Xi Software

GNUbatch Release 1

Introduction and User Guide

This manual is for GNUbatch (Introduction and User Guide).

1 What is GNUbatch?

GNUbatch is a *job scheduler* to run under Unix and GNU/Linux operating systems. It executes jobs at specified dates and times or according to dependencies or interlocks defined by the user.

Schedules of jobs may be run on just one processor, or shared across several processors on a network with network-wide dependencies. Access to jobs and other facilities may be restricted to one user or several users in a group as required.

Control of and access to all the **GNUbatch** facilities is made available through a variety of interfaces including:

- Command line or "shell commands"
- Full screen text-based using "curses"
- X-Windows (Motif toolkit)
- X-Windows (GTK toolkit)
- MS Windows clients
- Web Browser interfaces (two variants)
- Unix or GNU/Linux based API (supporting C and C++)
- MS Windows based API (supporting C and C++)

In all cases apart from "command line" the access is fully interactive and the display is automatically updated to take account of changes to jobs and other facilities on the screen whether by the user, another user or by jobs starting and finishing.

A lot of effort has been made to make **GNUbatch** fully configurable on a per user or system-wide basis. You can alter the format and content of screen displays, help messages, command keystrokes and option names to suit your requirements such as foreign language support or personal taste.

Common sets of options and parameters may be saved by the user to provide a local or global set of defaults for particular tasks.

GNUbatch is implemented so as to minimally impact upon the system. In an idle state, it typically uses less resources than **cron**. It is implemented in ANSI C, apart from the MS Windows client, which is in C++, and parts of the web browser interfaces, which are in JavaScript. All of the product was developed and written by Xi Software. No additional third-party database or other software is used or relied upon.

All current Unix and GNU/Linux platforms are supported and different platforms may run the product concurrently inter-networking with each other.

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 may refer to Xi-Batch rather than GNUbatch).

2 Jobs

The basic unit of **GNUbatch** operation is a *job*. A job consists of two parts:

- A list of scheduling parameters, conditions and a name of a program to run.
- A (possibly empty) *script* to be passed to the program specified by the other part. This is usually a text file, but it does not have to be.

In the simplest cases (in many installations the only case) the program run by the job is the Bourne shell and the script is a shell script, possibly including arguments supplied by the job parameters.

A variety of interfaces are provided to create jobs, to edit them once submitted, and to delete them, possibly retaining a copy. Such interfaces may be invoked from within other jobs.

Jobs may be created on one host on the network, queued on another and executed on yet another.

The program which is run, and to which the script is passed, is called the *command interpreter*. Within reason, it may be any program, but is most often a shell such as the Bourne Shell sh.

3 Variables

There is a second kind of object maintained by the **GNUbatch** system, called variables.

Variables have a name and a value, which may either be a string or an integral numeric value. There are a separate set of commands and interfaces to create, delete and assign new values to variables. Variables may be "exported" or shared between several hosts on the network.

An important feature of variables is that updates are interlocked across the network, so that if two or more users or jobs anywhere on the network attempt to update the same variable from the same starting value at the same time, only one will "succeed".

The main function of variables (although the synchronisation features enable them to be used for other purposes as well) is to provide starting "conditions" for jobs. Jobs may be set so that they will only start if one or more variables are set to certain values.

Jobs can also be set so that as they start or as they finish they will "assign" values to variables, so possibly releasing other jobs to run. It is possible to distinguish between normal and error cases of job termination and handle them differently.

4 Queues

There is a single job queue as such on each machine. However a job can be given a "queue name" as a prefix to its title. It is also possible to group sets of options and parameters together with the queue name and achieve the effect of a logical queue with standard sets of parameters. However more flexible ways of organising and controlling sets of jobs are possible.

5 Getting Started

As an initial example, let us look at a very simple job submitted from the shell prompt. The shell-level command to do this is called gbch-r.

```
$ gbch-r
Warning: Expecting Input from Terminal
echo hello
sleep 30
(ctrl-D)
$
```

In this simple example, the batch job consists of two commands echo and sleep. **GNUbatch** will attempt to execute them as soon as possible (this may well be immediately). You will probably get a mail message like the following:

```
From batch Tue Feb 6 11:45:57 2001
Date: Tue, 6 Feb 2001 11:45:56 GMT
From: GNUbatch Scheduler <batch@example.com>
To: guest@example.com

Subject: GNUbatch job number 27309 completed

Your batch job, job number 27309, completed satisfactorily
Standard output from your job was:
---------------------------------
hello
```

Please note that the job scheduling program gbch-r displays a warning message if input from the terminal is expected. It is to be terminated by the user's end-of-file character, normally *ctrl-D*.

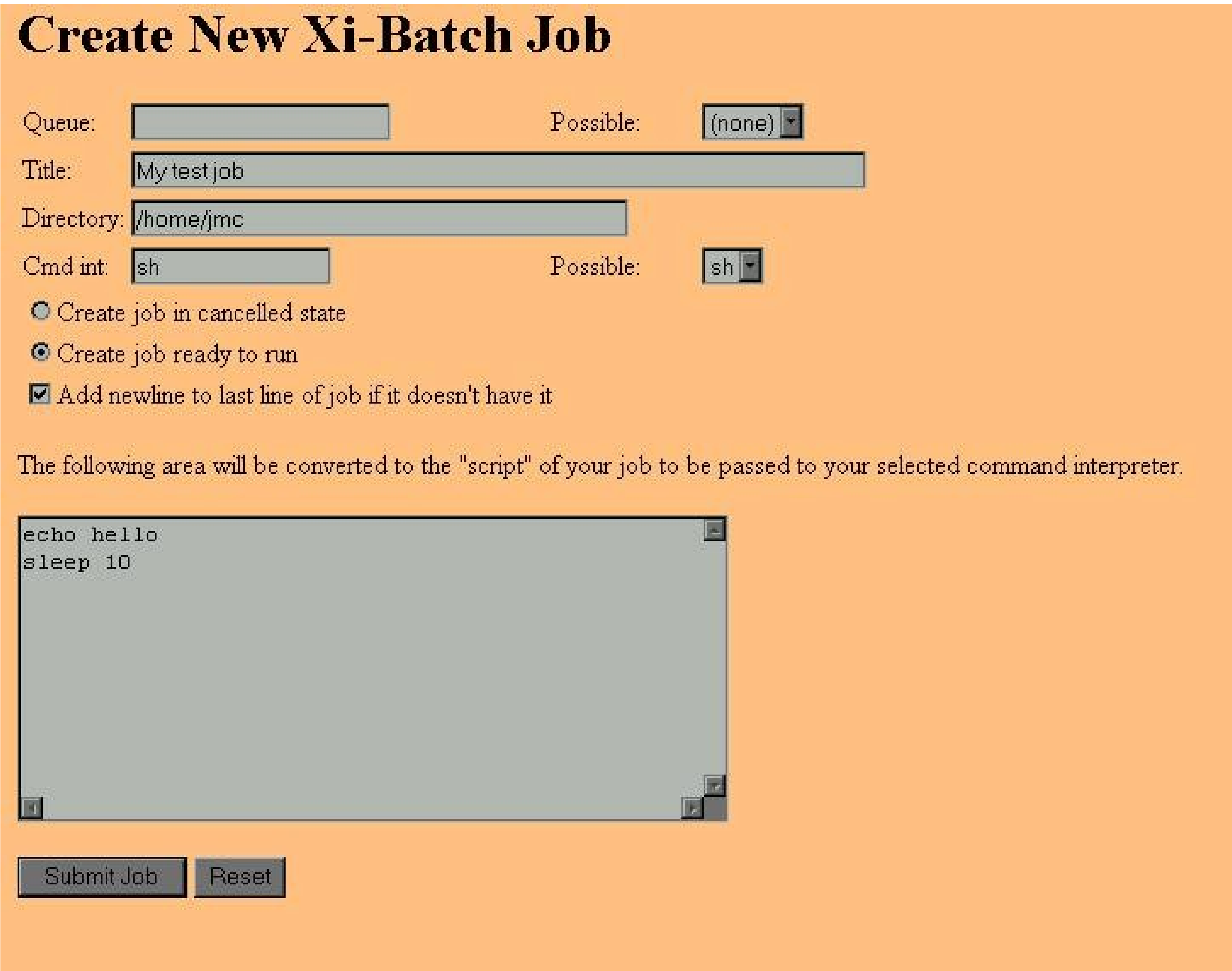
The actual input is passed to the *command interpreter*, which usually defaults to the Bourne shell (sh). The input shell commands may be taken from *Standard Input* as in the above example. Alternatively, they may be taken from one or more job files as follows:

```
gbch-r job1 job2
```

The files `job1` and `job2` each consist of a "shell script" to be executed and a separate job will be created for each file (if no "job title" is given, they will be given titles taken from the file names).

A large number of options may be given to gbch-r to set numerous parameters for the job. In many cases it is easier to set them all interactively, which may be done with the graphical submission programs gbch-xr or gbch-xmr (for X-Windows) or btrw (for Microsoft Windows).

You can also submit jobs from a web browser using the GNUbatch web interface, for example to submit a job equivalent to the above you could invoke it from the browser thus:



Create New Xi-Batch Job

Queue: Possible: (none)

Title: My test job

Directory: /home/jmc

Cmd int: sh Possible: sh

○ Create job in cancelled state

● Create job ready to run

☑ Add newline to last line of job if it doesn't have it

The following area will be converted to the "script" of your job to be passed to your selected command interpreter.

```
echo hello
sleep 10
```

(An alternative mode of browser job submission allows the user to pass a file of commands to the web server rather than typing in the script as in the above example).

Often users just submit the basic job to the queue, holding it for the moment, and then adjust the various parameters using one of the various job queue managers. To do this, the job or jobs are submitted using the "`-C`" (cancelled state) option, thus:

```
gbch-r -C job1
```

This is the usual method of submission for the web browser interface.

Most of the examples given in this manual are given in this state.

6 Job Queue Management

Once jobs have been placed on the job queue, there are various ways of displaying and updating them.

You can display the job status from the command line using the utility program gbch-jlist:

```
$ gbch-jlist
1119 jmc job1 sh 160 1000       Run
1137 jmc job3 sh 150 1000 12/02
$
```

As is conventional with Unix-style commands, without options this is a terse list of unexplained fields, however an option may be used to put a heading on each column:

```
$ gbch-jlist -H
Jobno User Title Cmd Interp Pri Loadlev Time  Cond Progress
1119  jmc  job1  sh          160    1000            Run
1132  jmc  job2  sh          150    1000 08/02
1137  jmc  job3  sh          150    1000 12/02
$
```

However there is a "remember my defaults" option for all commands which enables you to say that you always want the "`-H`" option unless you specify something different.

You can also tune which attributes of jobs are displayed and in which order and remember those as your defaults.

gbch-jlist by default just displays jobs local to the host, but you can display other hosts' jobs by using the "`-R`" option (which can likewise be saved as your own personal default):

```
$ gbch-jlist -HR
Jobno           User Title   Cmd Interp Pri Loadlev Time   Cond Progress
01119           jmc   job1   sh          160    1000             Run
avon:24918      jmc   newone sh          150    1000             Canc
01132           jmc   job2   sh          150    1000 08/02
01137           jmc   job3   sh          150    1000 12/02
avon:28156      jmc   newtwo sh          140    1000             Run
$
```

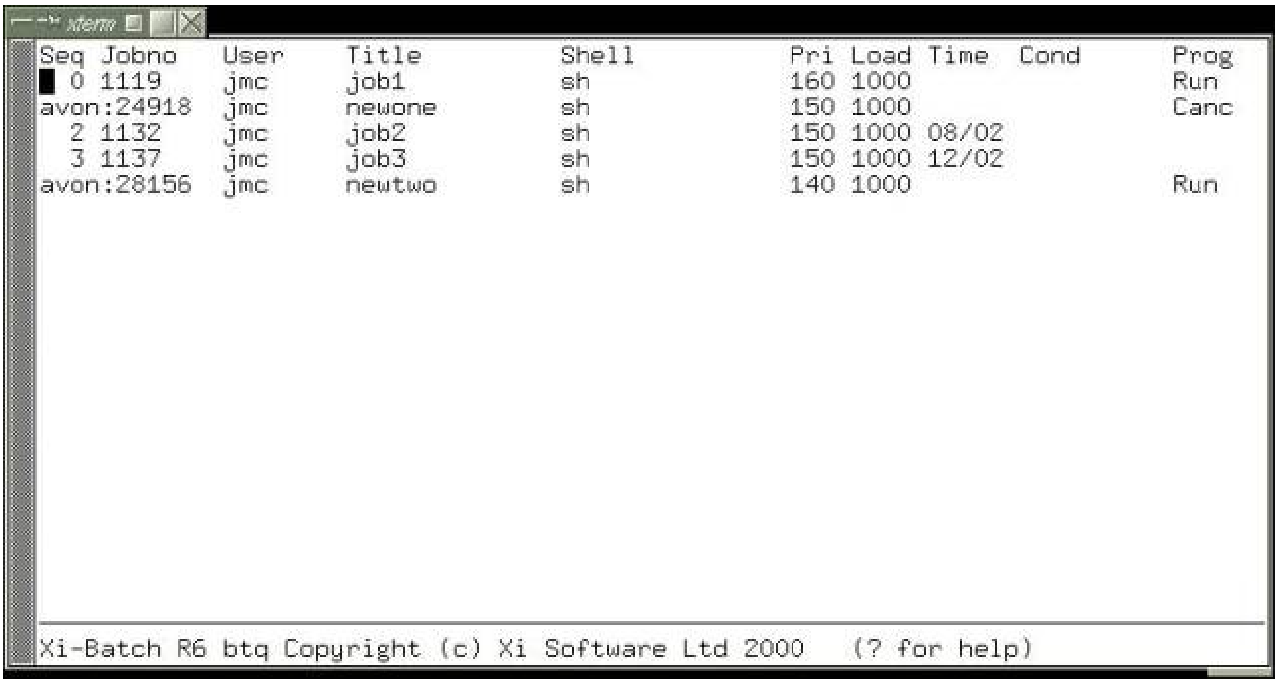
Note that the external jobs have the host name (`avon` in this case) prefixed to the job number. In all other respects, the external jobs can be displayed and manipulated in the same way as local jobs.

All this is a bit inconvenient, as to manipulate the jobs you have to quote the job numbers in an appropriate shell command, so most of the time you may want to display the job list and update attributes of jobs using one of the four interactive job queue managers.

A key feature of the interactive job queue managers is that the display is instantly updated when any changes occur, whether as a result of your activities, another user's, or merely because jobs start and finish.

6.1 Btq - character based terminals

An example display of the job queue listed above would be as follows:



```
Seq Jobno   User    Title         Shell          Pri Load Time  Cond      Prog
█ 0 1119    jmc     job1          sh             160 1000                 Run
avon:24918  jmc     newone        sh             150 1000                 Canc
  2 1132    jmc     job2          sh             150 1000 08/02
  3 1137    jmc     job3          sh             150 1000 12/02
avon:28156  jmc     newtwo        sh             140 1000                 Run

Xi-Batch R6 btq Copyright (c) Xi Software Ltd 2000   (? for help)
```

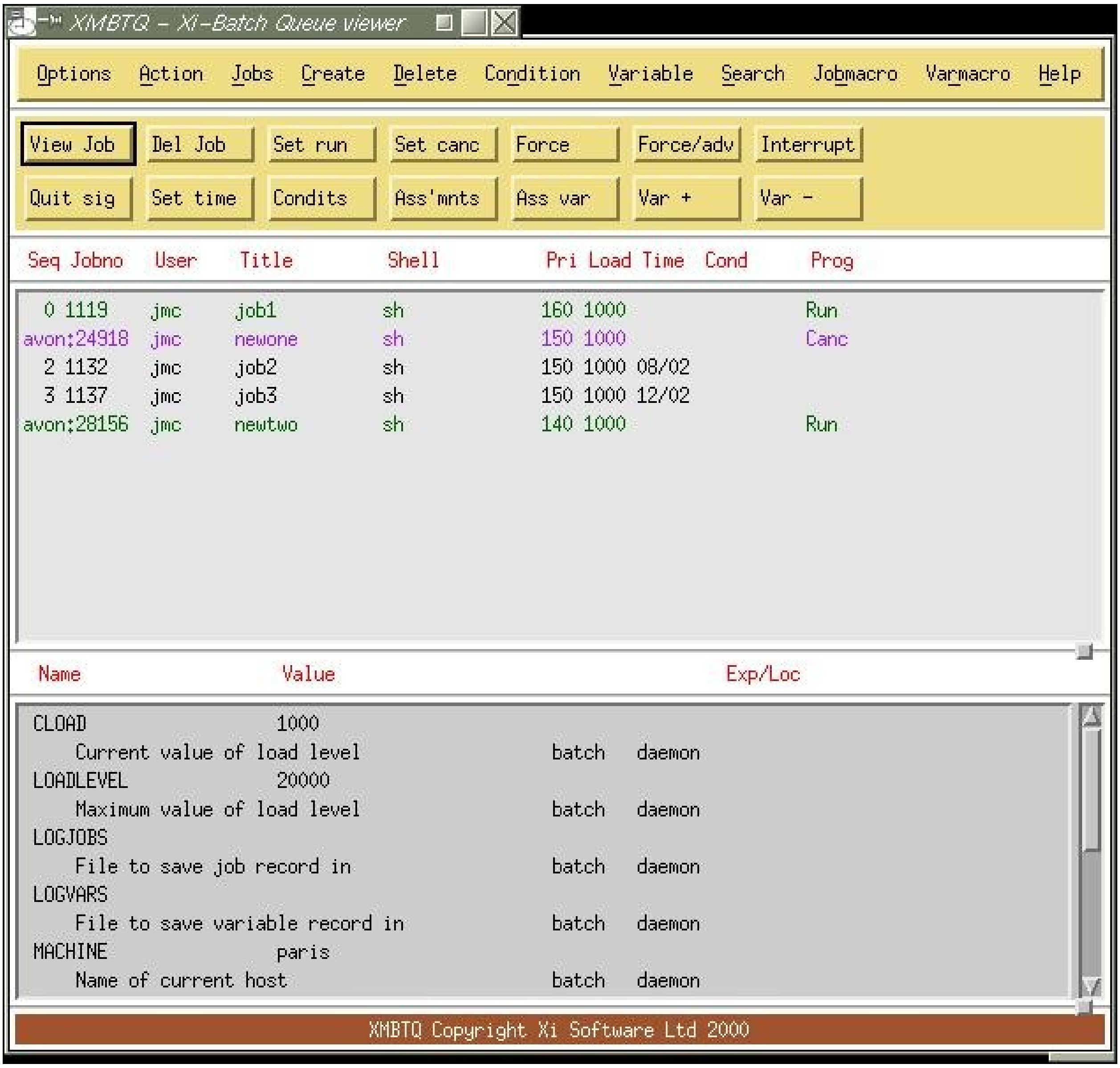
This is the default display format, which again may be varied by the user and the formats saved as her or her own personal defaults.

Various keystroke commands enable the user to select jobs and make changes to various parameters. These are defined further in the reference manual.

Note that all the text strings such as Run and Canc may be changed by the user to reflect his or her own terminology. Likewise the command keystrokes may be changed.

6.2 Xmbtq - X-Windows

The same job queue would look like the following under the X-Windows interface:



```
Seq Jobno    User     Title          Shell          Pri Load Time  Cond      Prog
  0 1119     jmc      job1           sh             160 1000                 Run
avon:24918   jmc      newone         sh             150 1000                 Canc
  2 1132     jmc      job2           sh             150 1000 08/02
  3 1137     jmc      job3           sh             150 1000 12/02
avon:28156   jmc      newtwo         sh             140 1000                 Run
```

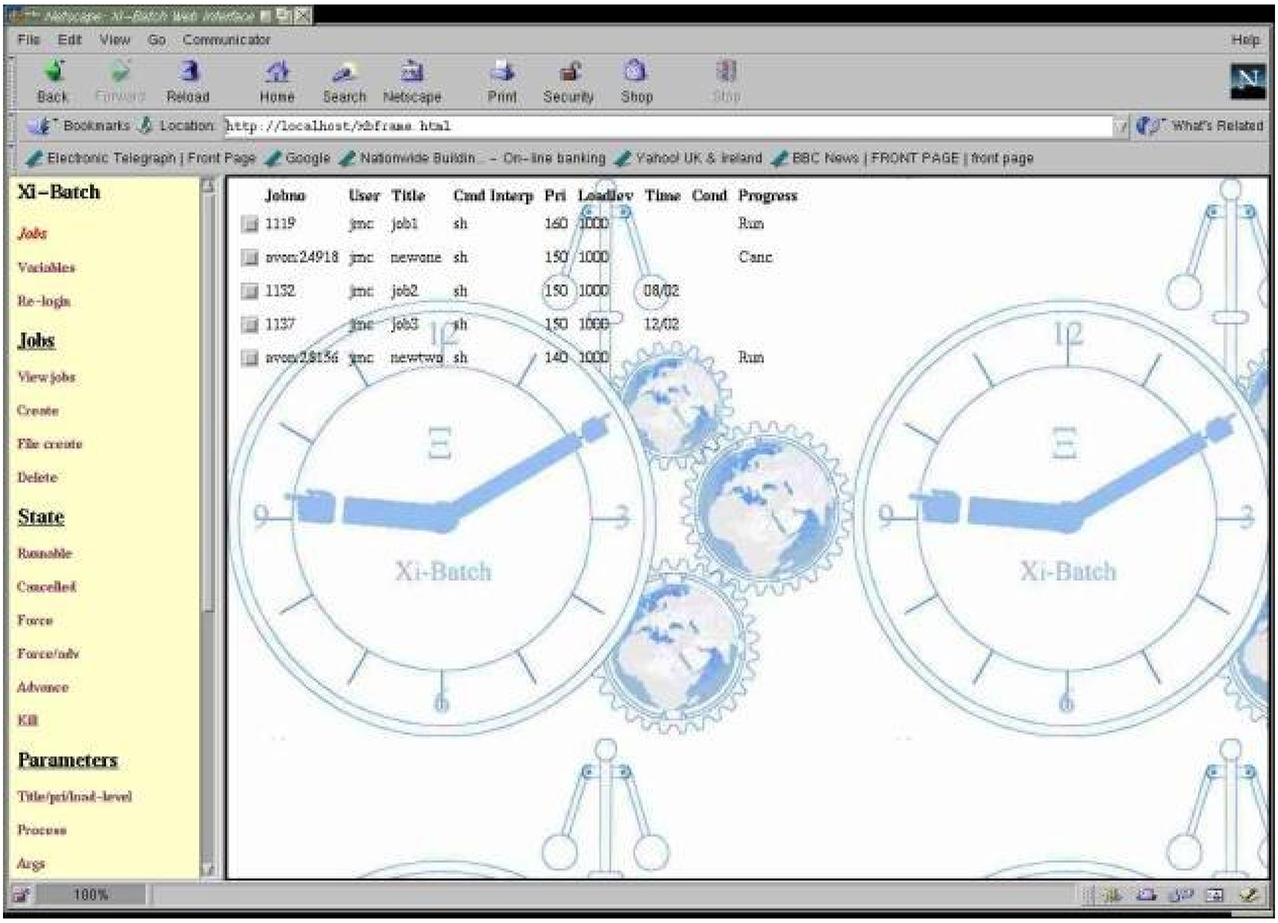
```
Name                     Value                                  Exp/Loc
CLOAD                    1000
    Current value of load level                  batch    daemon
LOADLEVEL                20000
    Maximum value of load level                  batch    daemon
LOGJOBS
    File to save job record in                   batch    daemon
LOGVARS
    File to save variable record in              batch    daemon
MACHINE                  paris
    Name of current host                         batch    daemon
```

Note how the jobs may appear in different colours depending on the state they are in. The colours used may again be selected by the user.

This display is split between jobs and variables. The split may be adjusted by the user as required.

6.3 Web Browser Interface

The following is the same job display as shown by the web browser interface (Netscape Navigator was used in this example).



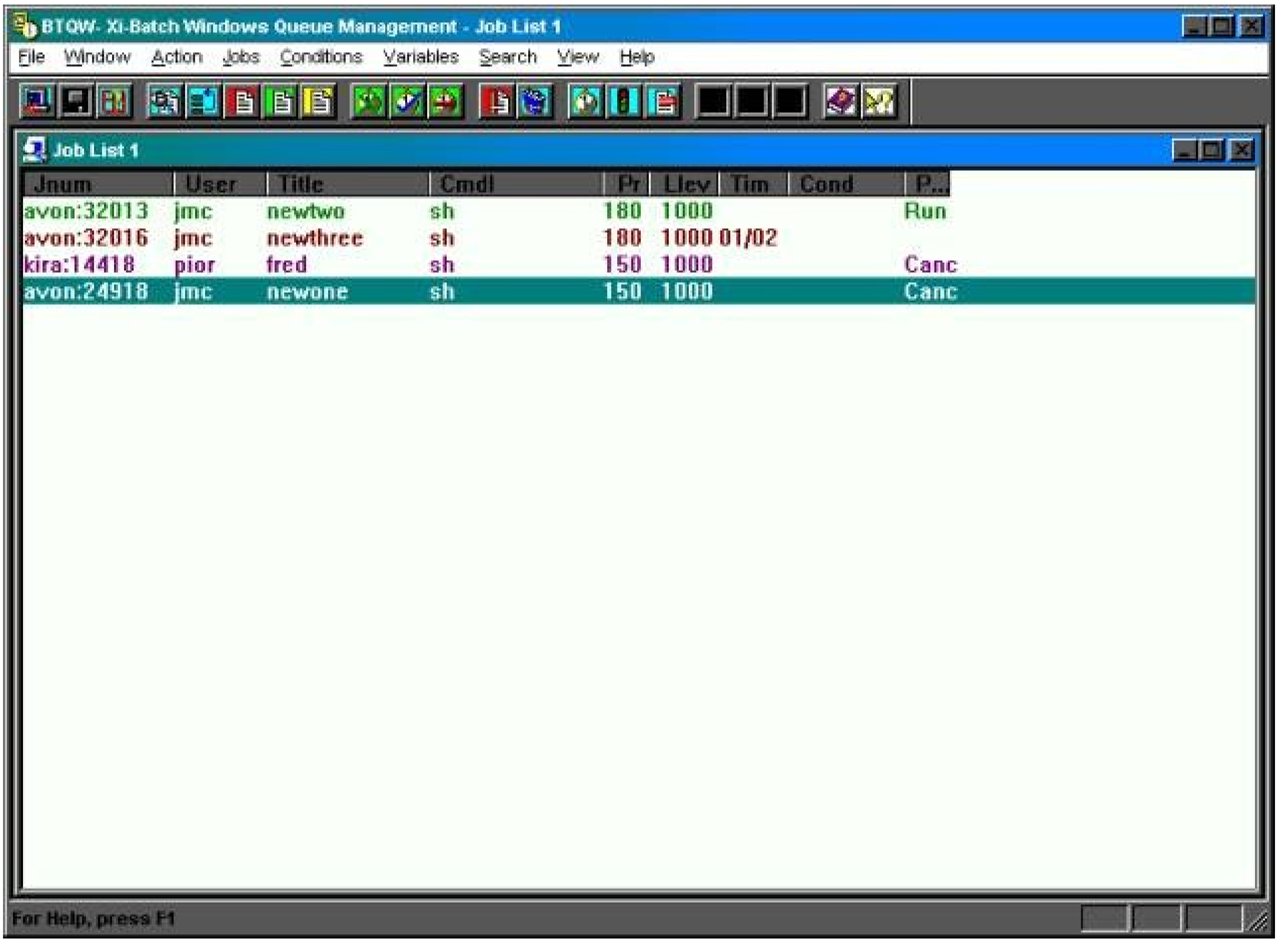
	Jobno	User	Title	Cmd Interp	Pri	Loadlev	Time	Cond	Progress
☐	1119	jmc	job1	sh	160	1000			Run
☐	avon:24918	jmc	newone	sh	150	1000			Canc
☐	1132	jmc	job2	sh	150	1000	08/02		
☐	1137	jmc	job3	sh	150	1000	12/02		
☐	avon:28156	jmc	newtwo	sh	140	1000			Run

The check boxes at the start of each job row may be used to select one or more jobs and the options on the frame on the left used to perform various operations.

Again the display formats and job attributes to be displayed may be selected by the user and stored as a "cookie" for the next time the job list is accessed.

6.4 MS Windows Interface

The final interface style is the MS Windows client. A set of jobs similar to the jobs displayed above might look like the following


BTQW- Xi-Batch Windows Queue Management - Job List 1
File Window Action Jobs Conditions Variables Search View Help
Job List 1
For Help, press F1

Jnum	User	Title	Cmdl	Pr	Llev	Tim	Cond	P...
avon:32013	jmc	newtwo	sh	180	1000			Run
avon:32016	jmc	newthree	sh	180	1000	01/02		
kira:14418	pior	fred	sh	150	1000			Canc
avon:24918	jmc	newone	sh	150	1000			Canc

Each host is separately identified.

Currently **GNUbatch** does not run as such on MS Windows, but there is an Agent program which may be invoked by **GNUbatch** on a host which may be used to upload and download files and run jobs on the Windows machine.

6.5 API

If none of those interfaces do what is required, and you want to devise your own, there is an API available for C and C++ under Unix and GNU/Linux, and also under MS Windows. All of the facilities described above are available using the API.

7 Load Levels

Load levels provide a simple method of controlling the number and types of jobs which may be run at once.

Each job has a value associated with it called the load level, which is usually derived from the command interpreter used. Suitably privileged users may vary this up or down for individual jobs as required.

There is a system-wide maximum load level, and a total for each user. A job will not be started on a given machine if the job's load level would cause the system-wide maximum to be exceeded or the user's total to be exceeded. However it may be possible for another machine to run the job if its values for those totals are not exceeded.

This provides for a finer level of tuning than just a maximum number of jobs would give. So if the maximum load level is 6,000, then 6 jobs of load level 1,000, 3 jobs of load level 2,000 or some combination coming to 6,000 or less can be accommodated.

8 Time Settings

The examples discussed earlier are set to run "as soon as possible". Depending upon the load level, this may be more or less immediately.

In many cases, a "time to run" is set, giving the earliest time and date at which the job is to be run.

In addition a "repeat specification" can be set, indicating whether the job is to be repeated and how the repeat time is to be calculated.

The repeat specification may be "run once and delete", "run once and retain" (so the job has to be manually reset), or repeat in units of so many:

- Minutes
- Hours
- Days
- Weeks
- Months
- Years

In the case of months, the specification provides for "relative to the beginning", in which case the same target day, such as the 1st or the 5th is selected each month, or "relative to the end", in which case so many days from the end of the month are selected.

At the same time you can say that you want to set "avoid days" such as weekends, and set up a calendar of holidays and avoid those days. A later day is selected in all cases except in the case of "months relative to the end", when an earlier day is selected. This makes it easy to say you want to repeat something on the "second to last working day of each month", taking weekends and holidays into account.

The next time to run is replaced by the time calculated from the repeat specification at the end of a successful normal run.

There are two operations to override the time of a job if required, "force" which performs an extra run of the job as soon as possible, and "force and advance" which performs the next run of the job now, advancing the time.

Another option specifies what should happen if the job cannot be run at the required time, whether it should be repeated or run at the next available time, together with other options.

9 Conditions and assignments

One of the key features of **GNUbatch** is the ability to control the execution of jobs, so that, for example, no more than one job of a given type may be executed, or a given series of jobs may be executed in a given order.

To achieve this facility, **GNUbatch** variables are used. Any number of variables may be created, and any user may create variables, provided that the names do not conflict with existing variables.

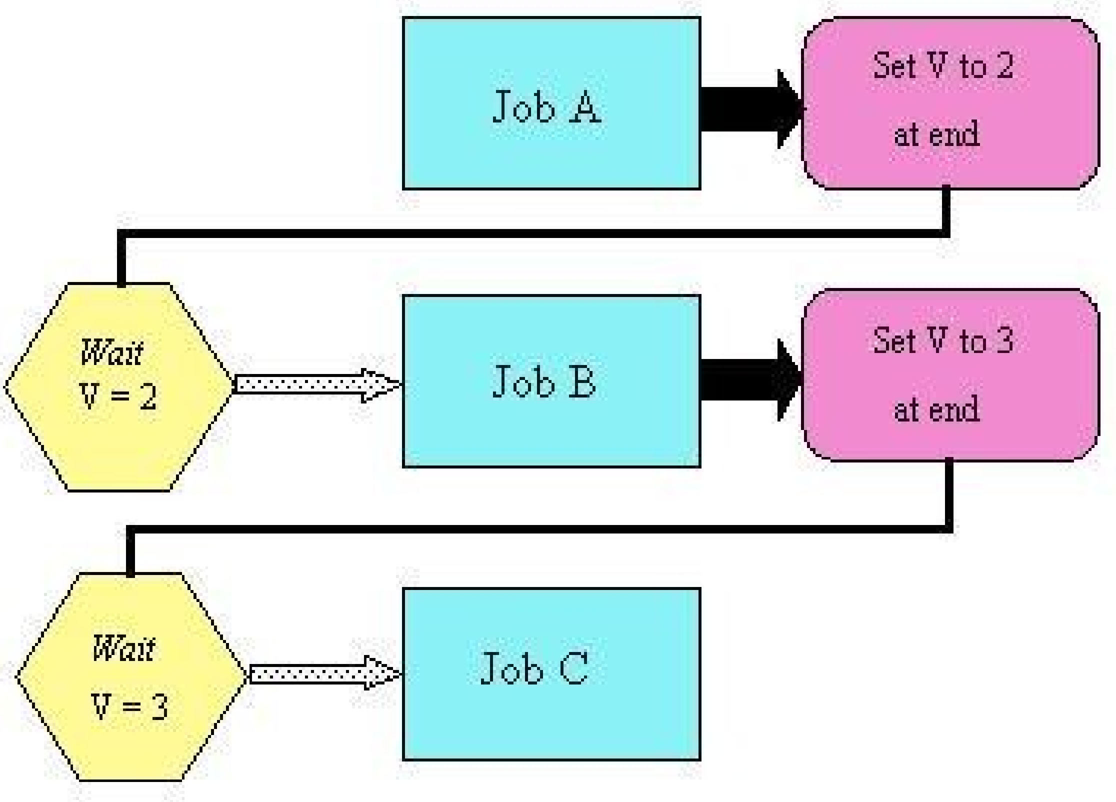
Variables may be given a numeric or a string value, and a comment to explain to other users the purpose of the variable.

Before a job starts, it is possible to check that up to 10 variables possibly on remote machines have particular numeric or string values.

When a job does start, it is possible to change the values of up to 8 variables, possibly on remote machines.

When a job finishes, it is likewise possible to change the values of up to 8 variables, possibly by reversing what happened when the job started.

The type of facility this is describing might be illustrated thus:



As an extra feature, you can separately distinguish the cases where the job ends due to

- Normal termination.

- Error termination (with exit code non-zero, or you can redefine what constitutes error exit).
- Abort (signal or core dump etc).
- Cancellation - operator halt before the job has a chance to run.

There are various ways of creating and initialising variables, either from the command prompt with btvar, or from the various full-screen and graphical interfaces or API.

As with jobs, variables may be listed and displayed using the shell command btvlist, analogous to gbch-jlist, and with similar options, for example

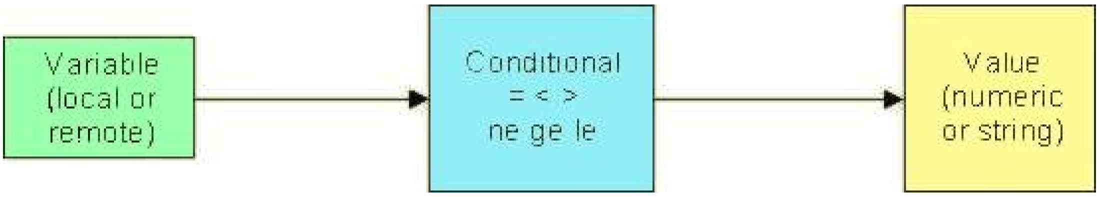
```
CLOAD           1000                # Current value of load level
LOADLEVEL       20000               # Maximum value of load level
LOGJOBS                             # File to save job record in
LOGVARS                             # File to save variable record in
MACHINE         avon                # Name of current host
STARTLIM        5                   # Number of jobs to start at once
STARTWAIT       30                  # Wait time in seconds for job start
torres:fooo1    128     Export      # Try this for export
fred9           23                  #
newvar1         333                 #
tryzzzz         3444    Export      #
zz1             445                 #
```

Certain of these variables are special and provide scheduling options for the system. Any or all of them may be used to control jobs.

Note how the variables on remote machines have the machine name prefixing the name of the variable.

9.1 Conditions on jobs

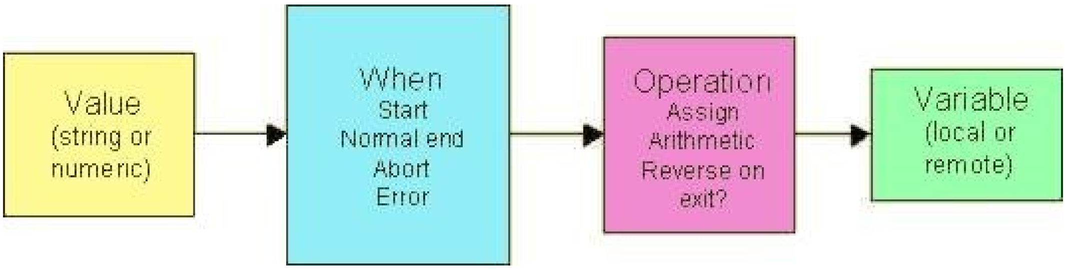
A job condition consists of the following:


Variable (local or remote)
Conditional = < > ne ge le
Value (numeric or string)

A condition involving a variable on a remote host may be marked as "critical". In such a case the job will be blocked if the remote host is unavailable. Otherwise the condition will be ignored.

9.2 Job Assignment

A job assignment consists of the following



In the same way as for conditions, assignments to remote variables may be "critical". If they are "critical", a job will not start if the referenced variable is on a machine which is not available. (If it starts and the machine becomes unavailable whilst the job is running, obviously not much can be done).

It is possible to set up and modify conditions and assignments from any host or client using any of the various interfaces.

9.3 Conditions on Files

It is possible to monitor for a file or set of files being created or changed and to make a job dependent upon that.

This is performed using a separate file monitoring process. The reason for doing this is that there is no option in Unix to set an alarm to "wake me up when a file changes" and consequently the monitoring process has to "poll" or look at the files at regular intervals.

One of the key design features of **GNUbatch** is that wherever possible the software is "event-driven", in other words the software does not use processor time and memory when there is nothing to do.

However this commonly-required facility is provided as a standard but stand-alone option to the software. There is no actual requirement for **GNUbatch** jobs to be submitted or released by the file monitoring option, any task may be performed.

10 Permissions and security

A key feature of **GNUbatch** is the careful attention which has been paid to ensure that no unauthorised access is made its facilities.

Each job and variable has an *access mode* defining which operations various users may perform upon the file, including changing the access mode.

Each user has a set of permissions defining his rights of access to special facilities of **GNUbatch** and a set of default access modes which apply to jobs and variables which he or she creates, which may or may not be departed from, according to other permissions.

The access rights and permissions for each user are controlled by various special utilities and apply to the machine upon which the user is attempting to work.

The Web browser interface requires a correct user name and password to be quoted before access rights are granted (although there is an option to set a default user, usually with limited access rights).

11 Charges and logging

A log is kept of each user's use of the system each time a job is run on behalf of that user and this can, if required, be the basis of accounting or logging of machine usage.

Additionally, a central log can be kept of each operation on each job or variable, showing the time, date, nature of the operation and the user responsible.

12 Conclusion

This manual is intended to give you a brief overview of **GNUbatch** and some idea of its facilities and what it can be used for.

To install and administer the product, please see the *Administration Manual*. For details of the individual parts of the software, please see the *Reference Manual*. Some of the options, such as the Windows client option, are described in separate manuals.

The product is under continuous development and enhancement and we greatly welcome constructive suggestions for this.