Progress
```

Each line represents either a field on the main screen or a separator. The `<` specifies that field may overflow onto its left hand neighbour if necessary. The number is the field width. The letter represents the key that is used to access the field and the last entry is the column heading. The adjacent fields will be separated by, the character or characters, in between the double quotes.

The job list can only ever show a sub-set of the job fields. The complete set of fields available for inclusion in the job list are:

`A`	Arguments	`a`	Avoiding
		`b`	Start time
`C`	Conditions (in full)	`c`	Conditions (abbreviated)
`D`	Directory	`d`	Delete time
`E`	Environment	`e`	Export / Remote runnable
		`f`	End time
`G`	Group	`g`	Grace period
`H`	Job Name (in full)	`h`	Title (no queue name)
`I`	Command Interpreter	`i`	Process id
		`k`	Signal to kill over running job with
`L`	Load Level	`l`	Maximum elapsed run time
`M`	Mode	`m`	Umask
`N`	Job id number	`n`	Sequence
`O`	Originating host	`o`	Time submitted
`P`	Progress	`p`	Priority
		`q`	Queue name
`R`	I/O Redirections	`r`	Repeat specification
`S`	Assignments (in full)	`s`	Assignments (abbreviated)
`T`	Date and time (in full)	`t`	Time or date
`U`	User	`u`	Ulimit
`W`	Last / Next time		
`X`	Exit code ranges	`x`	Exit code returned by last run
`Y`	Holiday dates being avoided	`y`	Signal num last run terminated by

9.1.8.2 Display Format for the Variables Screen

This differs from the job screen because two lines of information are used to represent each variable. These lines are specified independently of each other.

To bring up the display specification screen for the top line of all the variables, go to the variable screen and press "`,`" the comma key. The default format will look like:

```
Variable list format 1
    Width  Code
" "
       22  N  Name of variable
" "
       41  V  Value of variable
" "
       13  E  Export or Local
```

To bring up the display specification screen for the bottom line of all the variables, go to the variable screen and press "`;`" the semi-colon key. The default format will look like:

```
Variable list format 2
    Width  Code
```

```
"    "
       44   C   Comment
" "
        7   U   User Owner
" "
        7   G   Group Owner
```

The set of fields available for inclusion in the variable list are:

C	Comment
E	Export or Local State
G	Group variable belongs to
M	Modes
N	Name of variable
U	User who owns variable
V	Value
K	Clustered marker

9.1.8.3 Editing the Display Formats

The following key commands are available from within any of the display options screens:

Key	Function
i	Insert a new field before the current field
a	Insert a new field after the current field
'	Insert a new separator before the current field
"	Insert a new separator after the current field
w	Set the width of the current field
c	Set the shortcut key code for this field
<	Toggle the left flag (see below)
D	Delete the current field
S	Set the current separator string

The `i` and `a` commands insert a new field either before or after the field marked by the cursor. When inserting a new field, gbch-q prompts the user for the code of the new field. The user should enter the required code, as shown in the previous sub-section, then hit return. To move a field use the delete command to remove the original and the insert commands to position a new copy.

The `'` and `"` commands insert a new separator field either before or after the current field.

The `w` command alters the width of the current field. Pressing `w` moves the cursor across to the width field of the current selection, for entry of the new value.

The `c` command alters the character used to access the current field, from the main job window. Take care to avoid clashes.

The `<` toggle enables or disables field overflow into the left hand neighbouring field. When enabled, if the contents of a particular field are wider than the field itself gbch-q overwrites the field to the left. If the toggle is off, gbch-q truncates oversized field contents.

The `D` command deletes the current field.

Use the `S` command to change the separator from the default space between each field.

Press the *ESC* key, at any time, to abort the current operation.

When leaving the display option screen gbch-q asks if it should save the changes, as the default. Press `y` to save the changes as the new settings, otherwise press `n`. When saving the changes, gbch-q prompts for the location, in which to save, and then the name of, the configuration file.

Saving the changes results in an entry in the relevant `.gnubatch` file, of the form:

```
BTQCONF=filename
```

Filename is the name entered when saving. To undo any changes remove or comment out the `BTQCONF` line.

See the chapters on configurability (see page 210) and extensibility (see page 228) for details of more ways to customise the operation and displays of gbch-q. Here is an example showing a job screen with function keys specified, different header, footer, format and content:

```
Seq   Job Name             Args       Date/Time          Prog
-----------------------------------------------------------------------
  1   start                           08/02/01 10:54     Canc
  2   Process directory    /home      08/02/01 10:54
  3   Process directory    /usr       08/02/01 10:54
  4   Process directory    /tmp       08/02/01 10:54
  5   Collect data                    08/02/01 10:54
  6   Error Handler                   08/02/01 10:54
  7   cleanup                         08/02/01 10:54
  8   setup                           29/01/01 23:01     Done



----F1------F2------F3------F4------F5-----F6--------------------------
   help   enable  disable   set    view   view
          run     run       time   job    vars
-----------------------------------------------------------------------
         GNUbatch gbch-q (c) Free Software Foundation 2009
```

9.2 gbch-user - Interactive user administration tool

gbch-user is both a simple report generator and an interactive tool for administering user privileges and settings. It has four modes of operation, which are:

Listing

Produces a simple report for the current user showing their modes and permissions.

Mode edit

Enables the current user (if permitted, with change default modes privilege) to interactively display and adjust their default modes for creation of jobs and variables.

View mode

Allows users with *read administration file* privilege to interactively view the entire list of permissions for all users, but not to make any changes.

Update mode

Gives full interactive access to GNUbatch administrators to view and edit the entire list of permissions for all users. For this purpose the user must have *write administration file* privilege.

The mode can be specified as an option on the command line. Like the other interactive programs there are no `+freeze-current` or `+freeze-home` keywords, as these facilities are provided by an interactive screen within the program[9].

9.2.1 Display current permissions

With no options, or the option -d, a report is output for the settings of the current user. For an ordinary user, with the default installation settings, the report will look like:

```
The GNUbatch account for user tony group staff.
Minimum priority 100 maximum 200 default 150
Maximum load level 1000 total load 10000
Special jobs are allocated a load of 1000
Current charge is 319.15 units.
Privileges are: Create entry, Change default modes
Job  - Read by: User, Group
Job  - Write by: User
Job  - Reveal by: User, Group, Others
Job  - Display mode by: User, Group, Others
Job  - Set mode by: User
Job  - Give away owner by: User
Job  - Give away group by: User, Group
Job  - Delete by: User
Job  - Kill (jobs only) by: User
Var  - Read by: User, Group
Var  - Write by: User
Var  - Reveal by: User, Group, Others
Var  - Display mode by: User, Group, Others
Var  - Set mode by: User
Var  - Give away owner by: User
Var  - Give away group by: User, Group
Var  - Delete by: User
```

This display also gives the user's current charge.

9.2.2 Mode Edit

Users who have *change default modes* privilege, may change their default modes by using the -m option. The gbch-user `-m` display will look like:

---

[9] Although for consistency these are included in gbch-user for GNUbatch 2

```
Modes for user tony

                       Jobs           Vars
                       U   G   O    U   G   O
Read                   Yes Yes No   Yes Yes No
Write                  Yes No  No   Yes No  No
Reveal                 Yes Yes Yes  Yes Yes Yes
Display mode           Yes Yes Yes  Yes Yes Yes
Set mode               Yes No  No   Yes No  No
Assume ownership       No  No  No   No  No  No
Assume group ownership No  No  No   No  No  No
Give away owner        Yes No  No   Yes No  No
Give away group        Yes Yes No   Yes Yes No
Delete                 Yes No  No   Yes No  No
Kill (jobs only)       Yes No  No
```

The standard screen command keys, as described in the section on program gbch-q, plus the following context specific key commands are available:

Key	Function
?	Display help message
B	Beginning row
E	End row
J	Jobs column
V	Variables column
Y T	Set corresponding permission, move right
N F	Unset corresponding permission, move right
! ~	Invert permission and move right
$	Save program options

Note that some permissions, where it does not make sense to have one without the other, are coupled together. For example if the *read* permission is turned on, the *reveal* permission will be turned on at the same time if it is unset.

9.2.3 View and edit permissions

The display for the View, `-v`, and Edit, `-i` options is similar, so they are treated together. gbch-user goes into the main screen, which looks something like this:

```
User     Group    Def Min Max Maxll Totll Spcll  Privs
DEFAULT         150 100 200  1000 10000  1000  CR|Cdft

root    other   150 100 200  1000 10000  1000  RA|WA|CR|SPC|ST|Cdft|UG|UO|GO
daemon  other   150 100 200  1000 10000  1000  CR|Cdft
bin     bin     150 100 200  1000 10000  1000  CR|Cdft
sys     sys     150 100 200  1000 10000  1000  CR|Cdft
adm     adm     150 100 200  1000 10000  1000  CR|Cdft
uucp    uucp    150 100 200  1000 10000  1000  CR|Cdft
nuucp   nuucp   150 100 200  1000 10000  1000  CR|Cdft
listen  adm     150 100 200  1000 10000  1000  CR|Cdft
spooler bin     150 100 200  1000 10000  1000  CR|Cdft
batch   bin     150 100 200  1000 10000  1000  RA|WA|CR|SPC|ST|Cdft|UG|UO|GO
lp      lp      150 100 200  1000 10000  1000  CR|Cdft
```

```
wally   staff   150 100 200  1000 10000  1000  RA|WA|CR|SPC|ST|Cdft|UG|UO|GO
pior    staff   150 100 200  1000 10000  1000  CR|Cdft|UO
tony    staff   150 100 200  1000 10000  1000  CR|Cdft
jmc     staff   150 100 200  1000 10000  1000  RA|WA|CR|SPC|ST|Cdft|UG|UO|GO
nobody  nobody  150 100 200  1000 10000  1000  CR|Cdft
noaccessnoaccess150 100 200  1000 10000  1000  CR|Cdft

============================================================================
GNUbatch gbch-user (c) Free Software Foundation 2009      (? for help)
```

The top row of permissions, headed `DEFAULT`, represents the default values given to any new user. The table underneath lists all of the individual users, which can be scrolled through if it is longer than will fit on one screen. Use `^` or `\` to search forward or backwards respectively.

The leftmost two columns show the individual user and group, respectively. This is followed by the users default, minimum and maximum permitted job priorities. If no priority is specified when a job is submitted the default is taken.

Next is the maximum permitted load level, `Maxll` for any one job owned by the user. This is followed by the maximum total load level, `Totll`, of jobs running at one time for the user. The scheduler sums the Load of all the jobs currently running for each user and ensures that it does not exceed the maximum total load level.

Last of the numeric fields is the default *special create* load level, `Spcll`, used when a suitably privileged user sets up a new command interpreter. This only applies if the user has got *special create* privilege, SPC. In the above example only users `root`, `batch`, `wally` and `jmc` have *special create* privilege.

This is followed by a list of the privileges granted to the user, which are represented using the following abbreviations:

Abbreviation	Privilege
`RA`	Read admin file
`WA`	Write admin file
`CR`	Create entry
`SPC`	Special Create
`ST`	Stop scheduler
`Cdft`	Change default modes
`UG`	Combine user and group permissions
`UO`	Combine user and other permissions
`GO`	Combine group and other permissions

The standard screen command keys, as described in the section on program gbch-q, plus the following context specific key commands are available:

Key	**Function**
`$`	Set and save program options
`d`	Set default priority
`l`	Set lower limit of priority
`u`	Set upper limit of priority
`m`	Set maximum load level

Key	Function
`t`	Set total load level
`s`	Set special create load level
`b`	Display charge
`p`	Set privileges
`c`	Set default access modes
`a`	Copy defaults (priorities and load levels) to current user
`A`	Copy defaults to all users
`D`	Set system default default priority
`L`	Set system default lower limit of priority
`U`	Set system default upper limit of priority
`M`	Set system default maximum load level
`T`	Set system default total load level
`S`	Set system default special create load level
`P`	Set system default privileges
`C`	Set system default default access modes

9.2.3.1 Setting user priorities

The three parameters default, min and max priority may be individually set for each user by typing `d`, `l` and `u` respectively.

The default priority will be assigned to each new job unless the user overrides it. When specifying a priority the user may not create jobs with, or change jobs to a lower priority than the minimum or a higher one than the maximum.

It is normally the case that

`min <= default <= max`

for each user, but two other possibilities are useful:

- If *`default < min`*, or *`default > max`*, then the user must specify the priority each time a job is submitted (i.e. there is no default for the user).
- If *`min > max`*, then the user is prevented from submitting jobs. This can also be achieved by turning off *create* privilege (not to be confused with *special create* privilege) for the user.

To edit these fields, move the cursor to the relevant user and type the appropriate key. Then just type in the new number and press ENTER. Type *ESC* to abort.

9.2.3.2 Setting default priorities

The three system default priority parameters default, min and max priority may be set by typing `D`, `L` and `U` respectively.

9.2.3.3 Setting user load levels

The three parameters `maxll`, `totll` and `specll` load levels may be individually set for each user by typing `m`, `t` and `s` respectively.

The maximum load level, `maxll`, restricts the load level of any one job submitted by that user to be less than or equal to the given value. The load level of the command interpreter is compared against this value.

The total load level, `totll`, restricts the total of load levels of running jobs for the given user to the specified value. If a job would cause this value to be exceeded it will not be run until some other job for the user has completed.

The special create load level, `specll`, is the default load level for any command interpreters created by the given user. This parameter is ignored for users without the *special create* privilege.

9.2.3.4 Setting default load levels

The three system default parameters `Maxll`, `Totll` and `Specll` load levels may be set by typing `M`, `T` and `S` respectively.

9.2.3.5 Applying default settings to one or all users

Each new user added to the password file inherits the values of the default settings. An individual user can be set to the default values by selecting their entry and entering a lower case `a`. To apply the defaults to all users enter an upper case `A`.

9.2.3.6 Displaying charge

To display a user's charge field, press `b`. A one-line popup display shows the charge for the current user. gbch-user no longer provides facilities to change the charge.

9.2.3.7 Setting user's privileges

To edit a user's privileges, press `p`, which will open a screen looking like this:

```
Privileges for user tony group staff

User tony may Read admin file: No
User tony may Write admin file: No
User tony may Create entry: Yes
User tony may Special create: No
User tony may Stop scheduler: No
User tony may Change default modes: Yes
User tony may Combine user/group perms: No
User tony may Combine user/other perms: No
User tony may Combine group/other perms: No
```

The standard screen command keys, as described in the section on program gbch-q, earlier plus the following context specific key commands are available:

Key	**Operation**
ENTER	Move down, quit back to main screen if on last line
`Y T`	Set corresponding privilege
`N F`	Unset corresponding privilege
! ~	Invert privilege

Note that some privileges, where it does not make sense to have one without the other, are coupled together. For example if turning on *Write Admin File* privilege, causes the *Read Admin File* privilege to be turned on at the same time.

Note that if a user inadvertently turns off their own *Write Admin File* privilege, he/she will not get a warning, however the change will not take effect until he/she exits from gbch-user.

9.2.3.8 Setting default privileges

To edit the default privileges, press `P`, which opens a screen identical to the user privileges screen except for the title:

```
Default privileges

Default is to Read admin file: No
Default is to Write admin file: No
Default is to Create entry: Yes
Default is to Special create: No
Default is to Stop scheduler: No
Default is to Change default modes: Yes
Default is to Combine user/group perms: No
Default is to Combine user/other perms: No
Default is to Combine group/other perms: No
```

These may then be changed in just the same way as for a user's privileges. At the end, if there are any changes, gbch-user prompts with the question:

```
Copy to everyone else (but you)?
```

Reply `Y` only if copying the default privileges to all other users is required. Otherwise the default privileges will only be applied to new users.

9.2.3.9 Setting default and user modes

To edit the default job and variable creation modes (access permissions) for any given user press `c`, or to edit the default modes press `C`.

For any given user, the display and editing is identical to that for that user entering gbch-user with the *mode edit* option `-m`. For the default modes, the display is identical except for the tile.

For example, editing modes for user `tony` will look like this:

```
Modes for user tony

                       Jobs           Vars
                       U   G   O    U   G   O
```

```
Read                   Yes Yes No   Yes Yes No
Write                  Yes No  No   Yes No  No
Reveal                 Yes Yes Yes  Yes Yes Yes
Display mode           Yes Yes Yes  Yes Yes Yes
Set mode               Yes No  No   Yes No  No
Assume ownership       No  No  No   No  No  No
Assume group ownership No  No  No   No  No  No
Give away owner        Yes No  No   Yes No  No
Give away group        Yes Yes No   Yes Yes No
Delete                 Yes No  No   Yes No  No
Kill (jobs only)       Yes No  No
```

The standard screen command keys, as described in the section on program gbch-q, plus the following context specific key commands are available:

Command	**Meaning**
B	Beginning Row
E	End Row
J	Jobs column
V	Variables column
Y T	Set corresponding permission, move right
N F	Unset corresponding permission, move right
! ~	Invert permission and move right

Note that some permissions, where it does not make sense to have one without the other, are coupled together. For example if you turn on *read* permission, the *reveal* permission will be turned on at the same time.

At the end of changing the default modes, but not an individual user's modes, gbch-user will prompt with the question:

```
Copy to everyone else (but you)?
```

Reply Y only if copying the default modes to all other users is required. Otherwise the default modes will only be applied to new users. If you do want the new default modes to apply to you, move to your user name and type a.

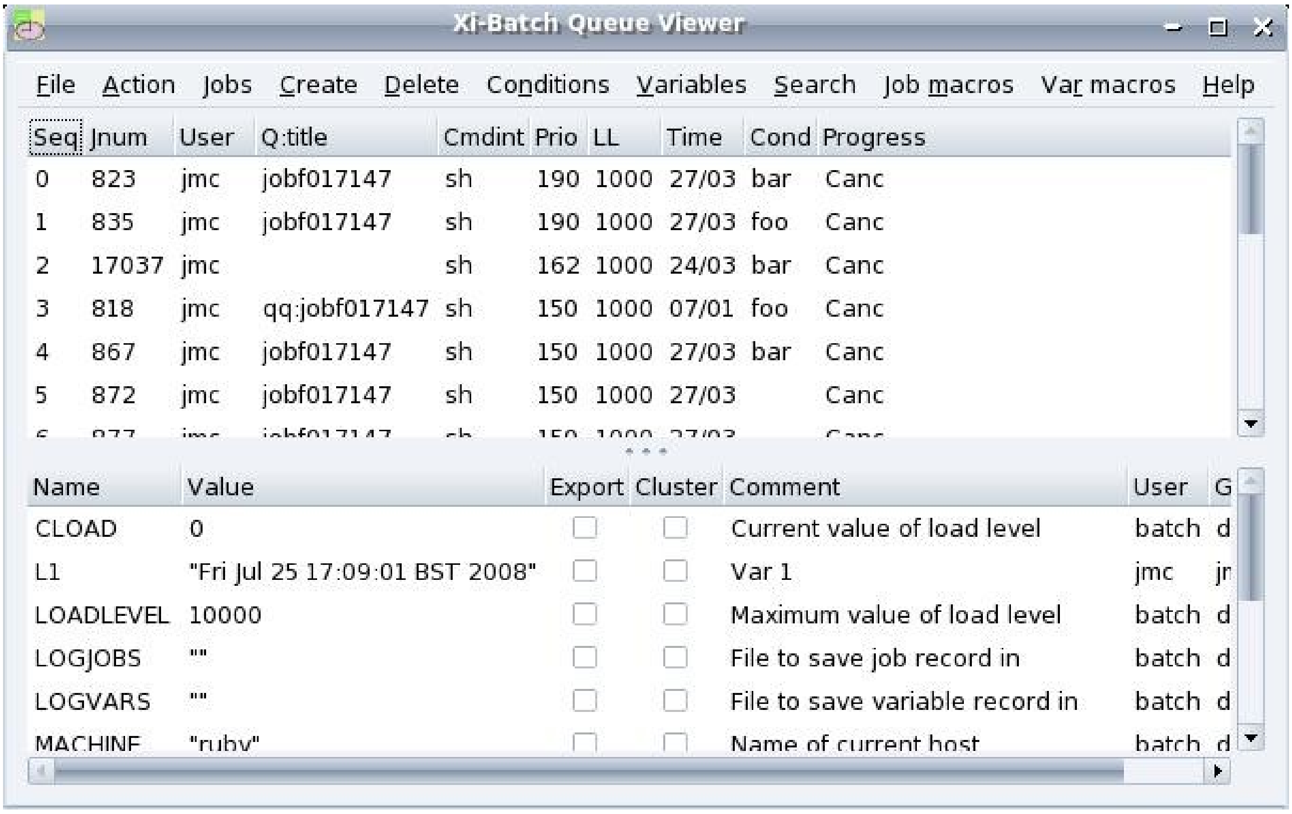
10 X Programs

This section describes the dialogs in the GTK programs gbch-xq, gbch-xr and gbch-xuser, and the Motif programs gbch-xmq, gbch-xmr and gbch-xmuser, which are only briefly described in the user program list in Chapter 8, together with gbch-xfilemon and gbch-xfilemon, the interfaces to the file monitoring tool.

Motif is not supported by GNU and the GTK programs are to be preferred as it is much easier to use, especially for defining macros.

10.1 gbch-xq – GTK GUI Batch Queue Tool

The main screen of gbch-xq is as follows:



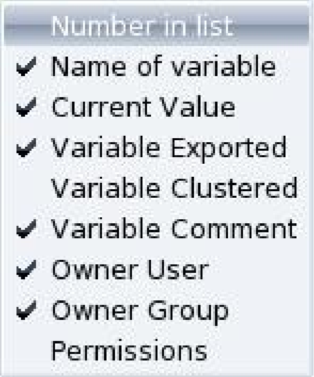
Seq	Jnum	User	Q:title	Cmdint	Prio	LL	Time	Cond	Progress
0	823	jmc	jobf017147	sh	190	1000	27/03	bar	Canc
1	835	jmc	jobf017147	sh	190	1000	27/03	foo	Canc
2	17037	jmc		sh	162	1000	24/03	bar	Canc
3	818	jmc	qq:jobf017147	sh	150	1000	07/01	foo	Canc
4	867	jmc	jobf017147	sh	150	1000	27/03	bar	Canc
5	872	jmc	jobf017147	sh	150	1000	27/03		Canc

Name	Value	Export	Cluster	Comment	User	G
CLOAD	0	☐	☐	Current value of load level	batch	d
L1	"Fri Jul 25 17:09:01 BST 2008"	☐	☐	Var 1	jmc	jr
LOADLEVEL	10000	☐	☐	Maximum value of load level	batch	d
LOGJOBS	""	☐	☐	File to save job record in	batch	d
LOGVARS	""	☐	☐	File to save variable record in	batch	d
MACHINE	"ruby"	☐	☐	Name of current host	batch	d

The top half of the display shows the jobs and the bottom half shows the variables. The division may be moved up and down and the display moved as required.

Right-click popup menus are available to invoke common operations on jobs or variables.

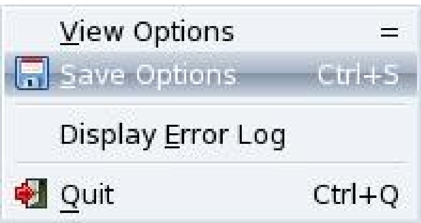
To select different fields of jobs or variables to be displayed, right-click on the title bars to give a popup menu as follows:


Number in list
✓ Name of variable
✓ Current Value
✓ Variable Exported
Variable Clustered
✓ Variable Comment
✓ Owner User
✓ Owner Group
Permissions

Select an item to have that displayed. New fields are added at the right, so the way to reorder the required fields is to remove them and then add them again in the required order.

When options are saved from the file menu, the selected fields and the order of display are saved along with the view options and macros.

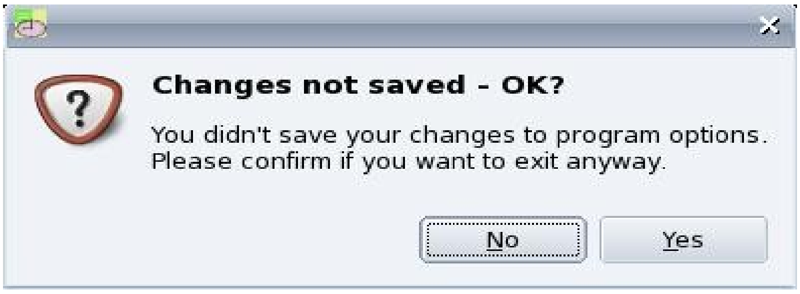
10.1.1 File Menu


View Options =
Save Options Ctrl+S
Display Error Log
Quit Ctrl+Q

The file menu provides options to reset the display, to save the options (and also list of job and variable macros, and the selected job and variable fields) to file, to display the system error log and to quit.

Standard keyboard accelerators are provided to save and quit.

Quit queries first if options have not been saved thus:

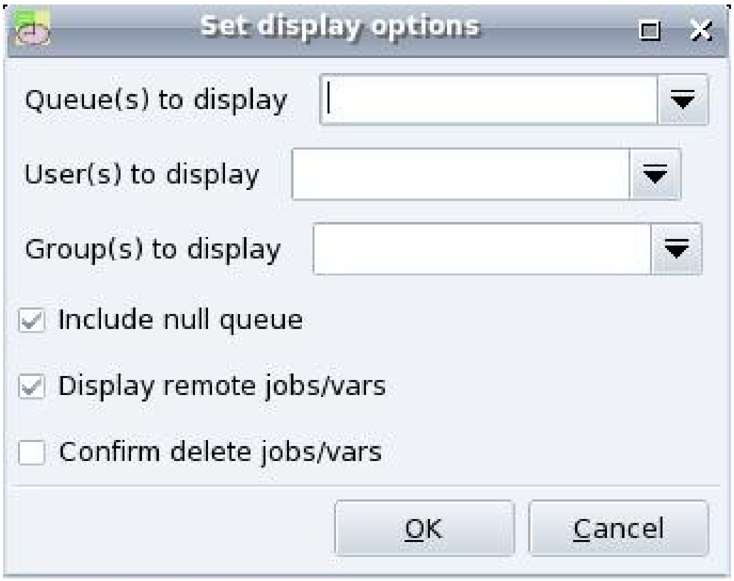


Changes not saved - OK?

You didn't save your changes to program options.
Please confirm if you want to exit anyway.

10.1.1.1 View options menu

This dialog allows the user to select which parts of the display to restrict attention to.


Set display options
OK
Cancel

Queue(s) to display

User(s) to display

Group(s) to display

☑ Include null queue

☑ Display remote jobs/vars

☐ Confirm delete jobs/vars

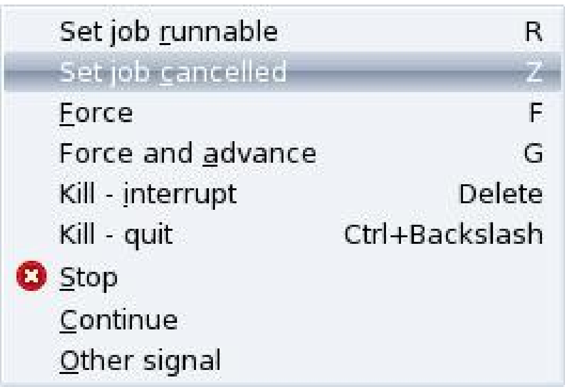
Each of the three combo boxes allows a “glob”-like pattern of job queues, users or groups to be selected to appropriately restrict the display – or a single queue, user or group selected from the drop-down.

“Include null queue” indicates whether jobs with no queue prefix are also displayed if the job view is restricted by queue name.

“Display remote jobs/vars” selects whether exported jobs or variables on other machines are to be displayed.

“Confirm delete jobs/vars” selects whether confirmation is requested before deleting.

10.1.2 Action menu

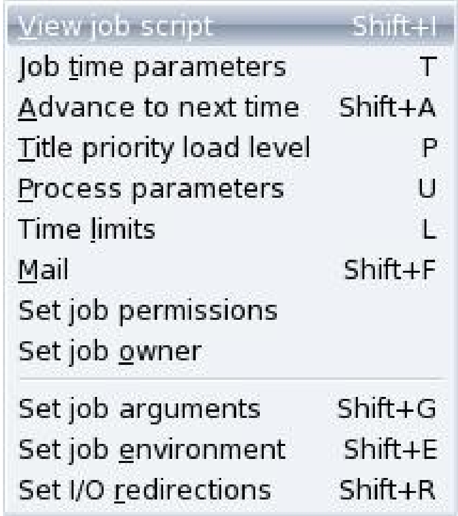

Set job runnable R
Set job cancelled Z
Force F
Force and advance G
Kill - interrupt Delete
Kill - quit Ctrl+Backslash
Stop
Continue
Other signal

The action menu provides for “one-off” actions to be performed on the currently-selected job.

An error will be displayed if no job is selected or the action is inappropriate for the state of the job.

“Other signal” will pop-up a menu showing the possible signals to send to a running job.

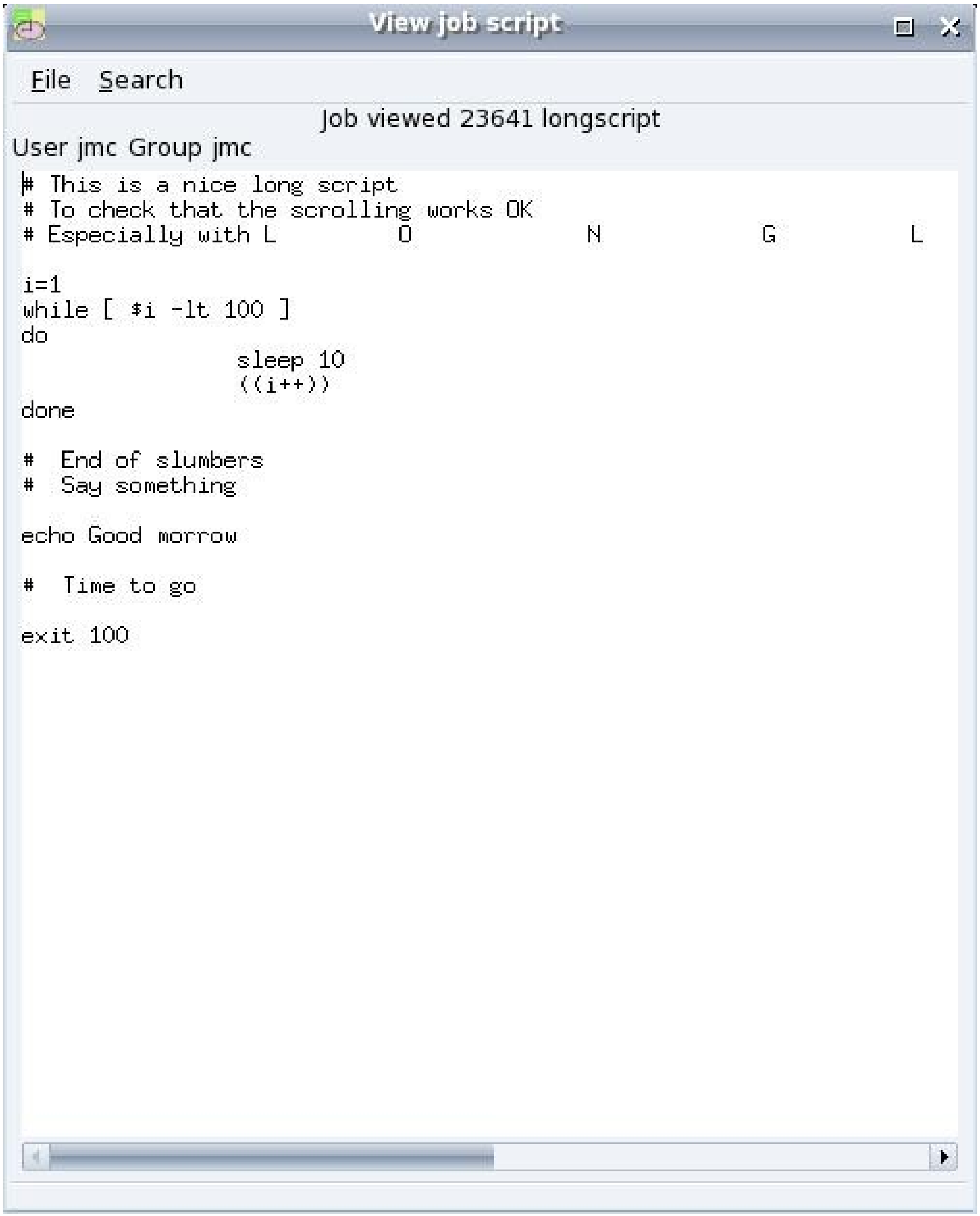
10.1.3 Job Menu


View job script Shift+I
Job time parameters T
Advance to next time Shift+A
Title priority load level P
Process parameters U
Time limits L
Mail Shift+F
Set job permissions
Set job owner
Set job arguments Shift+G
Set job environment Shift+E
Set I/O redirections Shift+R

The job menu allows a series of dialogs (some are linked together on a notebook) to be selected to set various job options.

In all cases a job should be selected from the list; if the job is inaccessible for the given user, then an error message will be displayed.

10.1.3.1 View job script



Job viewed 23641 longscript

User jmc Group jmc

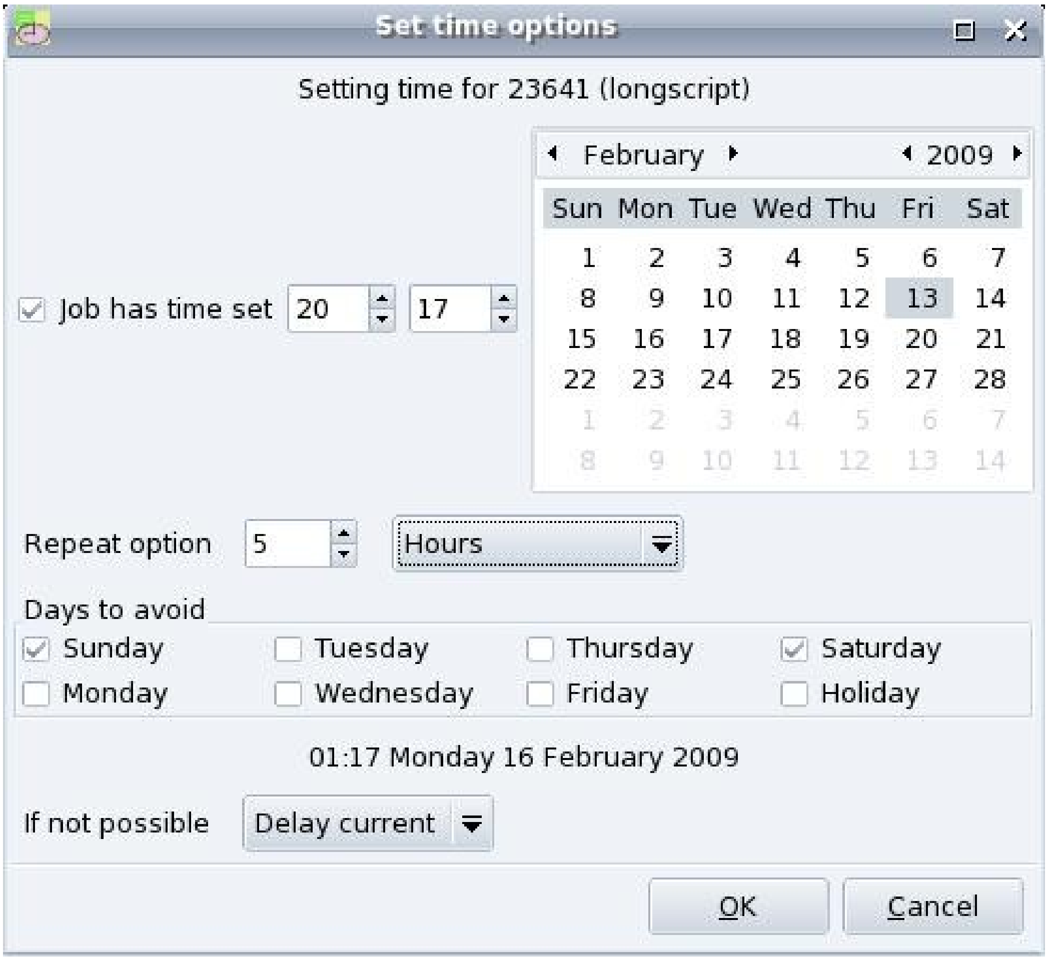
```
# This is a nice long script
# To check that the scrolling works OK
# Especially with L         O              N             G           L

i=1
while [ $i -lt 100 ]
do
                sleep 10
                ((i++))
done

#  End of slumbers
#  Say something

echo Good morrow

#  Time to go

exit 100
```

This option selects the display of the script in a fixed font allowing the user to scroll through the script.

Any number of job scripts may be simultaneously displayed.

10.1.3.2 Set job times

The following dialog box allows the attributes of job times to be displayed and changed:


Set time options
February
2009
Sun Mon Tue Wed Thu Fri Sat
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
1 2 3 4 5 6 7
8 9 10 11 12 13 14
OK
Cancel

Setting time for 23641 (longscript)

☑ Job has time set 20 17

Repeat option 5 Hours

Days to avoid

☑ Sunday ☐ Tuesday ☐ Thursday ☑ Saturday

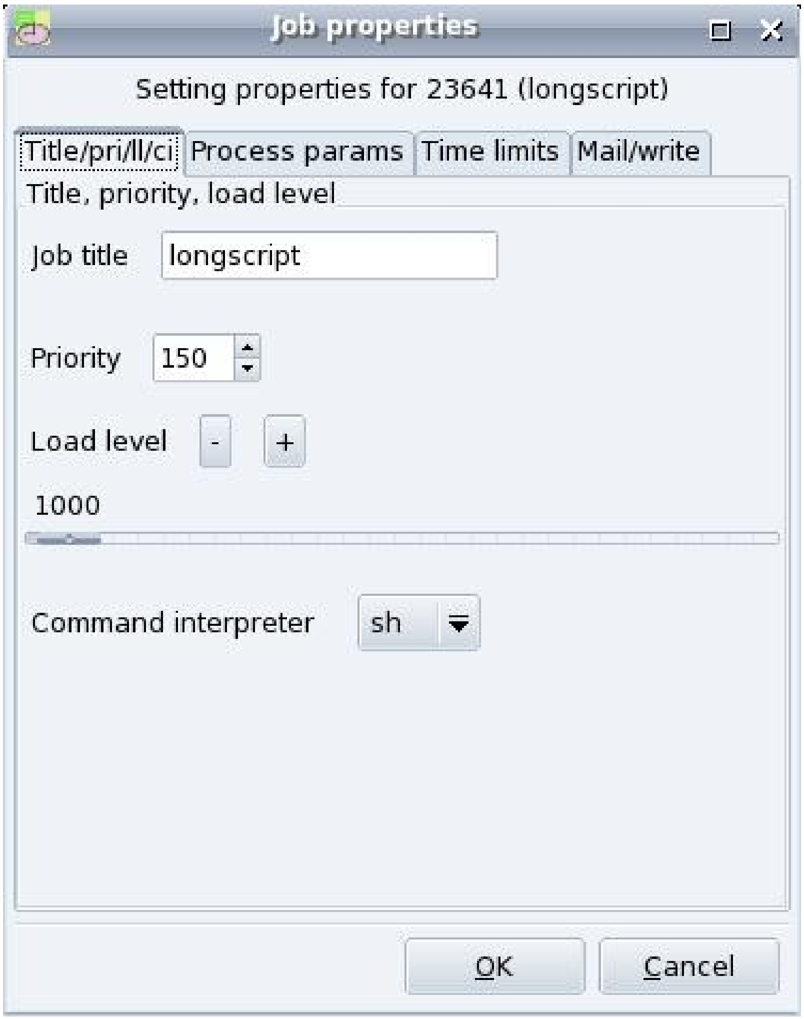
☐ Monday ☐ Wednesday ☐ Friday ☐ Holiday

01:17 Monday 16 February 2009

If not possible Delay current

Fields irrelevant to the selected repeat option are hidden from view.

10.1.3.3 Title / Priority Load Level


Job properties
Title/pri/ll/ci
Process params
Time limits
Mail/write
Title, priority, load level
-
+
OK
Cancel

Setting properties for 23641 (longscript)

Job title: longscript

Priority: 150

Load level: 1000

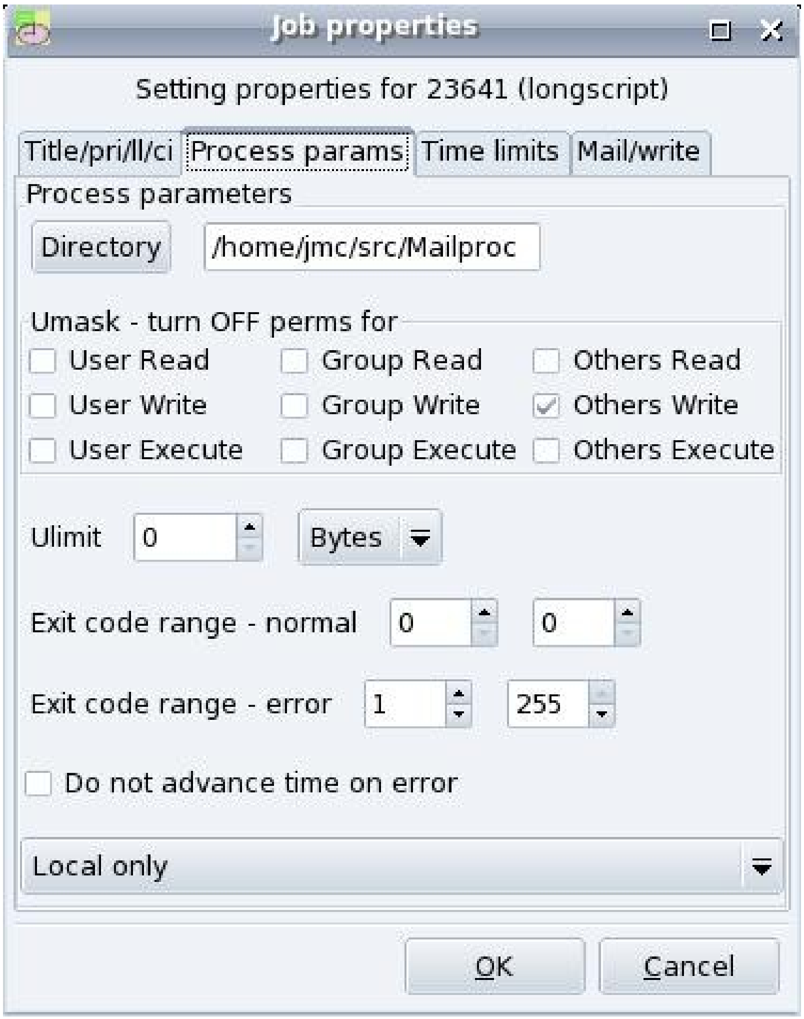
Command interpreter: sh

This is one of four dialogs of job parameters held in a “notebook”.

If a different command interpreter is selected, the load level is reset to that for the selected interpreter.

If the load level is changed by hand using the slider, the “-” and “+” buttons may be used to adjust the value by rounding to the nearest multiple of 10, 100 or 1000 successively.

10.1.3.4 Process parameters


Job properties
Title/pri/ll/ci
Process params
Time limits
Mail/write
OK
Cancel

Setting properties for 23641 (longscript)

Process parameters

Directory /home/jmc/src/Mailproc

Umask - turn OFF perms for

☐ User Read ☐ Group Read ☐ Others Read

☐ User Write ☐ Group Write ☑ Others Write

☐ User Execute ☐ Group Execute ☐ Others Execute

Ulimit 0 Bytes

Exit code range - normal 0 0

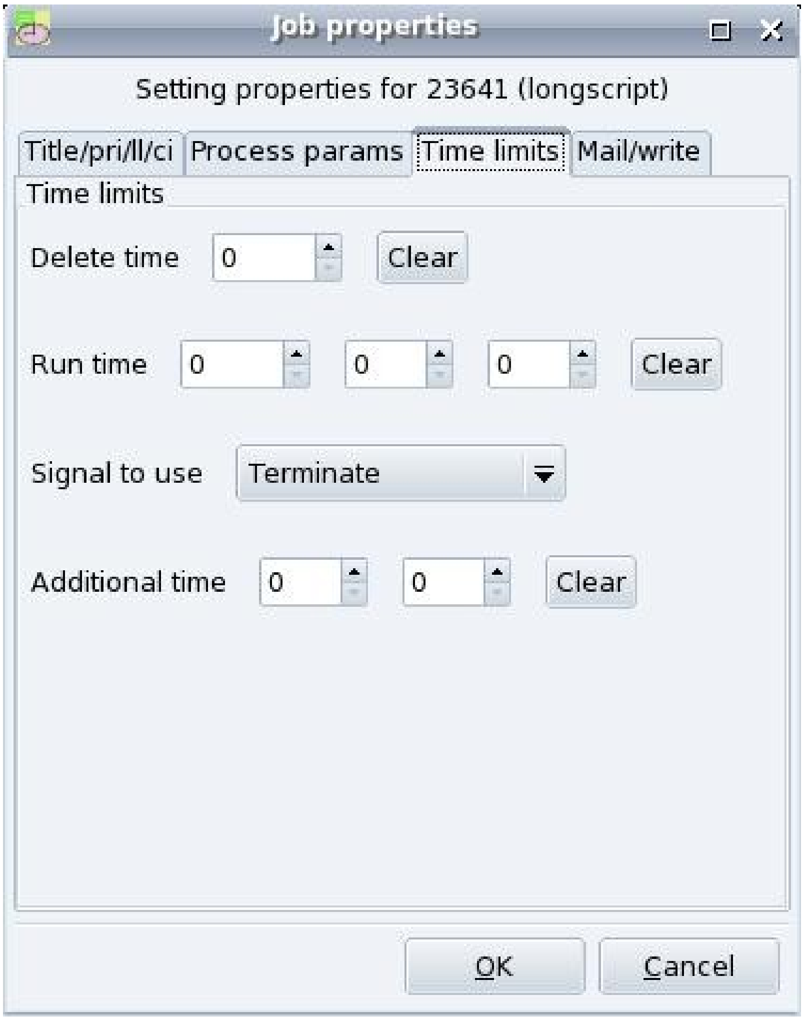
Exit code range - error 1 255

☐ Do not advance time on error

Local only

This is a second page in the notebook of job properties.

10.1.3.5 Time Limits


Job properties
Title/pri/ll/ci
Process params
Time limits
Mail/write
Clear
Clear
Clear
OK
Cancel

Setting properties for 23641 (longscript)

Time limits

Delete time 0

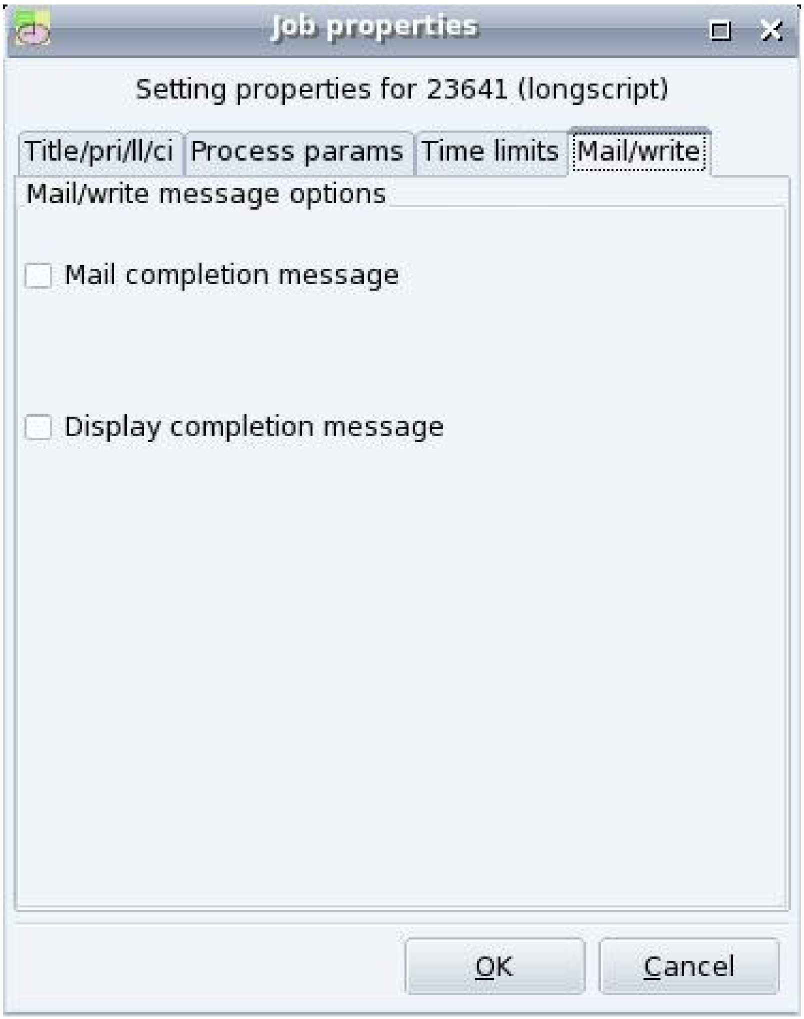
Run time 0 0 0

Signal to use Terminate

Additional time 0 0

This is a third page in the notebook of job properties

10.1.3.6 Mail / Write



Job properties
Setting properties for 23641 (longscript)
Title/pri/ll/ci
Process params
Time limits
Mail/write
Mail/write message options

☐ Mail completion message

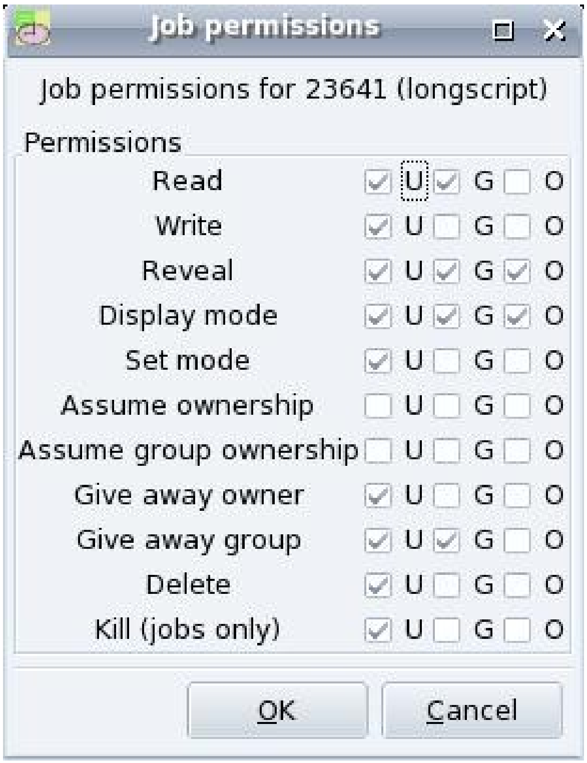
☐ Display completion message

OK
Cancel

This is the forth page in the notebook of job properties.

10.1.3.7 Job Permissions

This page allows the permissions to be selected and (if the user is allowed) changed.



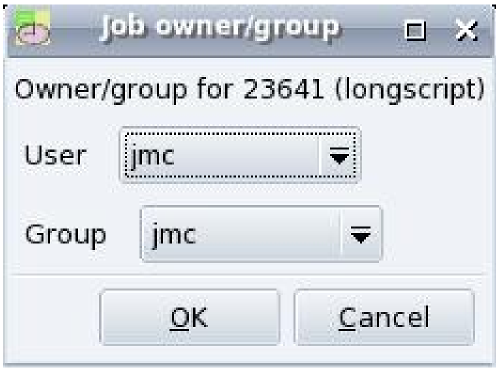
Job permissions for 23641 (longscript)

Permissions

Read	☑ U	☑ G	☐ O
Write	☑ U	☐ G	☐ O
Reveal	☑ U	☑ G	☑ O
Display mode	☑ U	☑ G	☑ O
Set mode	☑ U	☐ G	☐ O
Assume ownership	☐ U	☐ G	☐ O
Assume group ownership	☐ U	☐ G	☐ O
Give away owner	☑ U	☐ G	☐ O
Give away group	☑ U	☑ G	☐ O
Delete	☑ U	☐ G	☐ O
Kill (jobs only)	☑ U	☐ G	☐ O

Note that some permissions are “linked” so that setting one will imply others, for example “read” is implied by “write”.

10.1.3.8 Job Owner and Group



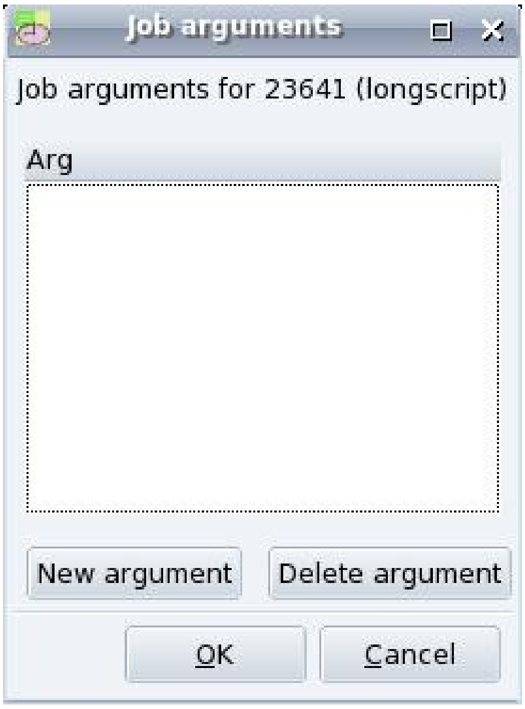
Owner/group for 23641 (longscript)

User jmc

Group jmc

This allows the owner and group to be changed (if permissible) using the drop-down to select the required owner and group.

10.1.3.9 Job Arguments



Job arguments for 23641 (longscript)

The job arguments dialog allows the list of job arguments to be reviewed.

Click “New argument” to add a new argument using a text entry box.

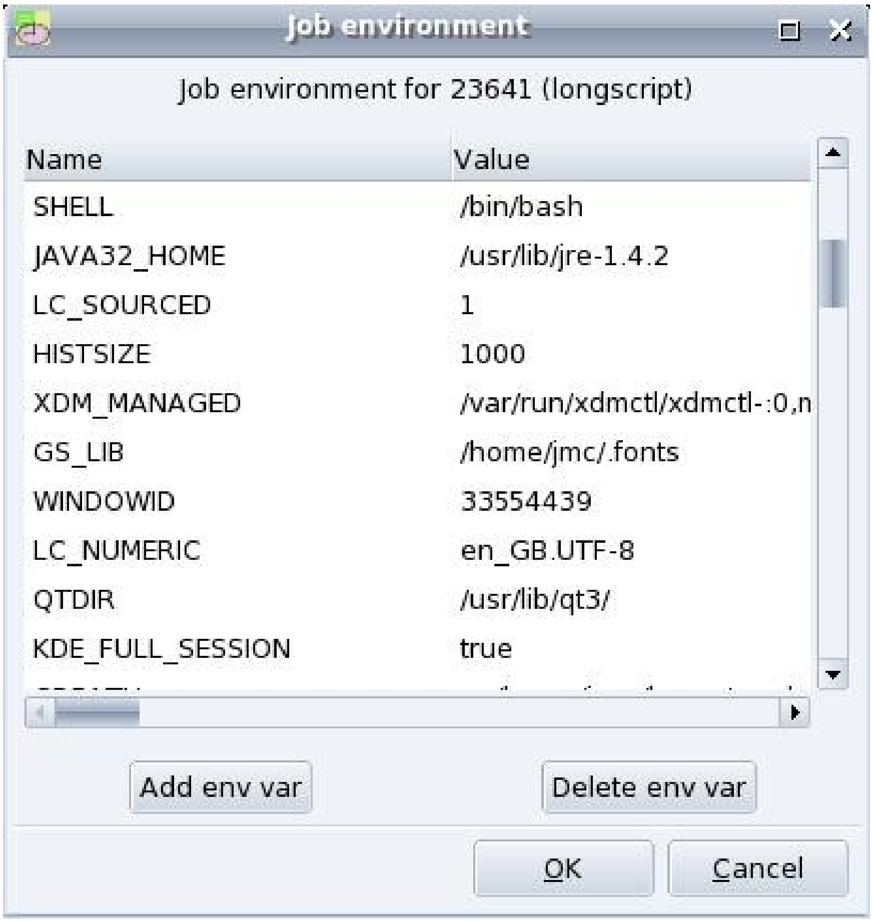
The new argument is added at the end.

To move arguments, drag and drop the arguments to reach the desired order.

Click “Delete argument” to delete an argument.

Arguments are edited in place – click on the text to allow it to be changed.

10.1.3.10 Environment variables



Job environment for 23641 (longscript)

Name	Value
SHELL	/bin/bash
JAVA32_HOME	/usr/lib/jre-1.4.2
LC_SOURCED	1
HISTSIZE	1000
XDM_MANAGED	/var/run/xdmctl/xdmctl-:0,n
GS_LIB	/home/jmc/.fonts
WINDOWID	33554439
LC_NUMERIC	en_GB.UTF-8
QTDIR	/usr/lib/qt3/
KDE_FULL_SESSION	true

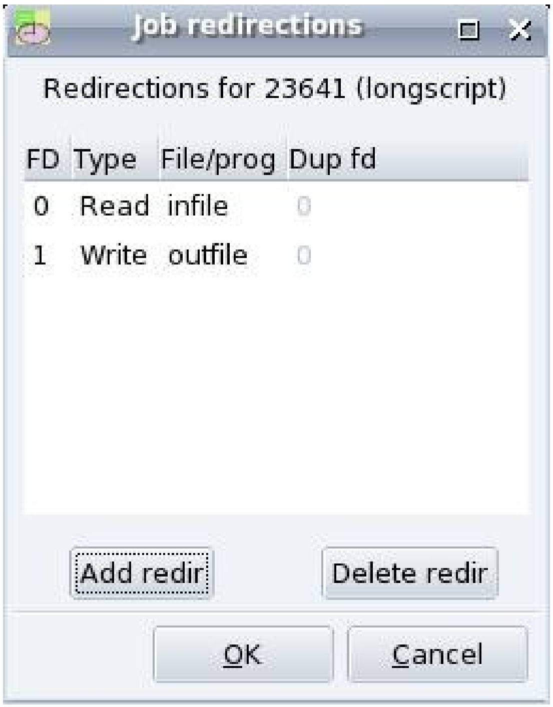
The environment variable dialog allows the user to review and edit the environment variables declared with the job.

Both the name and the value may be edited by clicking on the field.

Add a new variable by clicking on the button. The new variable is added at the end.

Variables may be moved around by “drag and drop” but this is not supposed to matter.

10.1.3.11 Redirections


Job redirections
Add redir
Delete redir
OK
Cancel

Redirections for 23641 (longscript)

FD	Type	File/prog	Dup fd
0	Read	infile	0
1	Write	outfile	0

The redirections dialog allows the list of redirections to be reviewed and edited.

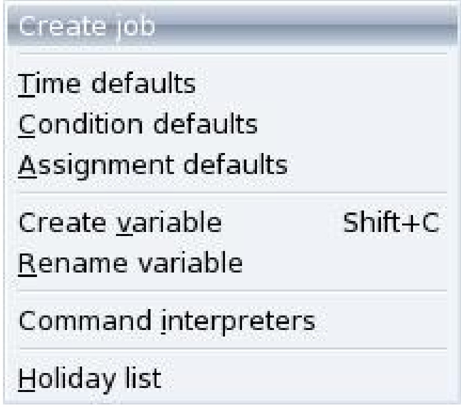
Each field may be separately edited by clicking as appropriate.

The file descriptor is a “spin box”.

Depending on the type of redirection, either the file name or the “dup fd” field is active, or neither if “close” is selected.

As with the arguments and environment variables, the order of the redirections may be changed by dragging and dropping.

10.1.4 Create menu

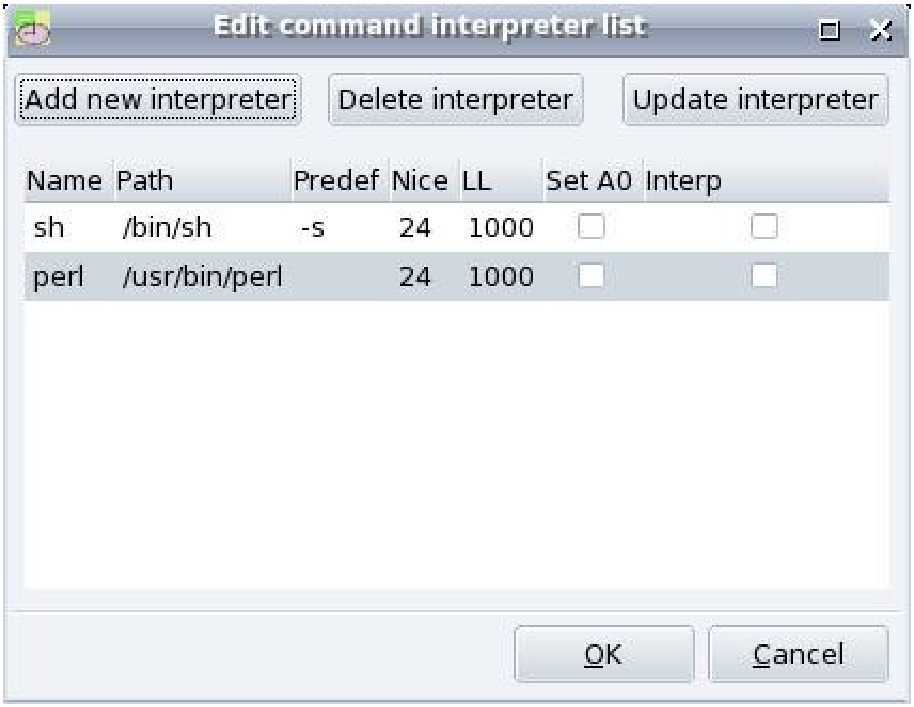

Create job
Time defaults
Condition defaults
Assignment defaults
Create variable Shift+C
Rename variable
Command interpreters
Holiday list

The create menu selects various options for creating and renaming jobs and variables, and also for setting defaults for subsequent selection of new job times, conditions and assignments.

Create job invokes a copy of gbch-xr to create jobs.

Command interpreters and holiday list give further dialogs.

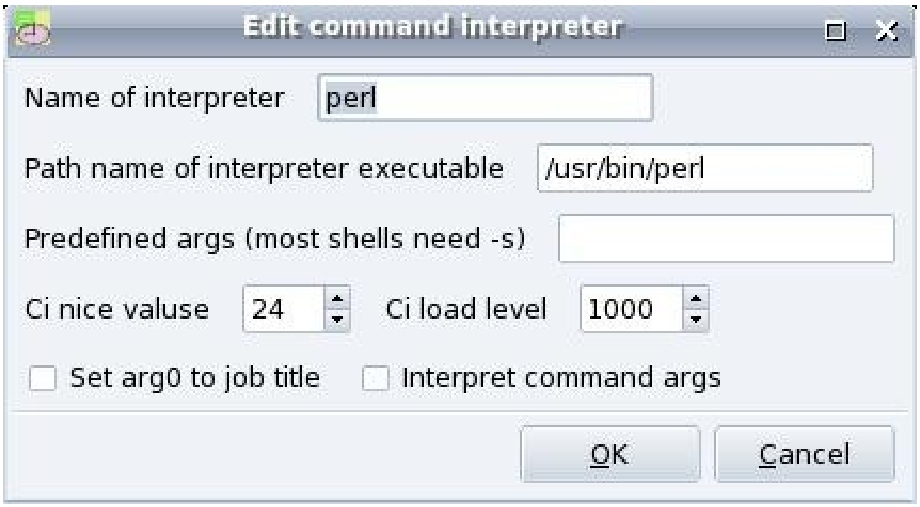
10.1.4.1 Command Interpreters


Edit command interpreter list
Add new interpreter
Delete interpreter
Update interpreter
OK
Cancel

Name	Path	Predef	Nice	LL	Set A0	Interp
sh	/bin/sh	-s	24	1000	☐	☐
perl	/usr/bin/perl		24	1000	☐	☐

The command interpreter list is displayed as shown with options as shown.

An interpreter can be edited by double-clicking to give:


Edit command interpreter
OK
Cancel

Name of interpreter: perl

Path name of interpreter executable: /usr/bin/perl

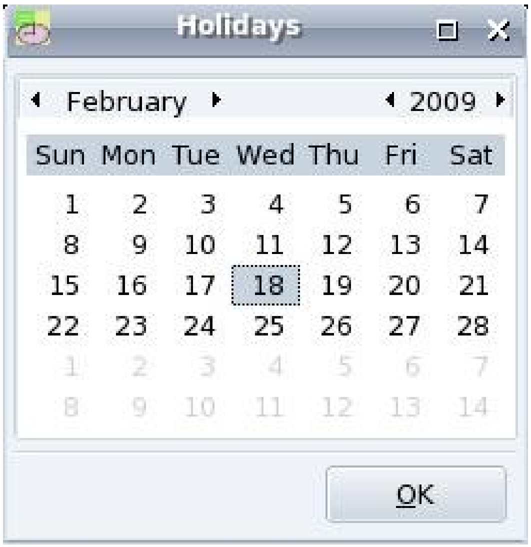
Predefined args (most shells need -s):

Ci nice valuse: 24 Ci load level: 1000

☐ Set arg0 to job title ☐ Interpret command args

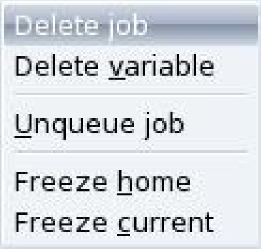
10.1.4.2 Holidays

The holidays file can be edited using the menu select which displays just a calendar.


Holidays
February
2009
Sun Mon Tue Wed Thu Fri Sat
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
1 2 3 4 5 6 7
8 9 10 11 12 13 14
OK

Holidays are displayed in bold – to toggle a holiday on or off double-click on the appropriate day.

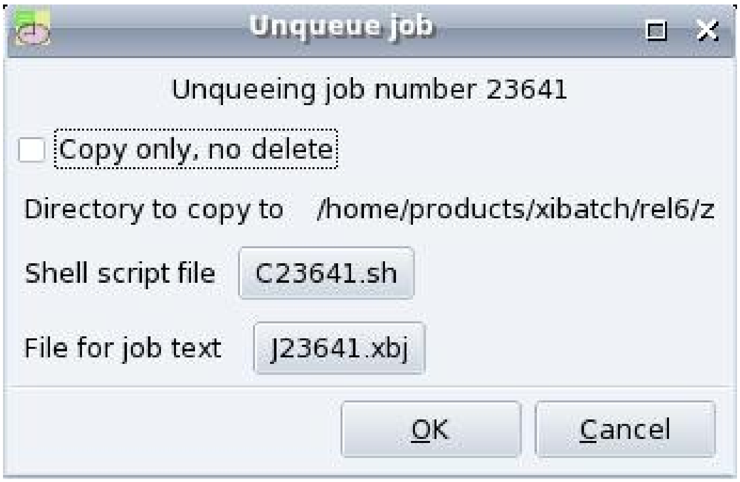
10.1.5 Delete menu


Delete job
Delete variable
Unqueue job
Freeze home
Freeze current

The delete menu provides the options shown.

The “freeze” options copy the options out of the currently-selected job into the saved options in the home or current directory to be picked up by gbch-r or gbch-xr.

The unqueue option displays the following:


Unqueue job

Unqueeing job number 23641

☐ Copy only, no delete

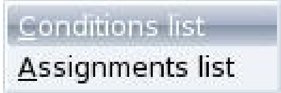
Directory to copy to /home/products/xibatch/rel6/z

Shell script file C23641.sh

File for job text J23641.xbj

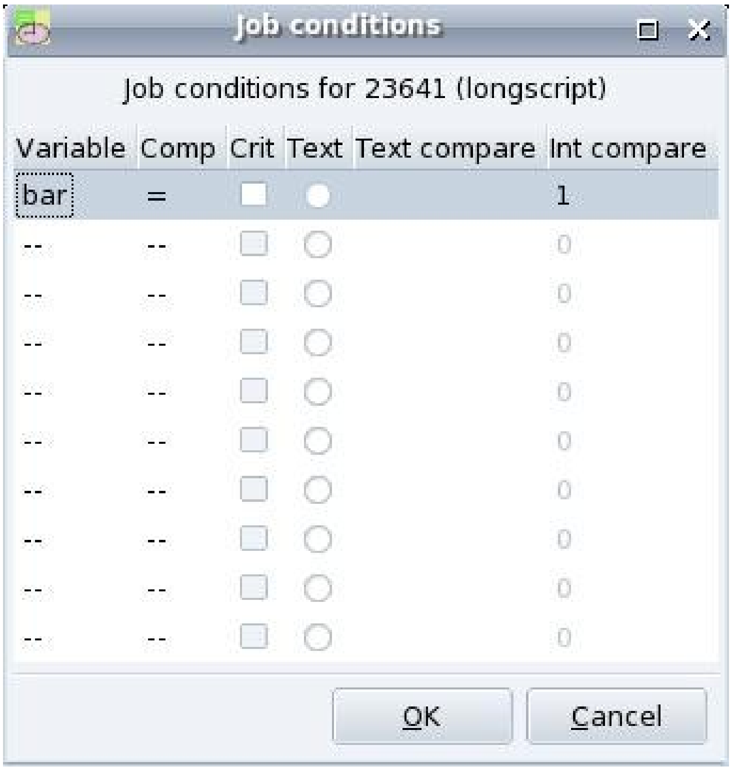
OK
Cancel

10.1.6 Conditions menu


Conditions list
Assignments list

The conditions menu enables the conditions and assignments for the selected job to be edited.

10.1.6.1 Conditions list



Job conditions for 23641 (longscript)

Variable	Comp	Crit	Text	Text compare	Int compare
bar	=	☐	○		1
--	--	☐	○		0
--	--	☐	○		0
--	--	☐	○		0
--	--	☐	○		0
--	--	☐	○		0
--	--	☐	○		0
--	--	☐	○		0
--	--	☐	○		0
--	--	☐	○		0

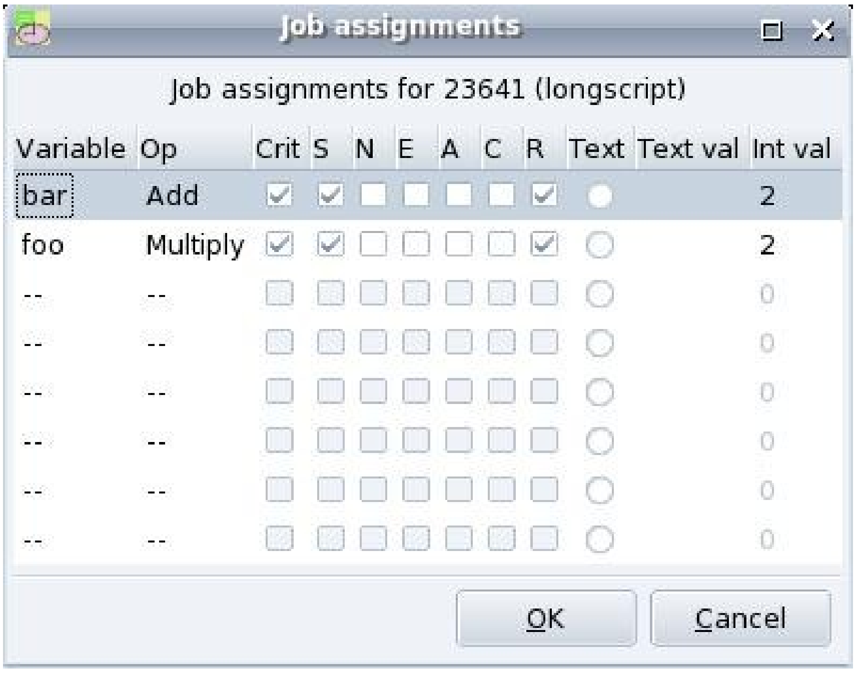
The conditions list is edited by clicking on the variable list to select the variable from a list of possible variables, the comparator and the other fields.

“Text” indicates whether the variable is to be compared as text or integer and activates the appropriate field.

To set a null condition, select “--” in the variable or the comparator field.

10.1.6.2 Assignments list

Editing the assignments list is almost identical to the conditions with the addition of the “flags” fields.

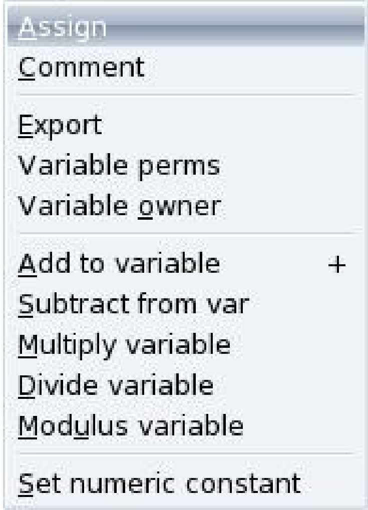

Job assignments
OK
Cancel

Job assignments for 23641 (longscript)

Variable	Op	Crit	S	N	E	A	C	R	Text	Text val	Int val
bar	Add	☑	☑	☐	☐	☐	☐	☑	○		2
foo	Multiply	☑	☑	☐	☐	☐	☐	☑	○		2
--	--	☐	☐	☐	☐	☐	☐	☐	○		0
--	--	☐	☐	☐	☐	☐	☐	☐	○		0
--	--	☐	☐	☐	☐	☐	☐	☐	○		0
--	--	☐	☐	☐	☐	☐	☐	☐	○		0
--	--	☐	☐	☐	☐	☐	☐	☐	○		0
--	--	☐	☐	☐	☐	☐	☐	☐	○		0

If no flags are set, then the assignment is taken as null.

10.1.7 Variables menu


Assign
Comment
Export
Variable perms
Variable owner
Add to variable +
Subtract from var
Multiply variable
Divide variable
Modulus variable
Set numeric constant

The variables menu performs the various possible operations on variables.

Assign assigns a specific value,

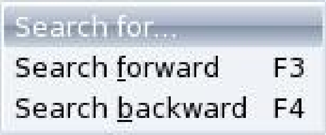
Comment sets the variable comment.

Export sets the “exported” or “clustered” flag.

Permissions and ownership is as for jobs (except no kill permission).

The various arithmetic operations are performed with the constant value given by the “set numeric constant” operation.

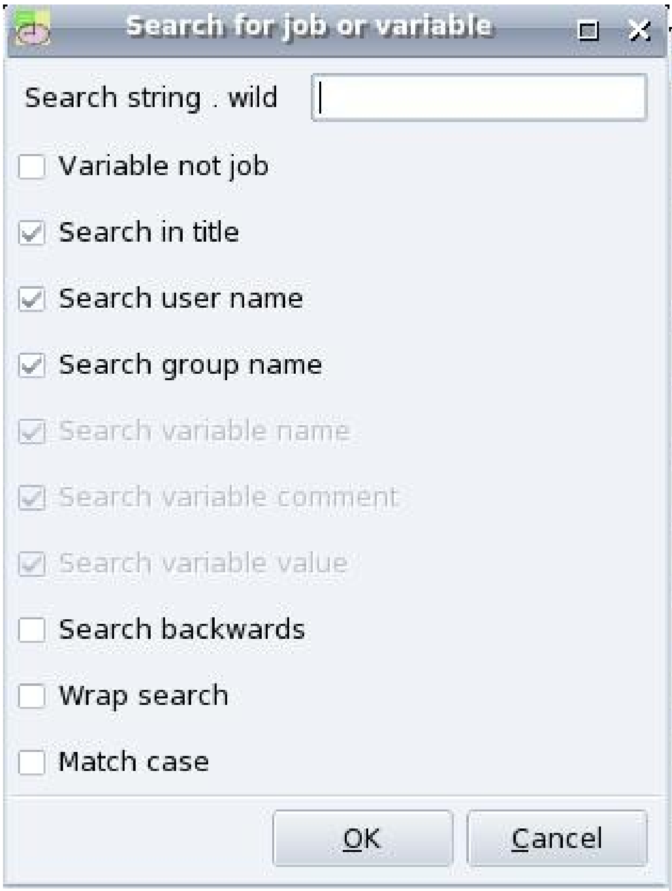
10.1.8 Search menu


Search for...
Search forward F3
Search backward F4

The search menu searches for and positions the job or variable list according to fields found in the job or variable fields.

“Search for” brings up a dialog as follows:

The first box denotes whether the string is to be found in the job or variable list. If this is selected, different fields are made active thus:


Search for job or variable

Search string . wild

☐ Variable not job

☑ Search in title

☑ Search user name

☑ Search group name

☑ Search variable name

☑ Search variable comment

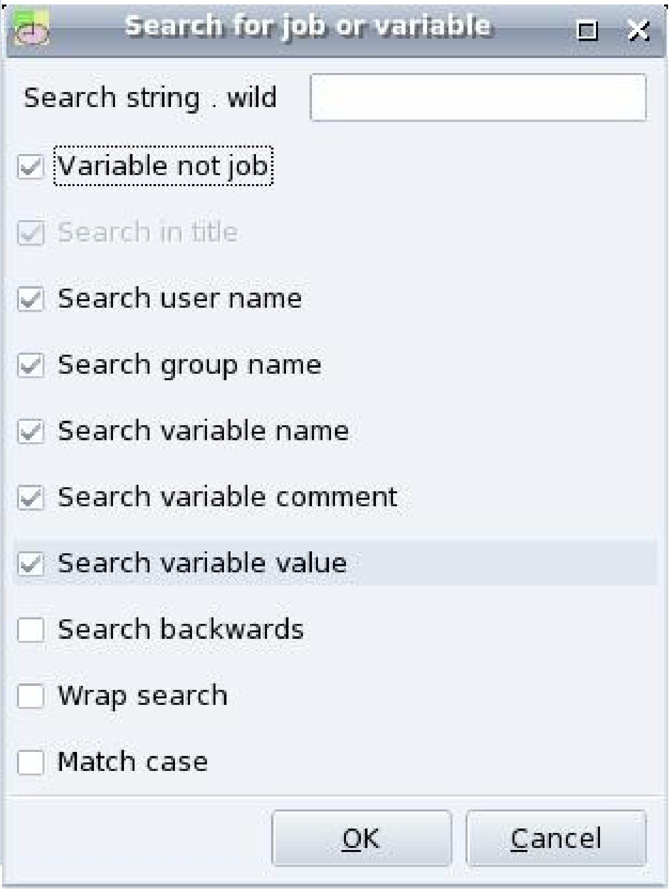
☑ Search variable value

☐ Search backwards

☐ Wrap search

☐ Match case

OK
Cancel


Search for job or variable

Search string . wild

☑ Variable not job

☑ Search in title

☑ Search user name

☑ Search group name

☑ Search variable name

☑ Search variable comment

☑ Search variable value

☐ Search backwards

☐ Wrap search

☐ Match case

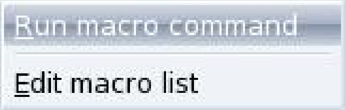
OK
Cancel

The “Search Forward” and “Search Backward” menu options repeat the last search with the other options the same.

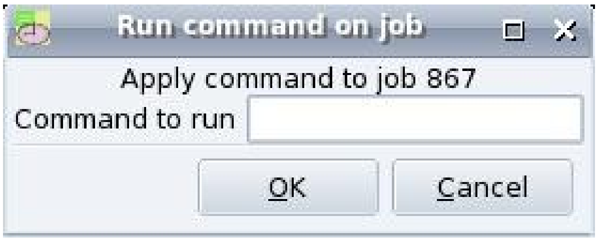
10.1.9 Macro editing

The two macro menus allow the user to specify arbitrary commands to be applied to the selected job or the selected variable. Up to 10 each commands can be saved and added to the macro menus and will be saved when the program options are saved for later use (on a per-user basis).

When first used the menus display as follows:


Run macro command
Edit macro list

To try a macro select a job and then the first option to get a prompt thus:


Run command on job
Command to run
OK
Cancel

Apply command to job 867

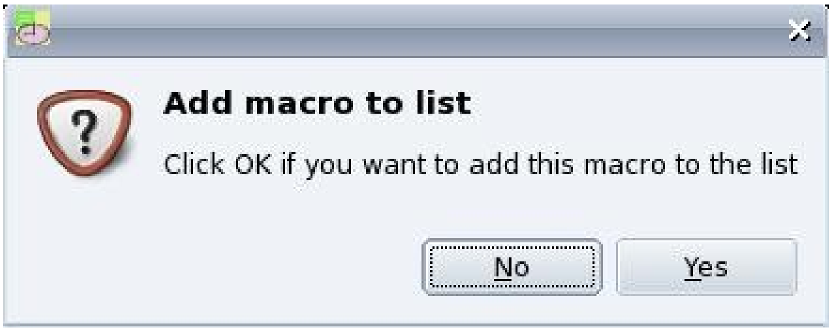
Supposing you wanted to have a macro to set the priority to 199.

You would put

```
gbch-jchange -p199
```

in the box.

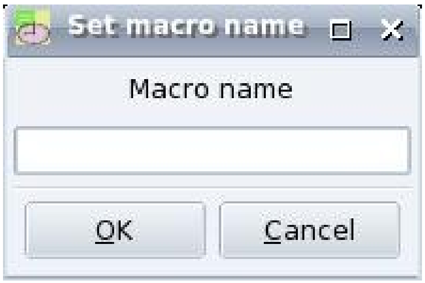
If it runs without error, then you would be prompted



Add macro to list

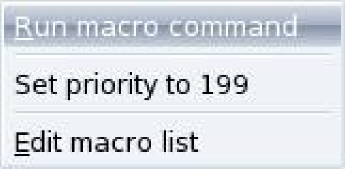
Click OK if you want to add this macro to the list

If you reply Yes you then get prompted with:


Set macro name
Macro name
OK
Cancel

This lets you enter a macro name e.g. “Set priority to 199”.

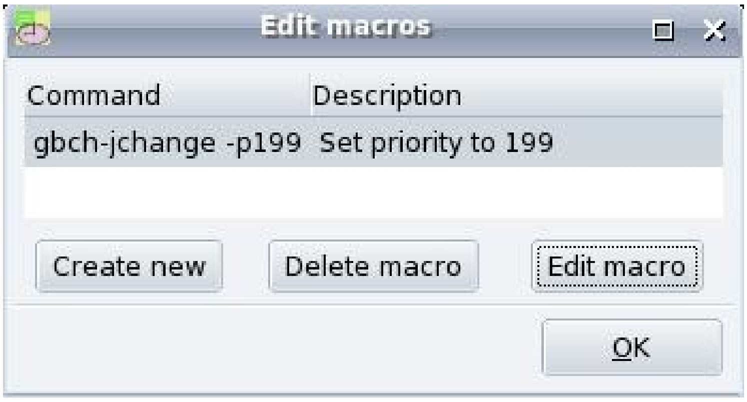
The next time you show the job macro menu it will display:


Run macro command
Set priority to 199
Edit macro list

“Set priority to 199” will apply the macro to the selected job at any time.

You can also edit the macro list with the final option in the menu and get a dialog such as:

This lets you edit the command and the description and update the menu.



Command	Description
gbch-jchange -p199	Set priority to 199

10.2 gbch-xmq - Optional Motif GUI Batch Queue Tool

gbch-xmq is a fully interactive Motif alternative to the standard batch queue manager, gbch-q. As with gbch-q the format of the screen display, the help messages and even the command keystrokes can be easily altered to suit your requirements.

Unlike gbch-q there are no command line options to gbch-xmq. The facility to change or specify resource settings for an X11 (and hence Motif) program on the command line can be used.

10.2.1 Options

The option `-xrm "str"` may be specified for str to override any resources specified in the program's resource file, `GBATCH`.

Do not worry about the terminology, it is sufficient to recognise what these resources look like. For example the resource which specified the title for the main window of gbch-xmq is:

```
gbch-xmq.title: gbch-xmq - GNUbatch Queue Viewer
```

The text before the colon "`:`" is the resource name (i.e. `gbch-xmq.title`) and the text after is the data, in this case a string. The data can be boolean, integer or a string for different resources.

10.2.2 Useful Resources

All of the resources for program gbch-xmq start with "`gbch-xmq`" To keep the initial column width to manageable proportions just the text from the "`.`" or "`*`" is listed.

`.Title`	Specifies the Title that appears on the main window of gbch-xmq.
`.IconName`	Sets the text for the icon when the copy of gbch-xmq is iconised.
`.keepJobScroll:`	This is a boolean with the values `True` or `False`. When True it stops the job list scrolling when jobs change their position in the queue. The default is `False`.

`.localOnly:`	This is a boolean with the values `True` or `False`. When True the display will be restricted to show only local jobs and variables. When `False` jobs and variables on all co-operating GNUbatch hosts will be shown. The default is `False`.
`.confirmAbort:`	This is a boolean with the values `True` or `False`. When True the program will always ask for confirmation when deleting jobs. The default is `True`.

`.onlyUser:`	Limits the displays to the jobs and variables owned by the specified user or set of users.
	The set of users may be one name, a list of names or a list of patterns for matching user names. They may be given as a comma-separated list of alternatives, including the use of shell-style wild-cards. For example
	`Fred`
	`jane,tony,john`
	`ops*,admin[1-9]`
	The wild-card options are:
	`*` Matches anything
	`?` Matches one character
	`[a-m]` Matches one character in list or range
	`[!n-z]` Matches one char not in list or range

`.onlyGroup:`	This limits the display of jobs and variables belonging in the specified primary group or set of primary groups. The set of groups is specified in exactly the same way as those for users.
`.queue:`	Limit display to jobs in the specified queue or set of job queues. The set of queues is specified in exactly the same way as for users.
`.incNull:`	This controls whether jobs in the null queue (no queue name) are included along with the specified queue or queues when restricting the display by queue. Set to `False` to exclude such jobs and `true` to include them, the default.
`*jlist.visibleItemCount:`	An integer specifying the number of lines in the job list pane of the Main Window.
`*vlist.visibleItemCount:`	An integer specifying number of lines in the variable list pane of the Main Window.

10.2.3 Examples

The display can be restricted to a particular user or set of users and the Window title changed to act as a reminder. One `-xrm` option is used to set the title and another is used to restrict the view by user:

```
gbch-xmq -xrm "gbch-xmq.title: Tony\'s Batch Jobs" \
-xrm "gbch-xmq.onlyUser: tony" &
```

In this example the user is `tony` and the Window title is "`Tony's Batch Jobs`". The command has been broken over two lines by preceding the newline character with a back slash. It is not uncommon to use several lines if specifying `-xrm` options.

10.2.4 The Main Window

When gbch-xmq is invoked the main window will be displayed. By default it will look something like this:

The main screen is divided into two key functional areas. The top area contains menus and short cut buttons for issuing commands. The bottom area displays the batch jobs and variables, which may be selected to have commands performed upon them.

The key features of the batch jobs and variables area are:

Variables and jobs can be operated upon by the appropriate menu options. The job or variable must first be selected. This is done by clicking on the line identifying it using the mouse. Once selected the line will be highlighted.

If you cannot see the variable or job that you want then you may:

- Use the scroll bar or search menu options to find it.
- Change the view to add the item or remove unwanted items from the display.

The key features of the top area are:

10.2.5 The Menus and Shortcut Buttons

All commands are performed by selecting a menu option or clicking on the equivalent shortcut button. Some of the menu options may also be selected using shortcut keys, which are indicated to the right of the relevant options in each menu.

10.2.5.1 The Options Menu

For tailoring the look and feel of gbch-xmq, saving the tailored settings, viewing the error log and quitting.

View options brings up the Display options dialog, to tailor the look and feel. Pressing the `=` key also invokes this option.

Save options creates a local copy of the View options.

Display Error log brings up a Viewer showing any messages held in the GNUbatch system log file, `btsched_reps`.

10.2.5.2 The Action Menu & Buttons

For high level actions: starting and stopping batch jobs.

Set job runnable will change a job from the Cancelled, Error or Abort state to the Ready or Run state. This option is also available using the '**Set run**' shortcut button.

Set job cancelled puts a job on held (i.e. not able to run). This option is also available using the '**Set canc**' shortcut button.

Force run sets a job runnable and overrides any time specification to allow the job to run as soon as any Variable Conditions are satisfied. This option is also available using the '**Force**' shortcut button.

Force run + advance sets a job runnable overriding any time specification to allow the job to run as soon as any Variable Conditions are satisfied. The repeat time on the job is advanced to the next repetition. This option is also available using the '**Force/adv**' shortcut button.

Kill - interrupt attempt to terminate a running job by sending it an Interrupt Signal.

This option is also available using the '**Interrupt**' shortcut button.

Kill - quit attempt to terminate a running job by sending it a Quit Signal. This option is also available using the '**Quit sig**' shortcut button.

Kill - Other signal attempts to terminate a running job by sending it a specified Signal. This option opens a selection dialog.

10.2.5.3 The Jobs Menu & Buttons

Provides options for inspecting and managing batch jobs which are currently visible in the job list.

View job opens a text browser showing the job script to be output. This option is also available using the 'View job' shortcut button.

Set job time parameters brings up a dialog for setting the start time, retention options, repetition details and list of days to avoid. This option is also available using the 'Set time' shortcut button.

Advance to next time skips the next scheduled execution of a job by advancing to the next repetition.

Title, pri, Command int, loadlev opens the dialogue for setting the Title, Priority, Command Interpreter, and load level for the job.

Process parameters brings up the dialog to select the process parameters: working directory, ulimit, umask, network scope and which exit codes represent an error.

Time limits opens the dialog for specifying time restrictions to terminate a runaway job.

Mail and write markers opens the dialog to specify what notification is required when a job finishes.

Job permissions brings up the dialog to set the access modes for the selected job.

Job Arguments opens a dialog for adding, editing and deleting arguments that are passed to the job on its command line.

Job Environment opens the dialog for adding, modifying and deleting the environment variables that are set up in the jobs run time environment.

Job I/O Redirections opens the dialog for adding, editing and deleting I/O redirections.

10.2.5.4 The Create Menu

Provides options for Creating variables, requeueing jobs and setting up defaults.

Create a job from file opens a dialog to requeue a job that was earlier unqueued and perhaps modified. The dialog requires the name of the command file of the unqueued job.

Set job time defaults sets default values for the set time dialog. These defaults are used when a job with no time, first has a time set.

Condition defaults sets up a default pre-condition for editing the pre-conditions on jobs which so far have none.

Assignment defaults sets up a default assignment for editing the assignments on jobs which so far have none.

Create Variable brings up a dialog to create a new variable, including setting up the name, initial contents and comment.

Rename Variable changes the name of a variable, including all references to it in the pre-conditions and assignments of batch jobs.

Edit command interpreter list brings up the dialog for adding, changing and deleting command interpreters.

View/edit Holidays brings up the dialog for viewing and changing the holiday calendar.

10.2.5.5 The Delete Menu

Contains options to unqueue jobs, delete jobs and deletes variables.

Delete job removes the selected job from the queue, providing it is not running. An error message is produced and the delete command ignored if the job is running or the user does not have suitable permission.

Delete variable removes the selected variable from the scheduler, providing no jobs specify it in a condition or assignment. An error message is produced and the delete command ignored if the variable can be identified as in use or the user does not have suitable permission.

Unqueue job opens the dialog to unqueue a copy of the selected job. The job may be deleted or not as required.

Copy job options as default to $HOME takes the options from the selected job and saves them in a `gbch-r` environment variable in a `.gnubatch` file in the users home directory.

Copy job options as default to Current takes the options from the selected job and saves them in a `gbch-r` environment variable in a `.gnubatch` file in the users current directory.

10.2.5.6 The Condition Menu

Provides options for setting up pre-conditions and assignments.

Set job conditions brings up the dialog to add, modify and delete pre-conditions on the selected batch job.

Set job assignments brings up the dialog to add, modify and delete assignments for the selected batch job.

10.2.5.7 The Variable Menu

Provides options for manipulating variables.

Assign variable brings up the dialog to modify the data held by the selected variable.

Assign comment brings up the dialog to modify the comment field of the selected variable.

Set variable Exported makes the variable accessible by all co-operating GNUbatch hosts.

Set variable Local restricts access to the variable to the local machine.

Set variable permissions brings up the dialog to modify the access modes for the selected variable.

Add to variable increments the value of the selected variable by the currently set Arithmetic Value.

Subtract from variable decrements the value of the selected variable by the currently set Arithmetic Value.

Multiply variable multiplies the value of the selected variable by the currently set Arithmetic Value.

Divide variable divides the value of the selected variable by the currently set Arithmetic Value.

Modulo variable performs a modulo on the value of the selected variable by the currently set Arithmetic Value.

Set arithmetic value for the above operations.

10.2.5.8 The Search Menu

Both the variable and job lists may be navigated by using search options to find items of interest.

Search for selected item or pattern.

Search forward from the current position

Search backward from the current position

10.2.5.9 The Jobmacro Menu

Provides options for running macro commands related to the selected batch job.

Run job command macro opens a dialog prompting for the name of the macro to run. This is then invoked by gbch-xmq with the job id number of the selected job.

<macro> runs the pre-defined macro, whose name or a brief description will appear in place of the <*macro* > place holder.

Up to 9 macros may be pre-defined.

10.2.5.10 The Varmacro Menu

Provides options for running macro commands related to the selected job control variable.

Run var command macro opens a dialog prompting for the name of the macro to run. This is then invoked by gbch-xmq and passed the name of the selected variable.

<macro> runs the pre-defined macro, whose name or a brief description will appear in place of the <*macro*> place holder.

Up to 9 macros may be pre-defined.

10.2.5.11 Help

Context sensitive help for using gbch-xmq.

Help displays help for the current window.

Help on changes the operating mode from taking commands to displaying help on any object (menu, button etc) that is selected.

About displays information, such as release number, about the version of gbch-xmq that is running.

10.2.6 Setting the View Options

The content and format of information displayed by gbch-xmq can be customised via the **View options** item under the Options menu. Confirmation for the delete commands may also be set under this option.

Selecting this option opens the following dialog window.

10.2.6.1 Setting the Confirmation level

By default gbch-xmq asks for confirmation before deleting any job from the queue. This may be relaxed to allow jobs to be deleted without confirmation.

10.2.6.2 Restricting the display

The display may be restricted by effectively filtering to only show information for selected users, groups, job queues and local or all GNUbatch hosts.

10.2.6.2.1 Restricting the display to the local host

All hosts running GNUbatch in the networked mode can be treated as a single system. By default gbch-xmq will show all of the externally visible jobs and variables. The view can be restricted to show just the local job queue and variables.

10.2.6.2.2 Restricting the display by job queue

The job display can be restricted by selecting a set of job queue names or patterns matching job queues.

10.2.6.2.3 Restricting the display by user & group

The display may be restricted to jobs and variables owned by a specified user or set of users. Similarly to users it may be restricted to one or more primary groups.

Sets of users or groups may contain just one name, a list of names or a list of patterns for matching names. The group and user names may be given as a comma-separated list of alternatives, including the use of shell-style wild-cards. For example

```
fred
jmc,tony,ukops_jmc,ukops_wal
ukops*,ukadmin[1-5]
[m-z]*
```

The wild-card options are:

`*`	Matches anything
`?`	Matches one character
`[a-m]`	Matches one character in list or range
`[!n-z]`	Matches one char not in list or range

10.2.6.3 Changing the fields displayed and their format.

There is far more information available for both jobs and variables than could be displayed in the main window of gbch-xmq. Different columns of information may be displayed as required. The field widths and handling of field overflow may also be adjusted.

For example: If your batch jobs often have arguments and long titles, and you do not need the shell column in the job display, then you could do the following:

- Delete the shell column.
- Make your gbch-xmq main window wider by dragging it with the mouse.
- Add a job argument column.
- Increase the Title width from 13 to say 25 characters.

To edit the job or variable displays click on the appropriate button:

The variable display uses two lines for each variable, which are edited separately, hence the separate buttons for editing each line.

10.2.6.3.1 Changing the Job Display

Clicking on the Reset Job Display Fields brings up the following window. The row of buttons at the top are for adding, changing and deleting fields or separators. The fields are the columns of information and the separators are the column dividers.

Underneath the row of buttons is a scrollable text window showing the display format. Each line holds the specification for one column or column divider, as follows:

a) Field description / title

b) Width in characters

c) Field Identifier

d) No action on field overflow

e) Overflow onto left hand field is permitted

f) Open quote before separator

g) Separator character(s)

h) Close quote after separator

To edit an existing field select the line showing the specification for that field and click on the Edit field button. To insert a new field select the line underneath the point at which you want to insert it and click on the New field button. Either of these actions will bring up a "Job display field" window looking something like this:

The width of field can be adjusted by typing in a new value or using the up and down arrow buttons. The "Use previous field..." button is to allow a field to overflow into the field on its left. Pressing the button changes it from the deselected to selected state and vice versa.

The other buttons allow selection of what information will be displayed in the column. Only one of these buttons may ever be in the selected state. When a button is selected the field width is reset to a suitable value for the data to be displayed. This value may then be adjusted as required.

10.2.6.3.2 Changing the Variable Display

The principles for the variable display are almost the same as those for the job display. The only difference is that each variable is listed on two lines, the format of which are edited separately. Similar windows are used, only the display information inside is different.

10.2.6.4 Saving the Format Changes

These are saved in local copies of the file `xmbtq.help` file which is given a user specified name at the time of saving. If a local help file has been set up, gbch-xmq will be told to use it by the `XMBTQCONF` parameter in the local `.gnubatch` file.

10.2.6.5 Saving the View Options (excluding Format Changes)

All of the view options, excluding job and variable display formats, are saved in a user's local copy of the resource file. This file is called `GBATCH` and is kept in the user's home directory. The customised options are added to the end of the file.

The view options may be overridden by entering the Motif parameters on the command line using the -xrm option for the window manager. For example to restrict the job display to just those belonging to users kate and tom give the option:

```
gbch-xmq -xrm "gbch-xmq.onlyUser: kate,tom"
```

The complete set of view options, held at the end of a user's `GBATCH` file, look like this:

```
!! gbch-xmq User-defined options 15:02:29 2001/06/10

gbch-xmq.keepJobScroll:            False
gbch-xmq.localOnly:                False
gbch-xmq.confirmAbort:             False
gbch-xmq.incNull:                  True
gbch-xmq.onlyUser:
gbch-xmq.onlyGroup:
gbch-xmq.queue:
gbch-xmq*jlist.width:              599
gbch-xmq*jlist.visibleItemCount:   10
gbch-xmq*plist.visibleItemCount:   8
```

10.2.7 Viewing a Batch Job

Selecting a batch job from the display then selecting the **View job** option from the **Jobs** menu opens this window.

The display may be scrolled through the script and panned across it using the vertical and horizontal scroll bars. The Search Menu provides options for specifying and finding text strings within the job script.

10.2.8 Changing Job and Variable parameters.

A job may be deleted, changed, reviewed by clicking on the line representing it in the job list and then selecting the required menu option or short cut button. Variables may be operated upon in exactly the same way. Some menu options and short cut buttons will have an immediate effect. Others will open a dialog window for additional information, such as this one for changing the Title, priority, shell and load level for a batch job.

Fields like Priority and Load level take a numeric value which may be typed in or adjusted using the increment and decrement buttons.

Plain text fields like Title simply allow any free text to be entered or modified.

Shown in previous examples are Square buttons which represent options that can be enabled or disabled (true or false). Diamond shape buttons (not shown in this example) are used for selecting one option out of a set of 2 or more possible options.

Some fields like the Command Interpreter in this example have a Select button next to them. These support two methods of parameter entry, straight text entry as in the Title field or by selecting one option from a list using a selection dialog. Clicking on the "Choose a command interpreter" button , for example, opens this selection dialog.

The information displayed is context and configuration sensitive, showing only the permitted and/or appropriate information.

This type of selection dialog does not insert multiple selections or pattern matching characters. However you can select an item from the list then type other information afterwards.

10.3 gbch-xr and gbch-xmr - Job Submission & Editing Tool

Used for creating, editing and submitting batch jobs, gbch-xr and gbch-xmr are GUI alternative to the command line program gbch-r. It provides all of the same options using the Graphical User Interface instead of command line options. It may also be used to edit the default options which are read from the current and home directories.

They can edit existing jobs that have been unqueued using gbch-q, gbch-xmq and gbch-jdel. Apart from submitting jobs to the queue they save them in the same "unqueued job" format. This uses two files, which are:

Command file

> Which is a shell script that holds the specification for the job. This shell script contains statements to reproduce the job environment followed by a gbch-r command. The gbch-r command has options to set up all of the job parameters and references the *job file* by name.

Job file

> Which contains the text, or script, of the job that is piped into the "command interpreter" by the scheduler.

A list of one or more jobs can be held by gbch-xr gbch-xmr at the same time. This is particularly useful when creating a group, or schedule, of related jobs.

The documentation below was written for gbch-xmr; gbch-xr is similar but uses the dialogs from gbch-xq.

10.3.1 Options

The option `-xrm "str"` may be specified for `str` to override any resources specified in the program's resource file, `GBATCH`.

Do not worry about the terminology, it is sufficient to recognise what these resources look like. For example the resource which specified the title for the main window of gbch-xmr is: for the program option or widget attribute. These override any resources specified in the program's resource file.

For example the resource which specified the title for the main window of gbch-xmr is:

```
gbch-xmr.title: gbch-xmr - Submit GNUbatch Jobs
```

The text before the colon "`:`" is the resource name (i.e. `gbch-xmr.title`) and the text after is the data, in this case a string. The data can be boolean, integer or a string for different resources.

10.3.2 Useful Resources

All of the resources for program gbch-xmr start with `gbch-xmr`. To keep the initial column width to manageable proportions just the text from the "`.`" or "`*`" is listed.

`.Title`	Specifies the Title as it will appear on the main window of gbch-xmr.
`.IconName`	Sets the text for the icon when the copy of gbch-xmr is iconised.
`.xtermEdit:`	This is a boolean with the values `True` or `False`. When `True` the editor specified by `gbch-xmr.editorName` will be run inside an X-Terminal session.
`.editorName:`	Specifies the name of the text editor to be invoked when editing the job file, or script, of a job.

10.3.3 Examples

The display can be restricted to a particular user or set of users and the Window title changed to act as a reminder. One `-xrm` option is used to set the title and another is used to restrict the view by user:

```
gbch-xmr -xrm "gbch-xmr.title: Tony\'s Batch Jobs" \
-xrm "gbch-xmr.onlyUser: tony" &
```

In this example the user is `tony` and the Window title is "`Tony's Batch Jobs`". The command has been broken over two lines by preceding the newline character with a back slash. It is not uncommon to use several lines if specifying `-xrm` options.

10.3.4 The Main Window

When gbch-xmr is invoked the main window will be displayed. By default it will look something like this:

The main screen is divided into two key functional areas. The top area contains menus and short cut buttons for issuing commands. The bottom area displays the list of batch jobs which are being worked on.

gbch-xmr uses similar windows and dialogs to gbch-xmq for specifying the job options.

To edit job scripts gbch-xmr invokes an editor of the user's choice. The editor is specified as a default parameter in the gbch-xmr options. On installation the default editor is vi.

The main screen has a row of menu buttons at the top, underneath of which is a row of short cut buttons. The bottom half of the window is a scrollable list of the jobs currently being worked on by gbch-xmr.

The bubbles in the above picture indicate what each field within a job entry represents.

10.3.5 The Menus and Shortcut Buttons

All commands are performed by selecting a menu option or clicking on the equivalent shortcut button. Some of the menu options may also be selected using shortcut keys, which are indicated to the right of the relevant options in each menu.

10.3.5.1 The Options Menu

For tailoring the look and feel of gbch-xmr, saving the tailored settings, viewing the error log and quitting.

View options brings up the dialog, to specify which text editor to use and whether it should be run inside an X-Terminal session or not.

Save options creates a local copy of the View options.

Select new directory for new jobs to be created in and for getting existing job files from.

Load defaults current dir reads any gbch-xmr default settings from the currently set directory.

Save defaults current dir saves the current gbch-xmr default settings in the currently set directory.

Load defaults home dir reads any gbch-xmr default settings from the user's home directory.

Save defaults home dir saves the current gbch-xmr default settings in the user's home directory. **Quit** terminates gbch-xmr.

10.3.5.2 The Defaults Menu

For specifying default options for all new jobs being created in this gbch-xmr session.

Default queue name Opens dialog to specify defaults for queue name, user name and Unix group.

Default set runnable state for jobs.

Default set cancelled state for jobs.

Default time Opens the standard job time and repeat specification dialog for setting default values.

Default title/pri/ll Opens the standard "title, priority, command interpreter and load level" dialog for setting default values.

Default Process params Opens the standard dialog for setting the process parameters: working directory, ulimit, umask, exit code ranges, advance time on error flag.

Default Time limits for detecting and stopping over running jobs.

Default Mail/Write job completion flags.

Default permissions for job access modes.

Default Arguments Opens standard dialog for specifying job arguments as defaults.

Default Environment Opens dialog for specifying a default job environment.

Default Redirections Opens standard dialog for specifying job I/O redirections as defaults.

Default Conditions Opens standard dialog for specifying job conditions as defaults.

Default assignments Opens standard dialog for specifying job assignments as defaults.

10.3.5.3 The File Menu & Buttons

Provides options for Submitting jobs, Creating, Editing and Deleting job files.

New job file creates a new, blank entry in the gbch-xmr job list. Once created the **Set command file name and Set job file name** menu options must be used before a job can be submitted or saved.

Open job file opens a previously saved or unqueued job for editing by gbch-xmr.

Close job file closes both the command file and the job file, then removes the entry from the gbch-xmr display.

Set job file name opens a file selector dialog for specifying the name of the job file.

Set command file name opens a file selector dialog for specifying the name of the command file.

Save job as 2 files: a command file containing the specification and a job file containing the script.

Edit job file containing the script.

Delete job files and gbch-xmr entry for the job which is currently selected from both the display.

Submit job which is currently selected to the scheduler.

10.3.5.4 The Jobs Menu and Buttons

Menu options for specifying the various job parameters.

Job queue name Opens dialog to specify the queue name, user name and Unix group.

Set job runnable This option is also available via the **Set ready** short cut button.

Set job cancelled This option is also available via the **Set cancel** short cut button.

Set job time parameters Opens the standard job time and repeat specification dialog.

Title, pri, Command int, loadlev brings up a dialog to specify the Job Title, Priority, Command Interpreter and Load Level.

Process Parameters Opens the standard dialog for setting the process parameters: working directory, ulimit, umask, exit code ranges, advance time on error flag.

Job time limits opens dialog to set parameters for detecting and stopping over running jobs.

Mail and write markers opens dialog for setting job completion mail and write flags.

Job permissions for access modes. I.e. read, write, etc.

Job Arguments Opens standard dialog for specifying arguments to be passed to the job.

Job Environment Opens dialog for the job environment variables.

Job I/O Redirections Opens standard dialog for specifying job I/O redirections.

Set job conditions Opens standard dialog for specifying job conditions.

Set job assignments Opens standard dialog for specifying job assignments.

10.3.5.5 Help

Context sensitive help for using gbch-xmr.

Help displays help for the current window.

Help on changes the operating mode from taking commands to displaying help on any object (menu, button etc.) that is selected.

About displays information, such as release number, about the version of gbch-xmr that is running.

10.3.6 Choosing a Directory

By default gbch-xmr will create new jobs and look for existing ones in the Current Working Directory when it was invoked. This can be changed to a new directory at any time by the **Select new directory** option under the **Options** menu.

This opens the standard dialog for selecting directories and files. Change to the required directory and click on OK. There is no need to specify a file name.

10.3.7 Creating a New Job

There are four essential operations required to create a new job. The first three are probably best done in sequence. This avoids gbch-xmr generating reminder messages later when it needs information from these operations. All of the operations are selected from the File menu as follows:

1. Select the **New job file** option, which produces a new entry line in the job list. Select this line, if it is not highlighted, by clicking on it.
2. Use the **Set command file name** option, to specify a name for the command file. A *Motif*file selection dialog will appear showing the contents of the current directory. Select a new file name. An alternative directory for the command file can also be specified in this dialog.

   When this is done, the file name will appear in front of the -> symbol on the selected entry in the job list.
3. Specify a name for the job file, by selecting **Set job file name** option. A similar *Motif*file selection dialog will appear, which is used to enter the file (and possibly path) in the same way as for the command file name.

   The job file name will appear to the right of the -> symbol this time.
4. It is now possible to create and edit the job script. Select the job by clicking on its entry in the job list and use the **Edit job file** option to invoke the text editor. The text editor is automatically loaded with the script for editing - in this case a blank file.

   There are no constraints on when or how many times the script may be edited, once the first three steps have been done.

When a new job is created it will be given a specification based on whatever default options are currently in force. These can include: queue name, job title and initial state which will be shown on the job list entry. Other settings can only be seen by opening the relevant specification dialogs, for example **Set times for job**.

10.3.8 Loading an Unqueued or Previously Saved Job

Use the **Open jobfile** option from the **File** menu to open a previously saved or unqueued job. This opens the standard Motif file selector dialog, in the currently set directory. The file list in the dialog is restricted to show only the Command files of each job. The dialog can be fooled by files which look like but are not valid command files. This is not dangerous, an error message will be displayed and another file can be selected.

Select the required job and click on OK. This loads the job specification into gbch-xmr and places an entry in the job list.

10.3.9 Setting up or Editing the Job Specification

Select the job by clicking on its entry in the job list. Any of the parameters in the selected jobs specification can now be edited using the options under the **Jobs** menu. This menu has a set of options almost identical to those in gbch-xmq. There are also shortcut buttons for some of these options.

10.3.10 Editing the Job Script

Select the job by clicking on its entry in the job list and use the **Edit job file** option from the **File** menu to invoke the text editor. The text editor is automatically loaded with the script for editing. Alternatively click on the **Edit job** short cut button instead of using the menu option.

Change the script, then save the changes and exit as appropriate for the editor. When this is done it is a good idea to save away the command file as well by selecting the **Save job** option from the **File** Menu.

10.3.11 Selecting a different Text Editor

By default gbch-xmr is shipped set up to run the vi editor inside an X-Terminal session. This can be changed to any suitable editor of the user's choice.

Select **View options** from the **Options** menu to open the Display options dialog. Type in the name of the desired editor, over the top of the existing name. If the editor has a Graphical User Interface then un-check the Run editor in "xterm" check-box. Otherwise make sure the box is checked to launch a terminal to run the editor.

10.3.12 Submitting Jobs

Jobs can be submitted by selecting their entry in the job list and clicking on the **Submit job** shortcut button, or by using the **Submit job** option from the **File** menu.

Jobs that have been submitted can be edited to produce other jobs and submitted again, as many times as required. If the job has not been changed before being submitted again gbch-xmr asks for confirmation.

10.3.13 Saving, Closing and Deleting Jobs

Jobs can be saved at any time. This saves the current specification of the job and leaves it open for further work.

Closing the job removes it from the job list. To avoid losing any changes save the job before closing it.

Deleting a job closes it and deletes the command and job files from the disk.

10.3.14 Specifying Defaults

The same default options as used by gbch-r, are loaded when gbch-xmr is started. The defaults are read from any `GBCH_R` keyword entries in relevant `.gnubatch` files and the `GBCH_R` environment variable if defined. These defaults are applied to any new jobs created in the gbch-xmr session.

The options under the **Defaults** menu enable the default options to be specified in the same way as the **Jobs** menu options change those for individual jobs.

Changes to the default settings can be saved using in the current or user's home directory. This is done using the **Save defaults current dir** and **Save defaults home dir** options from the **Options** menu.

A different set of, or the original, defaults can be loaded from a new directory or from the home directory. This is done using the **Select new directory** and **Load defaults ...** options from the **Options** menu.

10.4 Gbch-xuser, gbch-xmuser - GUI User Administration

gbch-xmuser is a fully interactive Motif alternative to the standard user permission manager (invoked using the command `gbch-user -i`). It is provided to maintain the list of user privileges, charges and default modes for both jobs submitted and variables created.

Unlike gbch-user there are no command line options to gbch-xmuser, it is always in the interactive mode (similar to `gbch-user -i`). The facility to change or specify resource settings for an X11 (and hence Motif) program on the command line can be used.

A list of user names or patterns for matching user names can be specified. This will restrict the display to the selected users, in the same way as restricting the display of programs like gbch-q.

Gbch-xuser is a GTK version of gbch-xmuser with a very similar interface.

10.4.1 Options

The option `-xrm "str"` may be specified for str to override any resources speciified in the program's resource file, `GBATCH`.

Do not worry about the terminology, it is sufficient to recognise what these resources look like. For example the resource which specified the title for the main window of gbch-xmuser is:

```
gbch-xmuser.title: gbch-xmuser - Edit user options
```

The text before the colon "`:`" is the resource name (i.e. `gbch-xmuser.title`) and the text after is the data, in this case a string. The data can be boolean, integer or a string. The data can be boolean, integer or a string for different resources.

10.4.2 Useful Resources

All of the resources for program gbch-xmuser start with `gbch-xmuser`. To keep the initial column width to manageable proportions just the text from the "`.`" or "`*`" is listed.

`.Title`	Specifies the Title as it will appear on the main window of gbch-xmuser.
`.IconName:`	Sets the text for the icon when gbch-xmuser is iconised.
`.sortAlpha:`	This is a boolean with the values `True` or `False`. When `True` the list of users will be sorted into alphabetical order by username.

10.4.3 Examples

When using an X-Terminal connected to several GNUbatch machines it can be very useful to include the hostname in the Window and Icon titles. The display can be restricted to a particular user or set of users and the Window title changed to act as a reminder. One `-xrm` option is used to set the title and another is used to restrict the view by user:

```
gbch-xmuser -xrm "gbch-xmuser.title: gbch-xmuser - $HOSTNAME" \
-xrm "gbch-xmuser.IconNAME: $HOSTNAME" &
```

In this example the machine's name will be picked up from the environment variable `HOSTNAME`. It is not uncommon to use several lines if specifying `-xrm` options.

10.4.4 The Main Window

When gbch-xmuser is invoked the main window will be displayed. By default it will look something like this:

The bottom area contains a list of all users in the password file with their GNUbatch permissions and settings. It will have a scroll bar if there are more users than can fit on the screen. Above this is a pane containing the default settings.

At the top of the screen is the menu bar supporting all of the gbch-xmuser commands. Each menu option opens a dialog or operates on the specified data immediately. With a few exceptions these are straight forward and easy to understand.

10.4.5 The Menus and Options

All commands are performed by selecting a menu option. Some of the menu options may also be selected using shortcut keys, which are indicated to the right of the relevant options in each menu.

10.4.5.1 The Options Menu

For changing the ordering of users in the display, saving the settings and quitting.

Display order brings up the Display options dialog, to tailor the look and feel. Pressing the Control and O keys also invokes this option.

Save options creates a local copy of the Display order.

Quit saves any changes to the default and user accounts. (N.B. It is at this point that all changes are saved).

10.4.5.2 The Defaults Menu

Administers the default account parameters.

Priorities opens the dialog for setting up the maximum, minimum and default priorities for the user(s).

Load Level opens the load level specification dialog. This sets the following:

- Maximum load level permitted for any one job submitted by the user
- Maximum number of jobs measured in load that the user may have running at any time
- A default value for load level for the user if they have *special create* privilege.

Mode brings up a dialog for setting the default access modes on all jobs submitted and variables created by the user.

Privileges opens a dialog showing and allowing changes to the privileges (e.g. submit jobs and create variables).

Copy to all users Copies the default settings to all users - *use with care*.

10.4.5.3 The Users Menu

Administers the accounts of individual users

Priorities opens the dialog for setting up the maximum, minimum and default priorities for the user(s).

Load Level opens the load level specification dialog. This sets the following:

- Maximum load level permitted for any one job submitted by the user
- Maximum number of jobs measured in load that the user may have running at any time.
- A default value for load level for the user if they have *special create* privilege.

Mode brings up a dialog for setting the default access modes on all jobs submitted and variables created by the user.

Privileges opens a dialog showing and allowing changes to the privileges (e.g. submit jobs and create variables).

Copy defaults resets the users account to the default settings - *use with care*!

10.4.5.4 The Charges Menu

Displays and manages the batch processing charges.

Display Charges for all selected users

Zero charges for selected users

Zero charges for ALL users

Impose fee on selected users.

10.4.5.5 The Usermacro Menu

Provides macro commands for use on the selected users.

Run command macro opens a dialog prompting for the name of the macro to run. This is then invoked by gbch-xmuser with the name(s)of the selected user(s).

< *macro* > runs the pre-defined macro, whose name or a brief description will appear in place of the < *macro* > place holder.

Up to 9 macros may be pre-defined.

10.4.5.6 The Help Menu

Context sensitive help for using gbch-xmuser.

Help displays help for the current window.

Help on changes the operating mode from taking commands to displaying help on any object (menu, button etc.) that is selected.

About displays information, such as release number, about the version of gbch-xmuser that is running.

10.4.6 Selecting multiple users for menu options

Many of the menu options can be carried out on more than one user at a time. First select all of the relevant users from the list. Then select the required menu option for the operation you wish to perform. The set of users remains selected after you have completed the operation in case there are other options that you want to use on them.

You can scroll up and down a long list of users without losing those you have selected so far. The following mechanisms allow you to select a group of users:

- Hold the selection mouse button down and drag it across a contiguous set of users.
- Using the mouse click on the first user in the required set. Move to the last user in the set , hold the shift key down on the keyboard and click on this user.
- Individual users may be added or removed from the set by clicking on them whilst holding the Control key down on the keyboard.

10.4.7 Copying defaults to all users

Available via the **Copy to all users** option under the **Defaults** menu. It copies the default settings to all users except the one you are logged in as.

This command should be used with care. It is possible to remove essential permissions from everybody including write administration file privilege.

Once this has happened you have to re-create the user permissions file from scratch.

10.4.8 Resetting a user to the default

This is a very useful mechanism for setting permissions for a group of users. Set the default to the required value then apply it to the required users. If you do not want new users to inherit this default setting remember to return it to the original state.

Be careful not to remove write administration file privilege from all users.

10.5 gbch-xfilemon and gbch-xmfilemon - Interfaces to gbch-filemon

These programs provide simple interfaces for GTK and Motif to gbch-filemon. They provide a single dialog box of options to invoke gbch-filemon with.

Options may be set to initialise the default settings in the dialog box.

10.5.1 Useful Resources in gbch-xmfilemon

All of the resources for program gbch-xmfilemon start with gbch-xmfilemon. To keep the initial column width to manageable proportions just the text from the "`.`" or "`*`" is listed.

`.title:`	Specifies the title as it will appear on the main window of xmfilemon.
`.iconName:`	Sets the text for the icon when gbch-xmfilemon is iconised.
`.pollFreq:`	Specify default poll frequency.
`.nomodTime:`	Sets the default time for the "stopped growing", etc options.
`.daemon:`	Set to `True` or `False` to denote whether gbch-filemon should detach as a daemon process.
`.contFound:`	Set to `True` or `False` to denote whether gbch-filemon should continue after finding the first file.
`.includeExist:`	Set to `True` or `False` to denote whether gbch-filemon should include existing files.
`.scriptCmd:`	Set to `True` or `False` to denote whether gbch-filemon should run a shell script or invoke a command.
`.style:`	Set default style of operation, (what to watch for) set to `appears`, `deleted` (or `removed`), `nogrow`, `nomod`, `nochange` or `noaccess`.
`.typeMatch:`	Set default type of file name matching, `any`, `pattern` or `specific`.
`.pattern:`	Specify default file name or pattern for pattern or specific matching.
`.directory:`	Specify default directory to search, if not current directory on entry.
`.script`	Specify default command or shell script.

11 Configurability

This section looks at the various options available to user programs such as gbch-r and gbch-q. In some cases, particularly with gbch-r and gbch-jchange, there is a perhaps bewildering array of options. It is not intended that users should have to remember them all as they can be specified by default and overridden as required.

This is done by using *configuration files* and environment variables. For example, the program gbch-q may be configured to take advantage of function keys and show a different set of information in a simplified format.

```
Seq   Job Name              Args       Date/Time          Prog
-----------------------------------------------------------------------
  1   start                           08/06/01 10:54     Canc
  2   Process directory    /home      08/06/01 10:54
  3   Process directory    /usr       08/06/01 10:54
  4   Process directory    /tmp       08/06/01 10:54
  5   Collect data                    08/06/01 10:54
  6   Error Handler                   08/06/01 10:54
  7   cleanup                         08/06/01 10:54
  8   setup                           07/06/01 23:01     Done



----F1------F2------F3------F4------F5-----F6--------------------------
   help    enable disable   set     view    view
           run    run       time    job     vars
-----------------------------------------------------------------------
         GNUbatch gbch-q (c) Free Software Foundation 2009
```

This configuration could be specific to a particular user or activity. In this case it is taken from a real configuration belonging to user wally when using jobs in a queue named par. The screen display has been changed as follows:

- The fields `jobno`, `Shell`, `Pri`, `Load` and `Cond` have been removed.
- The `Time` field is changed from the abbreviated form to show time in full, `Date/Time`. The `Title` field is widened to display more text.
- The `Argument` field has been added. This shows the differences between jobs which are identical except that they use different data as specified in the arguments.
- Column headings underlined and footer expanded to include function key reminders.

The set of jobs displayed has also been restricted to show just those in the queue named `par` that belong to user `wally`.

This is what the standard configuration of gbch-q looked like when invoked by user `wally`, on another terminal, at the same time:

```
Seq Jobno    User     Title          Shell    Pri Load  Time  Cond      Prog
  0 340     wally   e-mail:dial u sh      150 1000  16:33
  1 734     tony    prog_a        sh      150 1000  06/02           Run
  2 1420    wally   Output Exampl sh      150 1000  29/01           Err
  3 735     tony    prog_b        sh      150 1000  08/02 A_STATUS
  4 736     tony    prog_c        sh      150 1000  08/02 A_STATUS
  5 439     wally   wally         sh      150 1000                  Canc
  6 588     wally   Also Sprach Z sh      150 1000  04/02           Done
  7 564     wally   Daily Update  sh      150 1000                  Run
  8 455     pior    Simple Job    sh      150 1000  11/03           Abrt
  9 309     wally   par:start     sh      150 1000  08/06           Canc
 10 310     wally   par:Process d sh      150 1000  08/06 **Cond**
 11 312     wally   par:Process d sh      150 1000  08/06 **Cond**
 12 313     wally   par:Process d sh      150 1000  08/06 **Cond**
 13 314     wally   par:Collect d sh      150 1000  08/06 **Cond**
 14 315     wally   par:Error han sh      150 1000  08/06 **Cond**
 15 316     wally   par:cleanup   sh      150 1000  08/06 **Cond**
                             -- 9 more jobs below --
=======================================================================
GNUbatch gbch-q (c) Free Software Foundation 2009   (? for help)
```

On the standard configuration jobs owned by users `tony` and `pior` can be seen along with other jobs owned by `wally` which were not relevant to the task in hand.

11.1 Configuration files and environment variables

Configuration files are called `.gnubatch` and may appear in the current directory, or in the user's home directory. They are text files, containing environment variable type assignments. These "environment variables" may be used to specify program options and alternative message files.

Options to the user programs enable these files to be generated and edited automatically.

11.1.1 Environment Variables

The default options to each program may be overridden and others specified on the command line. A local default may be set up for each program by putting the options in an environment variable.

For example using the `-C` and the `-r` options to gbch-r to submit a batch job in the `Cancelled` state with a repeat time. If the job script is held in the file `fred` and the required repetition is every day, then the gbch-r command will look something like this:

```
gbch-r -r Days:1 -C fred
```

If several batch jobs are being submitted, requiring the same options, it would be easier to put them into environment variable `GBCH_R`. For example:

```
GBCH_R='-r Days:1 -C'
export GBCH_R
```

These options would be automatically specified each time, until the environment variable is unset or changed. To override an environment variable just re-specify the option on the command line. Command line options take precedence over environment variables. The general rule is that for every option which "does" something, there is a corresponding option to "undo" it, to provide for this case.

When something more permanent is required `.gnubatch` configuration files can be used.

11.1.2 Configuration files

A configuration file called `.gnubatch` may be put in the current directory to set options appropriate to running GNUbatch programs in this directory. The format of the file is similar to setting environment variables, but without the quotes or "export" statements.

For example:

```
GBCH_R=-r Days:1 -C
GBCH_JLIST=-F "%N %H %P"
# ..... and so on
```

The file may contain comment lines commencing with `#`. In fact any lines not understood are silently ignored.

As with environment variables the `.gnubatch` file may be overridden on the command line.

To specify default options for a user, whichever directory is in use, put a `.gnubatch` file in the user's home directory. For example; to set gbch-q to show only the user's jobs and variables on entry, put a `.gnubatch` file in the home directory containing the line:

```
GBCH_Q=-u user
```

This has a similar effect as setting up the environment variable `GBCH_Q` in the user's `.profile` or `.login` file.

There is an order of precedence for options in home directory `.gnubatch` files, current directory `.gnubatch` files, and in an environment variables. They are handled in the following order:

1. Any system-wide defaults are taken (e.g. the user's default job priority).
2. The home directory `.gnubatch` file is processed.
3. The environment is processed.
4. The current directory `.gnubatch` file is processed.
5. Options on the command line are processed.

Conflicting options encountered later completely override what came first, so that options specified on the command line will take priority whatever else was encountered. As mentioned above, there is a "reset" type option for every "set" type option, so for example the gbch-r option `-N` resets the option `-C`.

The functions of the option letters can be re-assigned using alternative help message files. These may also be specified using environment variables and/or the `.gnubatch` files.

11.1.3 Environment variable or keyword names

The table lists examples of the environment variables, or keywords in `.gnubatch` files, used to hold various program options. The environment variable or keyword for program options has the same name as the program it affects, except that it is in upper case and non-alphanumeric characters replaced by underscore, for example the keyword for gbch-q is `GBCH_Q`, gbch-r is `GBCH_R` etc.

The environment variable names are taken from the program names. If any of the programs are given a different name the environment variable names will change.

To set the environment variable, the format is just the same as for the options in the corresponding command. For example using the Bourne or Korn shells, type:

```
GBCH_R='-C -r Minutes:30'
GBCH_Q="-u $LOGNAME"
export GBCH_R GBCH_Q
```

The quotes (single or double) are required if spaces are included, which they usually are.

With the C shell type:

```
setenv GBCH_R '-C -r Minutes:30'
setenv GBCH_Q '-u $LOGNAME'
```

There isn't any hard and fast rule about whether to use home or current directory `.gnubatch` files, or environment variables.

In practice people tend to put "comfort"-type options, such as help display options and the display of job numbers in the home directory. Options specific to files in a given directory, such as batch job queue name and I/O redirections, would go in the current directory. Transient requirements or those set up by applications invoking GNUbatch programs are best put in the environment.

11.2 User reconfiguration

To allow maximum flexibility, all strings, such as screen headers, error and help messages, keystrokes and prompts used in GNUbatch are taken from a set of files. These message or help files can be edited and different versions of each file may be used for different contexts. Some examples are:

a. Customising the interface on a system-wide basis by editing the default copies of the files.

b. Producing different versions to take advantage of the best facilities on different types of terminal.

c. Tailoring on an individual basis for each user by allowing each user to have access to their own version or versions of the files.

d. Tailoring for a group of users by making what seem to be their individual copies, read only symbolic links to a master copy.

e. Activity based versions pointed to by environment variables.

As mentioned above all the keystrokes understood by gbch-q and gbch-user are "soft". The functions may be redefined as required. The following examples show the kinds of use these facilities can be put to:

1. Producing customised versions of the product incorporating site names, help messages etc. on the basic screen formats.
2. Providing "seamless joins" between GNUbatch and other software with different function key sets.
3. Providing interfaces appropriate to different terminals, in particular, taking advantage of function keys provided on those terminals.
4. Providing support for national languages - allowing different languages on different terminals on the same machine.

11.2.1 Message files

The standard message files all live in the directory `/usr/local/share/gnubatch/help` (All of these names may be over-ridden by assignment to environment variables as below). The files involved are:

`btuser.help`	The configurable information for gbch-user
`btq.help`	The configurable information for gbch-q
`filemon.help`	The configurable information for gbch-filemon and xmfilemon
`xmbtq.help`	Message file for gbch-xq and gbch-xmq
`xmbtr.help`	Message file for gbch-xr and gbch-xmr
`xmbtuser.help`	Message file for gbch-xuser and gbch-xmuser.
`btrest.help`	Which contains all the configurable information for all the rest of the user-accessible utilities
`btint-config`	containing the configurable information for the programs internal to GNUbatch: (btsched, xbnetserv, etc.)

The files are found by default from `/usr/local/share/gnubatch/help` as specified.

To specify an alternative file, use the configuration file or environment variable mechanism previously described. The following table lists the environment variables and/or keywords used for various user programs:

Environment variable or keyword	**Description**
`BTQCONF`	gbch-q message file
`BTUSERCONF`	gbch-user message file
`FILEMONCONF`	gbch-filemon gbch-xfilemon gbch-xmfilemon message file
`XMBTQCONF`	gbch-xmq message file
`XMBTRCONF`	gbch-xmr message file
`XMBTUSERCONF`	gbch-xmuser message file
`BTRESTCONF`	Message file for other utilities

For example to use an alternative message file for gbch-q, specify its use by means of the environment variable setting:

```
BTQCONF=`pwd`/my-gbch-q-file
export BTQCONF
```

It is important to specify the full path name, otherwise the file will be searched for in whatever directory is current.

Just like the command line options, the message file may be specified in a configuration file `.gnubatch` file. If it is located in the home directory, it will apply whichever directory is current, otherwise it will apply to the current directory.

The format of lines in the `.gnubatch` files is similar to that used to set environment variables in the shell. For example:

```
BTQCONF=$HOME/lib/mybtq.help$TERM
```

Note that environment variable names are also expanded here, so in this example the user is intending to specify a different file according to the setting of the `TERM` environment variable.

There are 3 facilities in the expansion of these lines intended to assist the user to supply defaults etc:

Firstly, as with the shell, sequences of the form `${VAR-default}` are replaced by the value of environment variable `VAR` if it exists, and otherwise the default string specified.

Secondly the sequence `$0` is replaced by the name (or the last component of the file name) by which the program was invoked. For example

```
BTRESTCONF=$HOME/lib/helps/$0.help
```

relates the message file's name to the name of the program invoked. Hence it would be `gbch-jlist.help` if gbch-jlist was run, `gbch-var.help` for gbch-var, and so on.

Finally, if it cannot find the file specified, then an attempt will be made to find a file name constructed by deleting the last part of the file name from the path and substituting the standard name (`btq.help`, `btrest.help` etc), before giving up. For example in the above case, if the files `gbch-jlist.help` or `gbch-var.help` could not be found in the directory `$HOME/lib/helps`, then a further try would be made with the standard name `btrest.help` to give the file name `$HOME/lib/helps/btrest.help`.

As for the argument keywords and environment variables, the current directory `.gnubatch` file takes precedence over environment variables, if any, which in turn take precedence over the home directory `.gnubatch` file. However once the keyword is found in one of those places, the remainder are not searched; this means that if a non-existent file is specified, say in the current directory `.gnubatch` file, then the program will abort without looking in the other places.

11.2.2 File format.

It is easiest to work on these files by copying one of the supplied ones and starting from there. The files are plain text which can be edited with any suitable text editor.

Message files have a common format and notation:

- Blank lines are ignored.
- Comment lines, beginning with `#`, are ignored.
- All other lines are *definitions*.

Example lines within the files might be:

```
K400:?
1K500:o,*
100P:Ok to delete (Y/N)
E100:Unknown command - expecting job queue control
H400:Please reply Y or N
```

The general form of the definition lines is:

1. An optional *state code*, denoting the state of the program in which the line applies.
2. A key letter, denoting the function, i.e. key definition, prompt, error or help message etc.
3. A code number for which the program looks when it requires a particular line.
4. A colon preceding the definition of the item.

The state and code numbers are arbitrary and compiled into the relevant programs. Hopefully the context and the comments in the supplied files will make the situations in which they are used clear enough.

The types of entries given by the key letters are as follows:

`K`	a key mapping
`H`	a "*help*" or similar type message
`E`	an *error* or other information message
`P`	a "*prompt*" or other screen field
`N`	a numeric parameter
`A`	an *option* definition or an *alternative*
`AD`	a default *alternative*

In addition there are screen heading lines, which have a slightly different format using other letters, defined later.

11.2.2.1 Key definitions

Key definitions are read in when the program is first started. As a result conflicts, where a key is defined for two or more different uses, and other errors are detected before further processing is done.

There are two types of key definition; *global* definitions which apply everywhere within the program, and *local* definitions, which only apply in a particular context, enabling you to use the same key in another context for a different purpose.

A global key definition looks like this:

```
K400:?
```

whilst a local definition looks like this:

```
1K500:o
```

The first number is referred to as a *state code*. The number after the `K` gives the internal code into which the key sequence is translated and used to select the program action.

In all the contexts where state codes apply, there is a help message supplied with the same code as the state code. For example state 1 corresponds to the state where the cursor is in the job queue, keys relevant to this state only are prefixed with this state code, and the help message lines for this state start with "`H1:`".

The balance of the key definition gives the character sequences which make up the key definition; these will be translated to the internal code. In the above examples the character `?` is globally translated to the internal code 400 and `o` to 500 in state 1.

Non-printing and control characters are represented using the prefixes `\` and `^`. In particular commas must be represented as `\,` and spaces and tabs as `\s` and `\t` respectively.

To give two or more keys for a given command, separate them by commas, as follows:

```
K400:?,*
```

Hence the need to represent commas using `\,` as just mentioned. The symbols for the non-printing characters are:

Character	**Description**
`^a` etc	Appropriate control character
`^^`	Denotes a single `^`
`\\`	Denotes a single `\`
`\,`	Denotes a `,`
`\s`	Space
`\t`	Tab
`\b`	Backspace
`\e`	Escape
`\n`	Linefeed
`\r`	Carriage return
`\f`	Form feed
`\xnn`	Character given by hexadecimal *nn*
`\`*nnn*	Character given by octal *nnn*
`\kkeyname`	String or character given by the *keyname* (see below)

Many keys, in particular cursor and function keys, generate multi-character sequences. The sequences can be just written out thus:

```
K406:\e[A
```

The whole of the sequence up to the end or to a comma will be searched for and translated to the given internal code when it occurs. The arrival of the characters will be timed so it should be possible to distinguish between key sequences with similar prefixes (often the case with cursor and function keys) and the effect of typing the component characters separately. This particularly affects the escape character, which on its own is commonly used to abort input, but which also invariably prefixes function and cursor key sequences.

If necessary the timing of the arrival of characters can be tuned by means of the environment variable `KEYDELAY`. This is set to the number of 1/10ths of a second to wait between the characters of a suspected function key.

If another character is not received within this time, the character will be assumed to be a single key, otherwise it is taken as the start of a function key. The default for `KEYDELAY` is 3/10ths of a second, which should work correctly for the vast majority of terminals.

If this value is too low, then function keys may not always be recognised, whilst if it is too high, then the response to a single escape character, when required, will appear to be slow.

11.2.2.1.1 Special key sequences.

As well as the control and non-printing characters which may be inserted using the sequences "`\n`" and "`^c`" etc, additional constructs introduced by "`\k`" are provided to import the user's choice of keys for erase, kill etc, and the *termcap* or *terminfo* definitions of some of the special function keys on the terminal.

There are two problems which can arise:

1. Choices of single-character keys often conflict with existing uses of the same key, but not in a consistent way for each user or terminal. For example:
   a. The character *ctrl*-U (^U) is defined in the supplied files as "scroll-half up". Many people use this as a kill character.
   b. Some terminals have cursor keys which generate a single character. For example the left cursor key may generate a backspace (*ctrl*-H) character which cannot be distinguished from a real backspace character (or indeed *ctrl* and 'h').
2. *termcap* and *terminfo* files may contain incorrect specifications about what sequences are generated by cursor and function keys. This may not always be the fault of the files themselves if the user has extensively used softkey-setup facilities on the terminal.

The way to import these values is to use the sequence "\k" followed by the symbolic name of the required item. The items available are:

User defined keystrokes	
Symbolic Name	**Meaning**
`\kINTR`	user's interrupt key
`\kKILL`	user's kill key
`\kERASE`	user's erase key
`\kQUIT`	user's quit key

which are looked up on entry to gbch-user or gbch-q from the *termio* or *stty* settings, as set by stty, and:

Terminal keys	
Symbolic Name	**Meaning**
`\kUP`	string sent by up arrow key
`\kDOWN`	string sent by down arrow key
`\kLEFT`	string sent by left arrow key
`\kRIGHT`	string sent by right arrow key

Terminal keys	
`\kHOME`	string sent by home key
`\kFnn`	string sent by function key F*nn*

which are looked up in the *terminfo* or *termcap* database for the terminal type specified in the environment variable `TERM`.

To use these sequences, put the relevant one in place in the key definition for example:

```
K400:?,\kF1
K406:k,\kUP
```

If something does go wrong due to one of the two problems given above, the program will display some error messages and terminate. For example

```
State 5 setup error - clash on character 08 with previously-given
value 420 and new value 530
```

In this case the problem can be pinpointed by looking for a key definition with 420 in (this is the code for erase last character) and another with 530 in (left cursor in some fields). This particular message often appears on terminals where left cursor is the same as backspace. In other words `\kLEFT` and `\kERASE` produce the same result - control-H or hexadecimal 08.

The other thing which might happen is that the function keys do not work properly or the message `Undefined key sequence` is displayed. This is because there is an incorrect *termcap* or *terminfo* file, or function keys have been reset.

A common problem on VT100-based terminals is that there are four settings depending upon which combination is selected of "normal" or "application" cursor keys and "numeric" or "application" keypad. It is possible that the *terminfo* file will assume one of the combinations whilst the terminal will be set to one of the other three.

This particularly affects cursor keys which are often defined in the *terminfo* file to be `\eOA`, `\eOB`, `\eOC` and `\eOD` (application keys) for up, down, right and left cursor respectively whilst the terminal when switched on generates the "normal" cursor keys `\e[A`, `\e[B`, `\e[C` and `\e[D` respectively. If this problem occurs we suggest spelling out the combinations explicitly, for example

```
K406:k,\e[A,\eOA
K407:j,\e[B,\eOB
```

Remember to adjust any help and error messages to reflect the keys that have been changed.

11.2.2.1.2 Help and error messages

Help and error messages are treated almost identically. The only real difference is that they are displayed slightly differently.

The text to be displayed is given on as many lines as required with the same prefix, for example:

```
E1:Unknown command - expecting job queue control
```

```
E100:Type ? for help
H1:? - help (this file)
H1:^L - redraw screen
```

There are specific variables or strings which it is helpful for the help or error message to quote in the message. They are inserted using "parameters" introduced by `%` as follows:

Parameter	Meaning
`%c0` to `%c9`	Numeric parameter 0 to 9 interpreted as a character, or hexadecimal if non-printing
`%d0` to `%d9`	Numeric parameter 0 to 9 interpreted in decimal
`%E`	System error message
`%F`	Name of message file
`%f`	Floating-point number (2 decimal places)
`%G`	Group name of *effective gid*
`%g0` to `%g9`	Numeric parameter 0 to 9 interpreted as group id.
`%H`	Group name of *real gid*
`%N`	Host name
`%P`	Program name
`%p`	Numeric process id
`%R`	User name of *real uid*
`%s`	Primary string parameter
`%t`	Secondary string parameter
`%U`	User name of *effective uid*
`%u0` to `%u9`	Numeric parameter 0 to 9 interpreted as user id
`%x0` to `%x9`	Numeric parameter 0 to 9 interpreted in hexadecimal

The actual parameters are provided by the context. Some of the information (program name, process ID, error code, real and effective user & group IDs) is always available. The rest of the information (the 2 string arguments, the 10 parameters) is set up as required for each error message by the calling program. The only way to discover what has been passed is to examine the error message, as given in the standard help/message files, and to use the same parameters, possibly with a different format.

11.2.2.1.3 Option syntax

There are two alternatives to using the standard single letter command-line options:

1. You can invoke options with keywords such as `+priority` or `--priority` instead of or as well as single-letter options such as `-p`.
2. Alternative option letters and keywords may be specified by use of different message files.

Option syntax is defined by means of lines of the form

```
A119:p,priority
```

As with other parts of the file, there is an internal state code, in this case 119, to denote the action to take. Alternatives are specified after the colon and separated by commas. They may consist of either:

1. Any single printing character, as for p in the above example, which denotes a "`-`" type argument, thus `-p`. The character is taken exactly as given, and can be any printing character (a comma or `\` must be escaped using a `\` thus `\,` `\\` but these, especially the latter, are very strongly discouraged, for obvious reasons).
2. A keyword consisting of alphanumeric characters, `-` and `_` starting with a letter, denoting a keyword-type argument, thus `+priority` or `--priority`. Letters in the keyword are case insensitive.

It is not compulsory to define either single-character or keyword type options for any function. If one type is left out then only the other type will be available. If both types are left out the option will not be available.

If you leave out option syntax definitions out of the file altogether, then a default set of options consisting only of the single-character variants of the options as defined in this manual for each program will be supplied.

Please be careful before getting carried away!.

When a job is unqueued from gbch-q, the program jobdump will be run and will pick up the version of the message files as for gbch-r. It starts with configuration files in the current directory from which gbch-q was invoked, using the options specified to generate arguments for a gbch-r command. The same considerations apply to options saved using `+freeze-current` and `+freeze-home` options to various commands.

Take care because this means that with completely different option definitions in various places, the current set may be inappropriate (particularly with the unqueue operation). It is best not to have different specifications for the options in different places on the machine(s). For example do not set things up so that `gbch-r -C` means submit job cancelled in some contexts but something different elsewhere.

11.2.2.1.4 Alternatives

In many contexts, such as the progress code of a job, it is necessary to select a string according to a numeric value. These strings are presented together in the form:

```
<state code>A<numeric parameter>:<text>
```

For example (from `btq.help`)

```
100AD0:
100A1:Done
100A2:Err
100A3:Abrt
100A4:Canc
100A5:Init
100A6:Strt
100A7:Run
100A8:Fin
```

The numeric parameters 0 to 8 represent the numeric values of the progress codes and the alternatives for internal state 100 yield the appropriate string.

These alternatives are also used in questions requiring a choice; a prompt will be generated consisting of all the alternatives in the order given. The default alternative is given by `D`.

Comments in the supplied message files show where this applies.

11.2.2.1.5 Prompts.

Strings introduced with sequences such as 100P: are *prompts*. These are transient messages which are generated in various places such as:

- Messages at the bottom of the screen indicating so many jobs below.
- Prompts for user input, e.g. when a job or variable parameter is being changed
- Prompts for confirmation on "sensitive" commands, like deleting a job.

Some prompts have parameters in (such as the "`n jobs below`" message), which are introduced with a `%` sequence such as `%s` or `%d`. These are similar to the constructs in the C programming language printf function[10].

11.2.2.1.6 Numeric parameters

There are some sequences of the form

```
1000N1001
```

in the files. These are used in 4 places:

1. In the gbch-q "job options" screen, to configure the order of prompts and the prompt to start with.
2. In the gbch-user display or set privileges options, to configure the order to present the privileges.
3. In the "save options" screens in both gbch-q and gbch-user to configure the order in which the prompts are presented.
4. To present certain numeric parameters, such as the default number of repeat units, or default constant for increment/decrement of variables.

Please see the comments in the supplied files for how to adjust these parameters.

11.2.2.1.7 Titles

The screen titles used by gbch-q and gbch-user are handled slightly differently. Although they can be multi-line, each line of the title is taken from a different message number. For example, `J`*`n`*`:` is the format for the *n*th line of the title for the gbch-q jobs list. (The `J` stands for "jobs"). The default is:

```
J1:j
```

This creates a one line title which has names of each column generated automatically.

---

[10] Exactly similar in fact, as it actually uses the printf function.

If the automatically generated title is not suitable it may be replaced by removing the letter `j` and putting the desired text after the : like this

J1:Seq Jobnum Owner Description ... *and so on*

The footer for the jobs screen is specified by lines starting `Fn:` for example

```
F1:========================================= ...
F2:GNUbatch %P (c) Free Software ...
```

Similarly the title and footer for the gbch-q variable list is taken from lines starting with `Vn:` and `Gn:` respectively. Each variable is described over two lines, hence two automatically generated column headings are available. The default title for the variable list uses these auto-generated headings like is:

```
V1:1
V2:2
V3:---------------------------------------------
```

The height of the title or footer is given by the highest-numbered line specified - 3 in the above case. If line `V2:` had not been specified a blank line would be inserted between lines `V1` and `V3`.

Program gbch-user, when invoked for interactive operation by the `gbch-user -i` command, takes the title and footer specifications from lines starting `Ln:` and `Fn:` respectively.

It is permissible to omit any or all of these titles completely if desired, leaving more space on the screen for jobs, variables or users, as the case may be.

11.2.2.1.8 Enhancements and line drawing in headers

It is possible to specify screen enhancements and line drawing characters within headers. To do this, the following sequences are interpreted within headers lines.

`\B`	Set bold mode
`\D`	Set dim mode
`\F`	Set flashing mode
`\U`	Set underlined mode
`\I`	Set inverse video mode
`\S`	Set standout mode
`\N`	Reset to normal mode

The effect of the these sequences are cumulative and apply until the end of the line or until the `\N` sequence is encountered.

For example the line

```
\UExample\N Header
```

would be displayed as

<u>Example</u> Header

Note that, not all *terminfo* files define all of the various enhancements, also that "standout" mode is usually some arbitrary combination of the others.

The line-drawing set, if available, is separately invoked by use of sequences thus;

`\L`	Turn on line drawing mode and display upper left corner line drawing symbol.
`\l`	Turn on line drawing mode and display lower left corner line drawing symbol.
`\R`	Turn on line drawing mode and display upper right corner line drawing symbol.
`\r`	Turn on line drawing mode and display lower right corner line drawing symbol.
`|`	Turn on line drawing mode and display vertical edge line drawing symbol.
`\-`	Turn on line drawing mode and display horizontal edge line drawing symbol.
`\+`	Turn on line drawing mode and display internal intersection line drawing symbol.
`\<`	Turn on line drawing mode and display intersection of horizontal line and left edge 'T' line drawing symbol.
`\>`	Turn on line drawing mode and display intersection of horizontal line and right edge 'T' line drawing symbol.
`\^`	Turn on line drawing mode and display intersection of vertical line and top edge 'T' line drawing symbol.
`\V`	Turn on line drawing mode and display intersection of vertical line and bottom edge 'T' line drawing symbol.

Once line-drawing mode is set, then any of the above characters which immediately follow are interpreted without needing to have a "`\`"in front, so for example

```
\-------
```

would generate a horizontal line.

Any character not in the above set of line-drawing characters will cancel line-drawing mode. So that you can put a potential line drawing character in a box next to a line, then the sequence

```
\.
```

cancels line-drawing mode without inserting a period.

For example to produce a display like this;

use the sequence

```
T1: \L---------R
T2: \|  Box    \|
T3: \l---------r
```

This applies to all headers of screens in gbch-q and gbch-user.

To display a "`\`" in a header it must be "escaped" with a preceding "`\`" like this "`\\`".

11.2.2.2 Changing message files

A few tips may be useful:

- Tabs don't work in messages, since they may displayed anywhere on the screen; use multiple spaces instead.

- Keep the messages reasonably short, this avoids having to redraw large areas of the screen just to display a message and obliterate everything else going on.
- When re-defining keys, don't forget to adjust the help and error messages to correspond, and make sure that they are all consistently re-defined throughout all the states of all the programs.

Please note that there is nothing "magic" about the code numbers for global and local keystrokes. As distributed the state codes 400 up to 499 are assigned to global keystrokes, and 500 upwards to local ones. If, for example, it would be required to define a different help key in different places, then re-define `1K400` etc.

Likewise, the order of access to screens in gbch-q can be changed by redefining local versions of the quit key (405) and selection of job and variable screens.

11.2.3 Environment variables.

The default set of directory and file names used by GNUbatch is "built in", but most can be changed by assignment to environment variables. The relevant environment variables have already been described in relevant sections. To recap they are:

Variable	**Default**	**Description**
`BTQCONF`	`$SPHELPDIR/btq.help`	gbch-q message file
`BTRESTCONF`	`$SPHELPDIR/btrest.help`	message file for other utilities
`BTUSERCONF`	`$SPHELPDIR/btuser.help`	gbch-user message file
`FILEMONCONF`	`$SPHELPDIR/filemon.help`	gbch-filemon message file
`MAILER`	`/bin/mail`	program used to send mail
`SPHELPDIR`	`/usr/local/share/gnubatch/help`	message files
`SPOOLDIR`	`/usr/local/var/gnubatch`	internal databases
`SPROGDIR`	`/usr/local/libexec/gnubatch`	internal programs
`XMBTQCONF`	`$SPHELPDIR/xmbtq.help`	gbch-xq message file
`XMBTRCONF`	`$SPHELPDIR/xmbtr.help`	gbch-xr message file
`XMBTUSERCONF`	`$SPHELPDIR/xmbtuser.help`	gbch-xuser message file

These variables (excluding those starting with `SP`) may be set individually by users, to get their own help files, for instance.

Notice the use of environment variables in the default values. This allows the names of files to depend on the values of other environment variables. It is useful to extend this approach to allow, for instance, customisation based on the terminal type by using the `$TERM` environment variable.

11.3 Variation of search order

The order in which `.xibatch` files are scanned, either to locate message files, or to read options, may be varied from the default if required.

A keyword, optionally placed in the Master configuration file `/usr/local/etc/gnubatch.conf`, in each case may be used to vary this.

11.3.1 Search order for message files

The default search order, for example with gbch-q, which searches for a file named in the variable `BTQCONF`, is to look:

1. As specified in a `.gnubatch` file in the current directory.
2. As specified in the environment variable of that name.
3. specified in a `.gnubatch` file in the user's home directory.
4. In a standard place, namely `btq.help` in the system help files directory, by default `/usr/local/share/gnubatch/help`.

This order may be respecified by assigning a "PATH" variable style value to the environment variable `GB_HELPPATH`, consisting of directory names separated by colons. The user's home directory may be denoted by "`~`" and any other user's home directory by "`~user`". Any environment variables are fully expanded.

If a directory name is relative (not starting with "`/`"), it is taken as being relative to the directory from which the program was started. The current directory should be represented as just "`.`".

Finally the environment is represented as a "`!`".

Thus the default search order is represented as:

```
GB_HELPPATH=".:!:~"
```

which causes the search to start in the current directory, then the environment, then in the home directory. Note that there is no way of suppressing the fallback to the standard location in the system help files directory.

It might be relevant for example, for a French user, with copies of the help files in a separate directory, to set a different search path, thus:

```
GB_HELPPATH=".:!:/usr/spool/gshelp/French"
```

This will affect all GNUbatch user-level programs, which search for a help file named by a keyword listed above.

11.3.2 Search order for program options

An almost identical system is used for program options, with the keyword `GB_CONFIGPATH` except that the search is in the opposite order. Also the additional symbol "-" is used for command-line options.

The default is therefore:

```
GB_CONFIGPATH="~:!:.:-"
```

Indicating the search first in the home directory, then the environment, then the current directory, and finally the command line options.

Note that it is possible to turn off the interpretation of command line options, if required, in this way, by omitting the "`-`", limiting the command line arguments to, for example, just job numbers for gbch-jchange etc.

Assignments to this will affect all GNUbatch user-level programs.

11.3.3 Freezing options

The `+freeze-home` and `+freeze-current` options in the user level commands and equivalent on-screen facilities in gbch-q and gbch-xmq etc are not affected by these options, the software does not attempt to "backwards-interpret" these paths. In particular please note that these facilities save all of the current values of these options regardless of whether they are default values or where they were read from, so you may want to "prune" the values of the options saved.

12 Extensibility

There are various mechanisms for enhancing or extending the functionality of GNUbatch, which go beyond customisation of the user interface. Some of these mechanisms are separate products with their own documentation. These are the 'C' programmer's API for Unix and the API for Windows PCs.

This chapter covers the facilities which are built into the basic product and the Motif GUI option as standard. They consist of hooks where GNUbatch can invoke custom built user, administrator and internal programs. Such programs are usually shell scripts, but compiled programs can be used just as easily.

12.1 Message Handling

It is possible to "intercept" the message handling of job completion messages.

There are two kinds of job completion messages, as well as the automatic mailing of standard output and standard error to the user unless redirected.

These are "mail completion message" and "write completion message to terminal". The latter case may vary, depending upon whether the originating host was a Unix machine or a Windows client.

Standard action is to invoke the mail program for "mail completion", btwrite to send messages to the user's terminal if on a Unix machine and dosbtwrite to send messages if on a Windows client.

The actual messages which are sent are generated by the internal program btmdisp, and this usually extracts the message texts from the internal help file `btint-config` in the system help directory `/usr/local/share/gnubatch/help`.

You can customise this in various ways:

1. You can customise the messages in the `btint-config` file itself.
2. Various users can specify their own `btint-config` file replacement.
3. Each of the 3 message sending programs can be reselected on a system-wide or individual basis.

12.1.1 Customising the system message file

All of the various mail or "write" messages are grouped together. There are variations according to whether the job terminating had an error or not, and a title or not.

Various `%t` etc parameters are inserted, the meaning should be reasonably clear from the context.

12.1.2 Specifying a customised message file

A user can specify a message file for his or her own use to replace the system message file, by including a line with the keyword `SYSMSG` in a `.gnubatch` file in his or her home directory. This takes the form

```
SYSMSG=alternative/file
```

If this is not an absolute location (i.e. starting with "`/`"), then the location will be relative to the home directory.

Only job completion messages need to be placed in the file designated thus. Only jobs owned by the user will use this file; other users will receive a message from the system file or from their own message file if specified.

If the specified file cannot be opened, then the system message file is reverted to.

12.1.3 Specifying alternative job completion - system wide.

The three job completion programs, the mail program, btwrite and dosbtwrite may be respecified by assignment to the environment variables `MAILER`, `WRITER` and `DOSWRITER` within the master configuration file, `/usr/local/etc/gnubatch.conf`.

The replacements may be shell scripts rather than programs.

Remember to use the "`:`" notation rather than "`=`" unless you want these assignments to be passed to Batch jobs.

For example:

```
MAILER:/usr/sbin/sendmail
```

Each program is passed a user name as an argument, and a message on standard input derived from the message file.

12.1.4 Specifying alternative job completion programs - per user

Alternative message programs may be specified by assignments to `MAILER`, `WRITER` and `DOSWRITER` within a user's `.gnubatch` file in his or her home directory.

Thus:

```
WRITER=/my/write/program
```

Note that the "`:`" notation is not recognised here.

If the program does not exist or fails, the message is silently lost.

All programs are run completely under the identity of the job owner.

12.2 Command Interpreters

By default, new GNUbatch installations have just one command interpreter set up, usually the Bourne shell. Additional command interpreters can be added. These can use the same shell programs as existing command interpreters, but with different parameters or just different names. The parameters include arguments, like `-s`, passed on the shell command line, nice values and GNUbatch load levels.

As well as the Bourne, Korn and "C" shells, any program which reads its commands from standard input may be set up as a command interpreter, such as perl and various SQL programs.

Some databases have command interpreters which will read their commands from standard input. The normal approach would be to write a shell script that invokes the database command interpreter with the name of a command file. If the command interpreter will read from standard input the "shell script wrapper" can be dispensed with.

Bespoke command interpreters can also be written. These can be compiled programs or shell scripts.

For example, a command interpreter could be written to `bypass` or `dummy_run` jobs. That is run the job but not execute any of the commands inside. The effect being to make it appear to the rest of the system as though the job ran and finished normally. This could be used for testing schedules of jobs at a simple level without doing any processing.

A shell script for such a command interpreter might look like this:

```
#!/bin/sh

echo Job bypassed
exit 0
```

Such a script could be expanded to check for job control variable operations that could invalidate testing a job schedule in this manner.

12.3 Custom Tools & Scripts

The command line programs for GNUbatch enable all job, variable, user and general information to be queried and / or modified. These can be built into new commands using shell scripts or used within user applications. Here are some ideas.

12.3.1 Custom Tools

A user may have an application which queries and performs updates on a database. Some of the activities, like handling telephone enquiries, obviously require immediate processing. Many tasks, such as changing a customers details and producing long reports, can be performed in the background or overnight.

The application could submit batch jobs to performs these activities, and query GNUbatch to find out how such jobs are progressing.

The actions that could be performed include:

- Getting a list of the batch jobs (submitted to run once and be retained) that have worked for a particular user or set of users. The application would use a command like:

  ```
  gbch-jlist -u $USER -F "%h %P#" | grep "Done"
  ```

  The option `-u $USER` restricts the display to the current user and the `-F` "string" option formats the output to contain just the title and progress fields. The output will look like this:

  ```
  prog_a         Done
  Tuesday_backup Done
  update         Done
  ```

- Submitting batch jobs to run after a specified time. The assignments and pre-conditions can also be used to handle other dependencies, such as
    1. Ensuring all database update jobs are completed before starting any query or report jobs.
    2. Stopping the back up jobs from starting until all others are done.
    3. The database access is likely to deteriorate if too many processes are accessing it at once. The variables can be used as semaphores to make sure the optimum number of jobs are using the database at any time.
- The gbch-q and gbch-xq tools can also be invoked from inside an application. So the application could invoke the tool with a restriction to show those jobs of relevance to the current part of the application being used.

12.3.2 Shell Scripts

An infinite variety of useful commands can be created using shell scripts. The activities described in the previous sub-section can just as easily be performed by stand alone shell scripts. Other simple examples include:

- Cancelling all batch jobs that match a certain criteria. For example test jobs may be in a queue called junk. If the script takes the queue name as an argument the command might be:

  ```
  canjobs junk
  ```

  and the script would be

  ```
  gbch-jlist -q $1 -F "%N" | while read JOB
  do
      gbch-jchange -C $JOB
  done
  ```

  This fragment of shell script could be written in a variety of ways, some of which would only take up one line. The form used above was chosen for clarity, rather than as a style guide.

- Viewing the output from a job which has stdout redirected to a file. If the script takes the job number as an argument the command might be:

  ```
  viewlog jobno
  ```

  and the script might be

  ```
  # Get the list of I/O redirections for the job.
  # Use tr to Separate each redirection out - one per line.
  # Pipe into sed to substitute the job id number for any
  # %d1 symbols in the path/file names. Then pipe into awk
  # to get the stdout filename.

  gbch-jlist -F "%R" $1 | \
  tr "," "\n" | \
  sed s/%d1/$1/g | \
  awk -F">" ' ! /^2/
  ```

```
&& ! /0/ { print $NF } ' \
read OUTFILE


#  If the stdout file exists then open it for viewing
#  in using view, the read only version of vi.

if [ -f "$OUFTILE" ]
then
      view $OUTFILE
fi
```

This is a cut down version of an actual script used on an X-Terminal which opened a separate window for `stdout` and `stderr`. If the job was running, the script ran the `tail -f` command in a terminal session or the vi editor if the job had finished. As a further refinement the background colour of the terminal session was set to reflect the jobs state - green = done, yellow = running and red = failed.

Useful shell scripts can usually be adapted to be run as macros from within gbch-q, gbch-xmq, gbch-user and gbch-xmuser. In the case of gbch-q and gbch-xmq they have to be modified to take a job number or variable name as their only argument from the command line. User administration macros can take a list of user names as their arguments.

To run the output file viewer the shell script may be used exactly as it is.

The example to cancel jobs in a particular queue can be modified to become a macro. A line must be added at the beginning to query the selected job, using gbch-jlist with the job number, to get the queue name. The result should look like:

```
Q_NAME=`gbch-jlist -F "%q" $1`
gbch-jlist -q $Q_NAME -F "%N" | while read JOB
do
     gbch-jchange -C $JOB
done
```

Once set up the user would run the macro, from within gbch-q or gbch-xmq, by selecting any one of the jobs in the required queue and then invoking the macro.

12.4 Macros for Interactive User Programs

Macro command facilities are incorporated in the commands gbch-q, gbch-user, gbch-xq, gbch-xmq and gbch-xmuser.

Up to 9 pre-defined macro commands can be set up for jobs and another 9 for variables in gbch-q and gbch-xmq. Similarly up to 9 pre-defined macro commands can be defined for users in gbch-user and gbch-xmuser. In addition all of these programs can run macros which are not pre-defined. In this case the programs prompt for the name of the macro to run.

Gbch-xq has a built-in mechanism to define and edit macro commands which is described in section 10.1.9.

The macro does not have to be anything to do with the context of jobs, variables or users. It will, however, always be passed the identity of the currently-selected item or user as a parameter or parameters.

The actual commands are placed in the relevant help files, which by default are `btq.help`, `xmbtq.help`, `btuser.help` and `xmbtuser.help` in `/usr/local/share/gnubatch/help`. For the Motif programs, the names on the menu options are specified in the `GBATCH` file.

12.4.1 Inserting the commands.

First specify the macro commands in the help file as follows:

```
27000P:Prompt for prompted command
27001P:cmd1
27002P:cmd2
27003P:cmd3
27010P:Prompt for prompted command
27011P:cmd1
27012P:cmd2
```

The `2700nP` prompts specify the pre-defined macros, where n is in the range 1 to 9, for jobs in gbch-q, gbch-xmq and users in gbch-user, gbch-xmuser. The 27000P entry specifies the prompt used for job / user macros which are not pre-defined.

Pre-defined Macros for variables are specified by prompts of the form `2701nP`, where *n* is in the range 1 to 9. The `27010P` entry specifies the prompt used for variable macros which are not pre-defined.

The command involved is any arbitrary shell command which may take a jobs, variables or list of users as the case may be. (The command may be given zero users by gbch-xmuser and gbch-user if none are selected).

gbch-q and gbch-user assume that all macro commands run silently without producing any output on the screen or interacting with the user. If a macro needs access to the screen and or keyboard, put an exclamation mark immediately following the colon in the specification. For example:

```
27001P:!cmd1
```

12.4.2 Menu Options in the Motif Programs

Menu options are automatically created in gbch-xmq and gbch-xmuser. The names given to the options can be set by editing the relevant lines in the resource file. The menu options for job macros in gbch-xmq are specified in lines of the form:

```
gbch-xmq*jobmacro*macro1.labelString: option text
```

Here `macro1` indicates that this is the definition for the first option in the jobmacro menu. It will run macro 1 as set up by the prompt definition for `27001P:` in the `xmbtq.help` message file. The option text specifies the text to appear in the menu for that option.

For example, the following lines:

```
gbch-xmq*jobmacro*macro1.labelString: View job output
gbch-xmq*jobmacro*macro2.labelString: Delete job output
```

will produce a menu looking like this

The equivalent lines for specifying variable macros in gbch-xmq and user macros in gbch-xmuser are respectively:

```
gbch-xmq*varmacro*macro1.labelString:
gbch-xmuser*menubar*macro1.labelString:
```

The resource files used by Motif are not quite as flexible in scope as the GNUbatch message files. Copies of the `GBATCH` file (holding GNUbatch resources) can probably only be set up globally and one alternative for each user in their home directory. The lines shown above can be tailored on invoking gbch-xmq or gbch-xmuser using the `-xrm` option. This is explained in the options sections for the relevant programs.

12.4.3 Binding the keys in gbch-q and gbch-user

To use macros in gbch-q and gbch-user it is necessary to specify what key to press to invoke the macro. This is often called binding the keys.

To bind the keys, assign key codes 600 to 609 in the relevant message file. For example:

```
K600:0
K601:\kF1
```

will assign the "0" key to the prompted-for command and F1 to the first "pre-defined" command. In gbch-q this will assign the same key to the first "pre-defined" command for both the variable and job macros.

To assign different keys for job and variable macros, use the state keys by prefixing with the relevant state number. This is 1 for the job screen and 2 for the variable screen. For example to define five job macros and one variable macro using consecutive function keys:

```
1K601:\kF1
1K602:\kF2
1K603:\kF3
1K604:\kF4
1K605:\kF5

2K601:\kF6
```

12.4.4 Example - Adding the "cancel all jobs in queue" to gbch-q

The shell script, described earlier, for cancelling all the jobs in a particular queue is an ideal candidate for a macro. To recap, the shell script is called `canjobs` and it contains the these commands:

```
Q_NAME=`gbch-jlist -F "%q" $1`
```

```
gbch-jlist -q $Q_NAME -F "%N" | while read JOB
do
      gbch-jchange -C $JOB
done
```

Cancelling a job is normally assigned to lower case `z`, so it might be helpful if cancelling all jobs in a queue is assigned to upper case `Z`. This command should also be added to the help message for the job screen. The following extracts show the changes and additions, in bold type, to the gbch-q message file. The dotted lines indicate omitted lines of text.

```
       ............
# Keys for displaying job list

H1:?            Help (this screen)
       ...
H1:P            Reset progress code
H1:r z Z        Set runnable, cancelled , all in queue cancelled
H1:g f          Run if possible ignoring time adv time/no adv time
       ...
1K523:r
1K524:z         (this is the key assignment for set cancelled)
       ...
1K532:a
1K540:\,

1k601:\KF1
1K602:Z         (this is the key assignment for the new macro)
       ............

# Example macro commands
# Jobs are 27000 + n, variables 27100 + n n is 0 to 9, 0 prompts

27000P:Run what:
H27000:Please give the command you want to run.
H27000:The job number will be given as a parameter.
27001P:!viewlog
27002P:canjobs
27100P:Run what:
H27100:Please give the command you want to run.
H27100:The variable name will be given as a parameter.
```

The first macro defined in the prompts is the viewlog script described earlier. It is obviously an interactive macro, hence has the exclamation mark "`!`"after the colon "`:`" to tell gbch-q to release the screen and redraw it afterwards.

12.5 File & Event Monitoring

Note: This section is largely superseded by the introduction of gbch-filemon, but is left in as an example of how various operations may be done.

An event can be detected by polling for a change in state of the object affected by the event. For a batch processing schedule, the most likely events of interest are file creation, modification or deletion. Two of the benefits of polling for a specific event, rather than intercepting all events, are:

1. Machine resources are only used for the events of interest. E.g. only the relevant file or files are being checked.
2. An event need only be polled for when it is of interest. E.g. a job may need a particular file as a pre-requisite, but there is no point polling for the file until the job's scheduled run time.

Two suggestions follow, for file monitoring, which can be adapted to any event which can be tested for by some change in state.

12.5.1 Polling for Arrival of a File

A simple shell script can be used to look for a file arriving, then setting a job control variable and exiting when it is found. There are 4 parameters that can be specified from the command line to provide a general purpose script. They are: filename (`$1`), time interval between tests (`$2`), variable name (`$3`) and value to write into variable (`$4`).

```
#!/bin/sh
# This example tests for a file $1 existing
# every $2 seconds. When the file is found it
# sets the contents of variable $3 to be the
# value $4.

while [ ! -f $1 ]
do
    sleep $2
done

gbch-var -s $4 $3
```

12.5.2 Continuous Polling for a constantly changing list of Files

If a large number of files are being monitored then it is probably more efficient to have one process doing the polling. The list of files could be specified in a file loaded by the file monitor program when it starts. This would mean however that the program had to be restarted whenever the list of files changes.

The job control variables can be used to hold the list of files to poll for as well as returning the status. In this example the variables conform to this set of conventions:

- Names of variables involved in file monitoring all end with the string `_FILE`, for example `UserAdmin_FILE`, `customer_FILE`.
- The comment field must contain the full path and name of the file to look for.
- Variables must contain the value "`Waiting`" when the file they represent is to be polled for.
- The value of variables will be set to "`Ready`" when their file has been detected.

The script looks like this:

```
#!/bin/sh

while :
do
    gbch-vlist -F "%N %V %C" | grep '_FILE$' | \
    while read NAME VALUE COMMENT
    do
        if [ "$VALUE" ="Waiting" ]
        then
            if [ -r $COMMENT ]
            then
                gbch-var -s "Ready" $NAME
            fi
        fi
    done
    sleep 120
done
```

Parameters which are hardwired into this example, like the number of seconds for the sleep command, could be supplied as arguments to the script file monitoring.