

# Xi Software

## **GNUspool Release 1** Installation and Administration Manual



## GNUSpool Administration Manual

This manual is for GNUSpool (Installation and Administration Manual).

Copyright 2008 Free Software Foundation, Inc.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license is included in the GNUSpool reference manual in the section entitled ``GNU Free Documentation License''.

## Table of Contents

1	Introduction.....	1
1.1	Documentation Standards and Program Usage.....	1
1.2	Command Line Program Options.....	2
2	Installation.....	3
2.1	Installation directories.....	3
2.2	Installing the system user.....	3
3	Defining and installing printers.....	4
4	Setting up networking.....	5
4.1	Adding a Unix host.....	5
4.2	Adding a fixed IP address Windows host.....	7
4.3	Adding a DHCP client.....	7
4.4	Local host address.....	8
4.5	Quitting and saving.....	8
5	Overview of GNUspool architecture.....	9
5.1	Daemon processes.....	9
5.2	System directories.....	9
5.3	IPC Facilities.....	10
5.4	Files used by GNUspool.....	10
5.4.1	Internal spool files.....	11
5.4.2	Help and Message files.....	11
5.4.3	Printer Definitions and Setup Files.....	12
5.4.4	Configuration files.....	13
5.4.5	Configuration files held in /usr/local/etc.....	13
5.4.5.1	GNUspool Hosts and Clients File.....	13
5.4.5.2	GNUspool Master Configuration File.....	13
5.5	Ports and Network Services.....	13
6	User administration.....	15
6.1	Adding New Users.....	15
6.2	Removing Users.....	16
6.3	Setting Privileges.....	16
6.4	Setting Default & User Priorities.....	17
6.5	Charging for Use.....	17
6.6	Setting Classcodes.....	18
7	Other administration matters.....	19
7.1	Startup and shutdown of GNUspool.....	19
7.1.1	Gspl-start.....	19
7.1.2	Gspl-stop.....	19
7.2	Gspl-conn and Gspl-disconn.....	19
7.3	Gspl-suspend and Gspl-release.....	19
7.4	Startup on boot.....	20
7.4.1	Starting all printers on boot.....	20
7.4.2	Cleaning and subdividing the spool directory.....	20
8	Backup of GNUspool System.....	21
8.1	Jobs.....	21
8.2	Printers.....	21
8.3	User permissions.....	22
8.4	Other files.....	22
8.4.1	Help and resource files.....	22
9	Print jobs and form types.....	23
9.1	Print Jobs.....	23
9.2	Formtypes (Paper Types & Suffices).....	23
10	Configurability.....	24
10.1	Default Options.....	25

## GNUspool Administration Manual

10.1.1 Setting Up Defaults.....	25
10.2 Setting views of the Job and Printer lists.....	26
10.2.1 Selecting Jobs and Printers by Printer Name.....	26
10.2.2 Selecting Jobs by Owner.....	26
10.2.3 Selecting Jobs by Title.....	26
10.2.4 A real example.....	28
10.3 Job and Printer list Formats.....	28
10.3.1 Adding New fields.....	33
10.4 Help & Error Messages.....	33
10.5 Keys & Commands.....	33
10.5.1 Specifying Different Keys.....	34
10.5.2 Disabling Commands.....	34
10.5.3 Customising Commands.....	35
11 Extensibility.....	36
11.1 Printer Error Handling.....	36
11.2 Scripts.....	37
11.2.1 Printer Log Files.....	37
11.2.1.1 Viewing a Log File.....	37
11.2.1.2 Archiving / Purging Log Files.....	38
11.3 Macros.....	39
11.3.1 Publishing Documents for Printing on Demand.....	39
11.3.1.1 Enhancements.....	40
12 Advice & Trouble Shooting.....	42
12.1 Files & Directories.....	42
12.1.1 User Program Directory.....	42
12.1.2 The spd Internal Directory.....	43
12.1.3 The libexec/gnuspool Internal Directory.....	44
12.1.4 The ptrs Internal Directory.....	45
12.2 IPC Facilities.....	46
12.2.1 Looking at IPC Facilities.....	46
12.2.2 Problem - Cannot Start GNUspool.....	47
12.2.3 Deleting IPC entries owned by gnuspool.....	48
12.2.4 Processes.....	48
12.3 Messages about key clashes entering gspl-pq or gspl-user.....	49
12.4 Warning messages about unknown key name.....	49

# 1 Introduction

GNUspool is a fully functioned, high performance Print Spooler and Management System for a wide range of machines running a Unix-style Operating System.

The original version was written between 1984 and 2008 by John Collins at Xi Software Ltd and marketed as "Xi-Text". The names, including many of the program and interface names, have been changed to GNUspool or derivatives and the installation directories have been changed to conform to GNU standards.

The product consists of a "core product" or "basic product" which contains the scheduling software, command-line and character-based interfaces. Additional options provide for:

- An interface to receive incoming jobs from LPD-style clients
- An interface to send outgoing print jobs to LPD-style clients
- An X-Windows Motif Toolkit Interface (not supported under GNU)
- An X-Windows GTK+ Toolkit interface
- An API for use with C and C++
- An Interface for MS-Windows (currently this relies on non-free software and needs rewriting – assistance welcomed).
- An API for use with MS-Windows
- Browser Interfaces

The basic manuals cover the "basic product" and the X-Windows interfaces. Additional supplements cover the other interfaces.

The basic manuals are:

- User Guide — a quick introduction and "cookbook" for use of GNUspool
- Reference manual — a complete description of all components of the basic product.
- Administrator Guide — this manual. Information about installation and customisation of the software.

Also available are:

- API reference manual for Unix and MS-Windows API
- MS Windows Interface Manual
- Browser Interface Manual

## 1.1 Documentation Standards and Program Usage

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 GNUspool or standard Unix facilities

**Warnings and important advice**

## 1.2 Command Line Program Options

Almost all of the programs that make up GNUspool 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; White space may be inserted into flag arguments as in

```
gspl-pr -c 3 -f invoices
```

or it may be left out as in

```
gspl-pr -c3 -finvoices
```

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

```
gspl-pr -mwsv
```

or separated, as in

```
gspl-pr -m -w -s -v
```

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

```
gspl-pr +copies 3 --formtype invoices
```

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

## 2 Installation

GNUspool is now built directly from the sources and installed by “make install” so there is no special installation routine.

### 2.1 Installation directories

The following directories are used to hold component parts of GNUspool. Note that there is a method for relocating parts of them after the product has been built, to use different directories. In conformity with GNU standards, the directories are by default based on `/usr/local`, but you may well want to relocate the spool directory, which can be large

Directory	Function
<code>bin</code>	User-level programs
<code>libexec/gnuspool</code>	Internal programs
<code>sbin</code>	Administration programs
<code>share/gnuspool</code>	Data and control files
<code>share/gnuspool/help</code>	Help files for programs
<code>share/gnuspool/ptrs</code>	Description of printers and form types
<code>var/gnuspool</code>	Spool directory for pending jobs
<code>var/gnuspool_alt</code>	Alternate spool directory

These directories may all be separately located if required; this is described later.

### 2.2 Installing the system user

Most internal files, and IPC facilities such as shared memory segments and semaphores, are owned by a user `gnuspool`. This user name is like a "sub-super-user" for GNUspool facilities. It is not intended that the `gnuspool` user name should be used extensively, however it (hopefully) serves to distinguish `GNUspool` files from other files.

The only files which are not owned by `gnuspool` are the scheduler process `spshed` and the network server process `xtnetserv`, which have to be owned by `root`, as they have to be able to transform to other users.

During the installation, it will be necessary to create user `gnuspool`. If it does not exist. We normally recommend a user id of 50 (i.e. in the "system users" below 100), and your "create new user" routine may need a special option to install this. It is not entirely necessary, however, but we do not suggest that all the standard user-level login directory and contents be included, as the user name is not intended for general use.

When setting up the default group for user `gnuspool`, we suggest that a group such as `daemon`, or `sys` is selected. If you are running Xi-Batch which hopefully will become GNUbatch shortly and shares some facilities, you may want to make it the same group as `batch` or `gnubatch`.

### 3 Defining and installing printers

In order to use GNUspool for actual printing, printers have to be defined and installed.

When a printer is defined, the characteristics of the printer, such as the control language to converse with it and the interface (serial, parallel, terminal server) are specified, and a "printer setup file" is created in the `ptrs` directory, denoted by `SP00LPT` and by default `/usr/local/share/gnuspool/ptrs` although you may have compiled in something else.

When a printer is installed, the printer definition is inserted into the list of printers available for access by GNUspool. If GNUspool is networked, it is immediately made available to other hosts on the network.

After installing the printer, the printer must be set running to start jobs being printed. This can be done on other hosts on the network, or via the windows interface etc.

There is a Perl script `gspl-ptrinstall` in the administration program directory `/usr/local/sbin` to assist with this.



## 4 Setting up networking

GNUs pool can run with jobs and printers shared between 2 or more Unix hosts, possibly different platforms. Additionally, there is the MS Windows interface and the API.

Three things have to be done:

1. Some standard "service" names need to be inserted in the `/etc/services` file.
2. Hosts need to be configured in a hosts file `/usr/local/etc/gnuspools.hosts`

To set up hosts, a special host editor program `hostedit` is available in the administration binaries in `/usr/local/sbin`. To run this enter:

```
gspl-hostedit -I /usr/local/etc/gnuspools.hosts
```

The `-I` option edits the file in place.

With a new installation, the takes the following form:



### 4.1 Adding a Unix host

Press "a" to add a host, you should get the prompt:

```
Is new entry a Unix host(Y) or Client(N)? [Y]
```

Just press ENTER. Next you should get the prompt.

```
Unix host name:
```

Type the name of the host, e.g. on our site, one is called "avon"<sup>1</sup>. You should then get the prompt:

Any alias for avon:

Unless you require an alias, just press ENTER. Next you will get the prompt:

Probe (check alive) before connecting? [Y]

Again press ENTER (this checks that the host is on-line before attempting a TCP connection, however in some cases a Firewall etc prevents UDP messages from getting through and this must be avoided).

Trust host with user information? [Y]

Normally you will want to do this. It causes various Unix hosts to regard Windows PCs as "logged in" to all hosts once they have successfully logged in to one.

Manual Connections only? [N]

This concerns whether you want the connection to the other Unix host to be attempted as soon as GNUs pool is started, or manually by means of `spconn`. Reply `y` if you require this.

Default timeout value OK? [Y]

Press ENTER unless you want a variation in the connection timeout. The display should now look like this:

```

jmc@torres.xisl.com: /home/jmc/src/build/Text-shared
add host change host delete host local addr default user quit
Host      Alias      IP Addr      Dft mc      User
Unix      Win
Current: torres.xisl.com      193.112.238.23
avon      193.112.238.29 probe      trusted

```

This should be repeated for each host (and should also be repeated on each host so configured). To make any amendments, move the cursor to the relevant host name and type `c` or to delete it type `d`.

<sup>1</sup> Named after the character in "Blake's Seven" rather than the cosmetics company

## 4.2 Adding a fixed IP address Windows host

Press **a** to add a host name, and this time type **n** to indicate that a Windows host is being added. The prompts are as follows:

Specific Windows Host (Y) or `roaming user' (N)? [Y]

Press ENTER to denote a specific IP address.

Windows client host name:

Type the name of the hosts, e.g. "kim".

Any alias for kim:

Again, press ENTER unless you want to give an alias.

Default user:

This is prompting for a default user name to be used for access — the normal user of that PC, although other users can use it, specifying a password. This can be omitted if required.

Password-check user(s)? [N]

This can be turned on to force password-checks on all users if required.

Default timeout value OK? [Y]

Again, this can be left as the default or not (it is slightly more important, as if the password checks are enabled, it gives the time after which the user will have to log in again with his password if he has not done anything in the meantime).

## 4.3 Adding a DHCP client

This is where the IP may change each time and the recognition is by the user name (or possibly a Windows user name matched to a Unix user name).

Again, press **a** to add a host name and **n** to indicate that a Windows host is being added, and now type **n** again to select DHCP clients.

Unix user name:

This is the user id under which activities will be run on the Unix host. Be careful to make the case of characters correct (usually all lower case).

Windows user name:

This is the windows user name, or blank if the same as the Unix user name. It is not case-sensitive.

Do you want to specify a default machine name? [N]

This enables the user to specify the (Unix) host name of the usual PC at which the user works. He or she will be able to "log in" at other hosts with the password.

Password-check user(s)? [N]

You can set this to insist on a password in all cases.

Default timeout value OK? [Y]

Again this gives the timeout value. After this time of inactivity, the user will have to log in again.

## 4.4 Local host address

On some systems attached to more than one network, it may be necessary to explicitly set out the local host IP address to be used in communicating with other hosts. To set a local IP address, type `l`.

## 4.5 Quitting and saving

Type `q` to quit and save the newly-created hosts file in `/usr/local/etc/gnuspools.hosts`.

## 5 Overview of GNUspool architecture

This is a brief overview of GNUspool for the assistance of administrators. Please see the System Reference Manual for a more detailed discussion.

### 5.1 Daemon processes

GNUspool relies on one or three continuously-running "daemon processes" for its operation, plus one process for each running printer.

If the product is non-networked (no linked Unix host or hosts, no Windows interface or API), then there is an `spshed` process which allocates jobs to printers.

If the product is networked:

- A further `spshed` process monitors and processes network traffic between other Unix hosts and the Windows Interface.
- A process `xtneterv` handles incoming jobs from the Windows Interface and manages API sessions. (It may "fork off" additional copies of itself for each API session).

In addition there is an `spd` process for each running printer and these may invoke other processes such as Ghostscript to convert output and various network drivers if the printer is connected via a network.

These processes are started by running `gspl-gspl-start` and terminated by using `gspl-stop`. Please do not attempt to start GNUspool in any other way.

Should some system crash ever require you to kill any of the `spshed`, `spd` or `xtneterv` processes, please do not use "kill -9" initially, as this will force immediate termination. Instead use just "kill", which will give `spshed` a chance to release system resources first.

### 5.2 System directories

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

These directories may be relocated from the values set when GNUspool 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/gnuspool.conf`, to ensure consistency. The directories and corresponding environment variables are as follows:

Directory	Environment Variable	Function
<code>libexec/gnuspool</code>	<code>SPROGDIR</code>	Internal programs
<code>share</code>	<code>SDATADIR</code>	Data and control files
<code>share/gnuspool/help</code>	<code>SPHELPDIR</code>	Help files for programs

Directory	Environment Variable	Function
<code>share/gnuspool/ptrs</code>	<code>SP00LPT</code>	Description of printers and form types
<code>var/gnuspool</code>	<code>SP00LDIR</code>	Spool directory for pending jobs
<code>var/gnuspool_alt</code>	<code>ALTSP00L</code>	Alternate spool directory

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

### 5.3 IPC Facilities

GNUs pool uses the "System V IPC facilities" to communicate with itself internally.

- One message queue is used to send requests for the main scheduler process, `spshed`.
- Two shared memory segments record jobs on the queue. These are periodically written out to the file `spshed_jfile` in the spool directory.
- One shared memory segment saves details of printers. This is likewise periodically written out to the file `spshed_pfile`.
- One shared memory segment is used to buffer pending requests as the size of messages which may be sent on message queues is limited on many systems.
- One set of semaphores controls access to the shared memory segments (this is only if use of semaphores is compiled in which we do not recommend).

The IPC facilities can be displayed by running `ipcs`. The items in question are owned by `gnuspool` with a key of `0x58691xxx`.

The system utility `gspl-ripc` provided in the installation directory may be used to inspect the IPC facilities especially after a crash and optionally to delete them.

The shared memory used for jobs and printers may have to be reallocated if the number of jobs or printers exceeds the initial amount allocated. On some systems this may be difficult if other software is in use which has exhausted the available shared memory space. If there are difficulties in this area, make sure to start GNUs pool with a reservation of the appropriate amount of space, as described under `gspl-pstart`.

Depending on the build parameters, the shared memory may be replaced by memory-mapped files and the semaphores by lock files.

### 5.4 Files used by GNUs pool

The following sections list the files used and created by GNUs pool.

### 5.4.1 Internal spool files

The following GNUs pool internal files are held in the spool directory, which by default is `var/gnuspools`:

File	Function
<code>spufilen</code>	File of user permissions with version number n
<code>spcharges</code>	File of user charges
<code>sputmp</code>	Temporary file used when reformatting spufile
<code>spshed_reps</code>	Log file recording system start-up, shutdown and errors
<code>spshed_jfile</code>	File used for recording the job queue
<code>spshed_pfile</code>	File used for recording the printer list
<code>SPnnnnnnnn</code>	File holding the text of a spool job
<code>PFnnnnnnnn</code>	Holds page offsets where the page delimiter is not a form feed.
<code>ERnnnnnnnn</code>	Used to hold information to be mailed back to a user.
<code>sppmm_jobi</code>	<i>If using memory-mapped files</i> live job hash file.
<code>sppmm_jobd</code>	<i>If using memory-mapped files</i> live job data file.
<code>sppmm_ptrs</code>	<i>If using memory-mapped files</i> live printers file.
<code>sppmm_xfer</code>	<i>If using memory-mapped files</i> live buffer file.
<code>spjob.lock</code>	Lock file for jobs
<code>spptr.lock</code>	Lock file printers
<code>spxfer.lock</code>	Lock file for transfer buffer

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

One of the main product options is to subdivide the spool directory into several smaller subdirectories, making the main directory more manageable.

Please note in particular the file `spshed_reps`. This is the system log file and processes such as `spshed`, `spd` and `xtnetsrv` send messages to it in the event of error conditions arising. If you have any support questions, please look in this file.

### 5.4.2 Help and Message files

GNUs pool 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 `gnuspools` and held in the directory `/usr/local/share/gnuspools/help`.

File	Function
<code>int-config</code>	Message file used by internal programs.
<code>rest.help</code>	Message file used by all other user programs.
<code>spq.help</code>	Message file used by <code>gspl-pq</code> .
<code>spuser.help</code>	Message file used by <code>gspl-user</code> .
<code>splpd.help</code>	Message file used by lpd interface.
<code>xmspq.help</code>	Message file used by <code>gspl-xmpq</code> and <code>gspl-xpq</code> .
<code>xmspuser.help</code>	Message file used by <code>gspl-xmuser</code> and <code>gspl-xuser</code> .

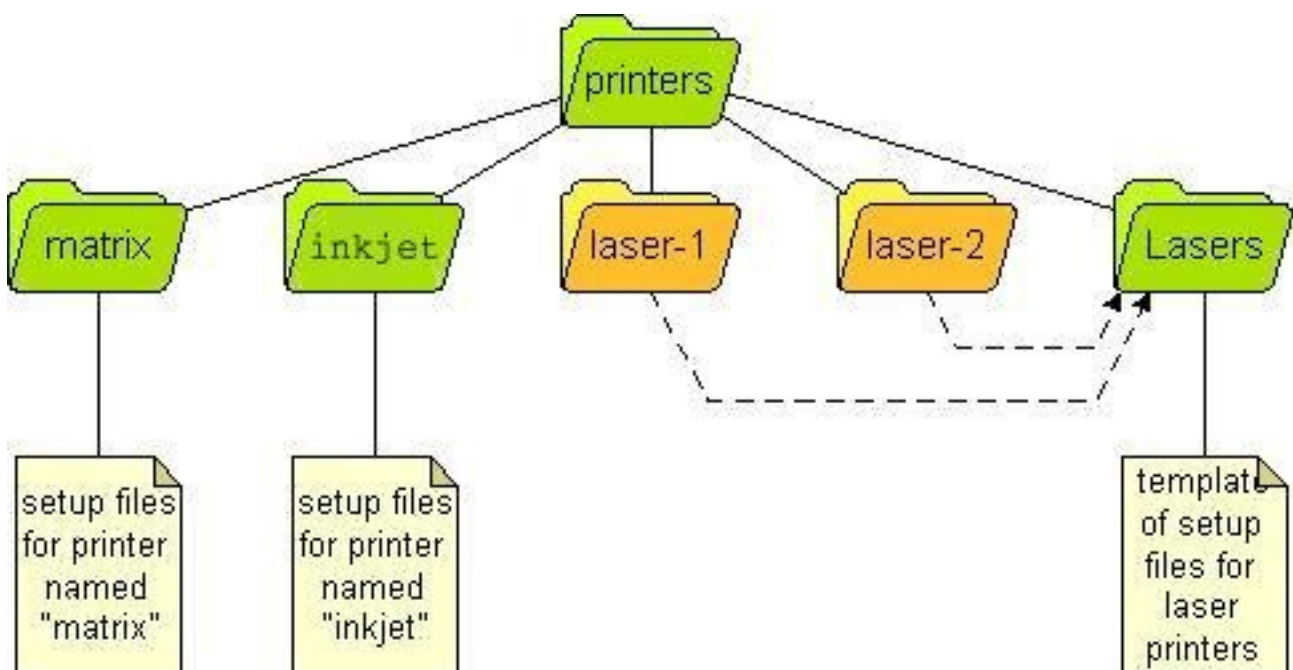
Refer to the chapters on Configurability and Extensibility of both the System Reference Manual and this Administrator Guide for details of how to modify these files as may be required.

Note that certain options in some of the programs `gspl-pq` and `gspl-user` may cause local copies of the files `spq.help` and `spuser.help` to be written out into user's areas, possibly under different names.

### 5.4.3 Printer Definitions and Setup Files

The printers directory contains the print control and formatting information for all of the printers that are logically attached to GNUspool on that Unix host. Within this directory there is a sub-directory for each printer. Each sub-directory holds one or more setup files for the printer, which make up the Printer Definition.

To maintain identical printer definitions for two or more identical printers a master directory can be set up as a template. Each of the printer directories can then be replaced by a symbolic link to the template directory.





Symbolic links can also be used inside printer directories where just certain components are identical across two or more printers.

Refer to the chapter on Printers - Setting up and Administration for details of printer setup and their definitions.

#### 5.4.4 Configuration files

Various programs allow sets of options to be written into local "configuration files" named `.gnuspool` so that further runs of those programs pick up a different selection of options from the defaults.

These are text files containing sets of options for various programs, which may be edited by the user, but are normally edited by the programs concerned.

#### 5.4.5 Configuration files held in `/usr/local/etc`

The following files are always held in the system directory `/usr/local/etc`.

##### 5.4.5.1 GNUspool Hosts and Clients File

The file `/usr/local/etc/gnuspool.hosts` is used on networked installations of GNUspool to denote details of the remote hosts and clients to which connection is to be made. This is fully described in the System Reference Manual.

##### 5.4.5.2 GNUspool 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/gnuspool.config`.

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 GNUspool may be included in this file, not just those shown in the example.

For example:

```
SP00LDIR=/usr1/spool/spd
SPROGDIR=/usr1/spool/bin
MAILER=/usr/lib/sendmail
```

Please note that the text to the right of the equal sign, `=`, is taken literally; there is no recursive expansion of `$name` constructs except for the message file names `SPQCONF`, `SPUSERCONF` and `SPRESTCONF` (for which up to 10 recursive expansions are applied).

## 5.5 Ports and Network Services

GNUspool also uses 7 entries in the services file, `/etc/services`. When installed with the default values the entries will look like this:

```
gnuspool          48100/tcp # Connection port for GNUspool
```

```
gnuspoo l      48100/udp # Probe port for GNUs pool
gnuspoo l-feeder 48101/tcp # Feeder port for GNUs pool
gnuspoo l-netsrv 48102/tcp # DOS client enqueue port for Xt
gnuspoo l-netsrv 48102/udp # DOS client enquiry port for Xt
gnuspoo l-api    48103/tcp # GNUs pool API library for GNUs pool
gnuspoo l-api    48103/udp # GNUs pool API library for GNUs pool
```

The port numbers can be changed, but all hosts must use the same numbers if GNUs pool connections are to be made.

The installation script will set up entries for the GNUs pool TCP and UDP connections in the `/etc/services` file.

N.B. If you are using a name service like NIS instead of the services file, the entries must be copied to that service.

It is possible that the port/socket numbers used as standard by GNUs pool will conflict with another product. If this is the case they can be moved by editing the `/etc/services` file after installation.

It is essential that the values used are the same on all GNUs pool hosts, Unix and PC, that are on the same network. Even if two hosts are never going to be connected, make the socket numbers the same.

## 6 User administration

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

There are 5 aspects to the GNUspool user account:

Privileges	Control access to usage and administration functions of the system. For example, the privilege to stop and start printers.
Charges	Provide an accounting mechanism for each users print usage.
Priorities	When there is a back log of jobs to print the queue is sorted using a combination of priority and how long a job has waited.
Classcodes	Group the users operationally. It is possible to put fire walls between different groups of users for security or convenience.
Restrictions	To limit each user to a specific set of printers and / or formtypes.

The privileges, classcodes and printer restrictions provide two different dimensions of security. The privileges control what users may do. The classcodes control what jobs and printers users may exercise their privileges on. Printer restrictions can be used instead of or as well as the classcodes.

In addition to these features the user interface can be configured differently for different users and activities.

There are 5 programs for administering user accounts which are:

<a href="#">gspl-charge</a>	A command line utility for querying and adjusting user charges
<a href="#">gspl-uchange</a>	A command line utility for modifying user accounts
<a href="#">gspl-ulist</a>	A command line utility for listing user accounts
<a href="#">gspl-user</a>	An interactive tool for viewing and changing user accounts on character terminals.
<a href="#">gspl-xuser</a>	A alternative to gspl-user.

The programs are described in detail in the System Reference Manual.

### 6.1 Adding New Users

To add a new user to GNUspool, they must first have a Unix login created. Once the Unix account is set up, one of the user administration programs [gspl-user](#), [gspl-uchange](#) or [gspl-xpuser](#) should be run to update the record of the user. [gspl-user](#) and [gspl-xuser](#) will notice the additional user and prompt for whether the records should be updated - reply **y** unless you have a very slow machine with a lot of users currently interacting with GNUspool.

Alternatively run:

```
gspl-uchange -R
```

to do this non-interactively. This could be incorporated into the "add new user" script if applicable.

If GNUspool is running the scheduler will be updated, but it is possible that information will not be fully propagated until the system is stopped and restarted. This can be recognised by the user or group names being represented by the user ids.

## 6.2 Removing Users

No special action needs to be taken when users are deleted.

## 6.3 Setting Privileges

The available privileges are:

Privilege	Description
Edit admin file	May edit the administration file. This makes a user a full GNUspool Administrator since they can grant themselves any other privilege.
Override class.	Allows user to override their own Classcode restrictions.
Stop spooler	Grants privilege to shutdown GNUspool
Use other forms	Enables user to specify formtypes, other than their default.
Use other printers	Lets user override their default printer. Granted by default.
Change priority on Q	May change priority of job on queue
Edit other users' jobs	May inspect or change other users' jobs
Select printer list	May move to printer list in <a href="#">gspl-pq</a>
Halt/restart printers	May start or stop printers
Add/delete printers	May add or delete printers on list
Set any priority on Q	May set any priority not just in range
Change own default prio/form	May run <a href="#">gspl-user -c</a>
Unqueue jobs	May take jobs off the queue for editing
View other users' jobs	May inspect but not change other users' jobs
Edit remote jobs	May edit jobs on remote machines
Edit remote printers	May change printers on remote machines
Access queue options	May access the forms or screens behind the job list in <a href="#">gspl-pq</a> and <a href="#">gspl-xpq</a> to modify parameters. Granted by default.
Save new default options	May make and save changes to default parameters. Granted by default.

Unless otherwise stated in the table, when GNUspool is installed these privileges are turned off. The currently specified default privileges are applied to all new users.

When GNUspool is installed only the super-user, [root](#), and user [gnuspool](#) are set up as system administrators. These two users are granted all of the privileges.

Changing users' privileges and the default settings can be performed with [gspl-user](#),

`gspl-uchange` or `gspl-xuser`.

When setting up GNUspool on a machine one of the first Administration tasks should be deciding which extra Unix users to make administrators.

## 6.4 Setting Default & User Priorities

All print jobs have a priority in the range of 1 to 255. Increasing the priority of a job increases its chances of getting printed sooner, as follows:

- When a job is added to the queue, a copy of the priority is made. This is the working priority.
- The bottom item on the queue is considered. If the priority of this item is less than the working priority of the new job, then the working priority of the new job is decremented and the process repeated with the next item.
- This process is continued until the job reaches the top of the queue or a job is encountered with a priority the same as or greater than the working priority.
- The working priority is saved in case the job's priority is subsequently changed. The change to the job's priority is reflected in the working priority, and the job moved up or down the queue according to the above rule.

Each user has a default priority value which is given to jobs submitted by them, unless they specify otherwise. Users will usually be restricted to a smaller range between their individual minimum and maximum priorities. When it is installed GNUspool sets the default values to be:

minimum	100
maximum	200
default	150

There is no particular reason for these values. They allow for non-standard users to set up jobs which have higher or lower priority than the normal user population.

When a new policy is decided the default values should be adjusted, then existing user accounts modified as required.

When a job is submitted, it is given the user's default priority unless overridden. 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, for example the user may be prevented from submitting jobs at all by making the maximum priority less than the minimum.

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.

## 6.5 Charging for Use

At the end of each job the cost of printing that job is added to the user's stored charge. The cost of the job is calculated using the algorithm:

Where:

$$\left[ \frac{\left\lfloor \frac{n_{cp} + 999}{1000} \right\rfloor \times cpt}{1000000} \right] \times \frac{jp^2}{sp^2}$$

*n<sub>cp</sub>* is the number of characters printed

*cpt* is the charge per 1000 characters printed for the printer

*jp* is the job's priority and

*sp* is hardwired to the installation default priority of 150 units.

If a job, submitted on one machine, is printed on another over the network, then the charge for the printer on the machine where it was printed, and the system default priority on the machine on which the job was submitted are used. The result is used to update the users' account on the latter machine.

In simple terms; the user is charged for the number of characters printed multiplied by the charge per character for that printer, scaled by an amount dependent on how much the priority of the job had been raised above the system default when it was printed.

Using the square of the priority is to penalise users in greater amounts the more they "jump the queue".

The programs [gspl-user](#), [gspl-xuser](#) and [gspl-charge](#) may be used to produce simple statements, query and alter the balance on user accounts.

It is intended to remove charging from GNUspool in the next release. When first written this was important to people but no one seems to use it any more.

## 6.6 Setting Classcodes

Each user has a default set of classcodes. A user must be in at least one class in order to use GNUspool. Only users with override class privilege can increase their set beyond their default set. All users can restrict their effective set to a sub-set of their default.

Changing users classcodes and the default settings can be performed with [gspl-user](#), [gspl-uchange](#) or [gspl-xuser](#).

These are described in detail in the System Reference Manual.

## 7 Other administration matters

### 7.1 Startup and shutdown of GNUspool

The user-level programs `gspl-start` and `gspl-stop` are provided for the startup and shutdown of GNUspool and `gspl-conn` and `gspl-disconn` are provided for attaching and detaching Unix hosts.

Two other administrative programs `gspl-suspend` and `gspl-release` are provided to temporarily suspend assignment of printers to jobs pending certain system-level activities.

Please try to use these utilities wherever possible rather than starting or halting internal processes yourself.

#### 7.1.1 Gspl-start

GNUspool can be started by just running the program `gspl-start`. However there are options to `gspl-start` which should be worth noting:

If `gspl-start` is given two numeric options, then shared memory (or memory-mapped file) space will be reserved for the given number of jobs and printers respectively. This is often necessary on installations where it is not possible to reallocate shared memory after GNUspool has been started, perhaps because other applications have taken all the available shared memory.

To start GNUspool reserving space for 5,000 jobs and 200 printers, use the command:

```
gspl-start 5000 200
```

#### 7.1.2 Gspl-stop

`Gspl-stop` should be used to halt GNUspool. Any jobs running will be aborted. This may only be run by a suitably-privileged user.

If there are problems with it and some of the processes continue to run and you need to kill `psphed` and other processes, try first with just "kill" and not "kill -9" as the latter will not give a chance for the software to delete the IPC facilities.

### 7.2 Gspl-conn and Gspl-disconn

`Gspl-conn` is used to connect to Unix hosts set up as for "manual connection" in the hosts file (see page 6), or disconnected using `gspl-disconn`. `Gspl-disconn` disconnects from other Unix hosts previously connected.

Either are invoked with the host name or alias required. They should return immediately, although the network startup or shutdown may take longer (or not at all if the other machine is unavailable).

### 7.3 Gspl-suspend and Gspl-release

Prior to connecting additional hosts, network traffic can be held off by invoking `gspl-`

`suspend` to prevent printing being started (this generates a lot of network messages). `Gspl-suspend` takes an argument in seconds for which it applies or it can be cancelled with `gspl-release`.

## 7.4 Startup on boot

Included in the main directory are sample startup routines as suitable for System V and some GNU/Linux “runlevel” based startup suites, AIX and HP-UX startup routines. We recommend that you install such a startup routine so that GNUspool is set running correctly when your system is booted.

The startup routine contains a `gspl-start` command which includes values of the space for jobs and printers to allocate, as described on page 19. Please edit the values, especially that for job space, to avoid allocating too much shared memory on the one hand or too little on the other.

If the job or printers run out of shared memory space, additional space is allocated, but this is not always possible later. Hence it is advisable to allocate space initially for the maximum number of jobs which are likely to be held in the queue.

### 7.4.1 Starting all printers on boot

We don't recommend that this be done as a general rule, but many of our users like all the printers (local to that machine) to be restarted on boot. If you want to do this, uncomment the lines given in the startup file.

### 7.4.2 Cleaning and subdividing the spool directory

Over time various “dead wood” files may be left in the spool directory. There may be jobs listed in the job list file (`spshed_jlist`) which don't have any counterpart data files (`Spnnnnnnnn`) or vice versa.

There is a utility program `gspl-setspsdir` which may be used to tidy up the spool directory and either requeue orphaned job files or to delete them.

As a further option, where there are a large number of job files, it is often helpful to share them out between subdirectories of the main spool directory, so no directory has a huge number of files in it. `Gspl-setspsdir` lets you set or change that.



## 8 Backup of GNUspool System

Certain utilities are provided to back up the GNUspool system, by saving the current jobs, printers and user accounts. You may want to do this when you have set up a complicated list of printers. Alternatively you may want to do this if you are upgrading to a new major release of GNUspool. We make efforts to keep a common format between minor releases, so it should not be necessary to do this.

All the backup options create a single shell script which, if run, would recreate the relevant jobs, printers or user list. This may be edited if required.

Jobs, printers and user profiles should be saved in any order, but it is probably best to restore jobs last as they may depend on some of the other items.

### 8.1 Jobs

To convert the job list to a shell script, the utility `gspl-cjlist` is used. This may be run at any time, even when GNUspool is running, but we would suggest that not too much activity is in progress at the time. `gspl-cjlist` may be found in the administration directory, by default in `/usr/local/sbin`.

You will need to create a directory into which the job data files are saved.

Here is a suggested routine for saving the jobs:

1. Create a backup directory, for example `/usr/gssave/May08`.
2. Create a subdirectory within that directory for the scripts, for example you might choose `/usr/gssave/May08/Scripts`.
3. Run `gspl-cjlist` on the existing job files.

Thus:

```
mkdir -p /usr/gssave/May05/Scripts
cd /usr/gssave/May05
gspl-cjlist -D /usr/local/var/gnuspool spshed_jfile Jcmd Scripts
```

The shell command to recreate the jobs will be put in `Jcmd`, and the job script files in `Scripts`.

For more details, and especially the handling of errors, please see the documentation of `gspl-cjlist`.

`gspl-cjlist` only considers and saves jobs on the machine on which it is run.

To restore the jobs, just run the shell script `Jcmd`.

### 8.2 Printers

Printers are saved in a similar manner to jobs, except no additional directory is required, using the program `gspl-cplist`. Only one file is generated, a single shell script of `spadd` commands to recreate the printers.

Assuming that the `/usr/gssave/May08` directory described above has already been

created and selected to save jobs and the path containing `gspl-cplist` has been set up as for `gspl-cjlist`, the following command will save the printers:

```
gspl-cplist -D /usr/local/var/gnuspool spshed_pfile Pcmd
```

When restoring jobs and printers, you will almost certainly need to restore printers first.

Please see the documentation of `gspl-cplist` for more information, particularly regarding error handling.

## 8.3 User permissions

User permissions may be saved in a similar manner to jobs and printers, using the command `gspl-spuconv`.

As with `gspl-cplist` a shell command is saved which when run will reset the user permissions and defaults for all users.

To continue the examples above for jobs and command interpreters, the following will create a command file capable of recreating the user permissions.

```
gspl-spuconv -D /usr/local/var/gnuspool spufile0 Ucmd
```

When restoring the whole system, it is not necessary, but probably desirable, to restore the user permissions prior to restoring jobs, variables, and command interpreters.

Please see the documentation of `gspl-spuconv` for more information, particularly regarding error handling.

## 8.4 Other files

### 8.4.1 Help and resource files

Help files in the help file directory (`/usr/local/share/gnuspool/help` by default) will be replaced when the software is upgraded. We suggest that you make a copy of the existing versions before you begin and if necessary consult us regarding integrating your customisations into the new release. In most cases the changes will be correction of spelling mistakes and clarifications of existing error messages.

Please note also that in some cases, such as when users save a customised set of screen formats for `gspl-pq` or `gspl-user`, a local copy of the files `gspl-pq.help` etc are saved in the user's home directory. These should probably be deleted and recreated as necessary.

The Motif programs `gspl-xmpq` and `gspl-xmuser` have information kept in a file called `GSP00L` in an `"app-defaults"` directory. When a user tailors their user interface, a local copy of the `GSP00L` file is created in their home directory, which may not reflect all the changes in the new versions of the software. This warning does not apply to the GTK+ versions `gspl-xpq` and `gspl-xuser`.

## 9 Print jobs and form types

This is a summary of some of the major features of print jobs and form types appropriate for this administration manual. For full information, please see the System Reference Manual.

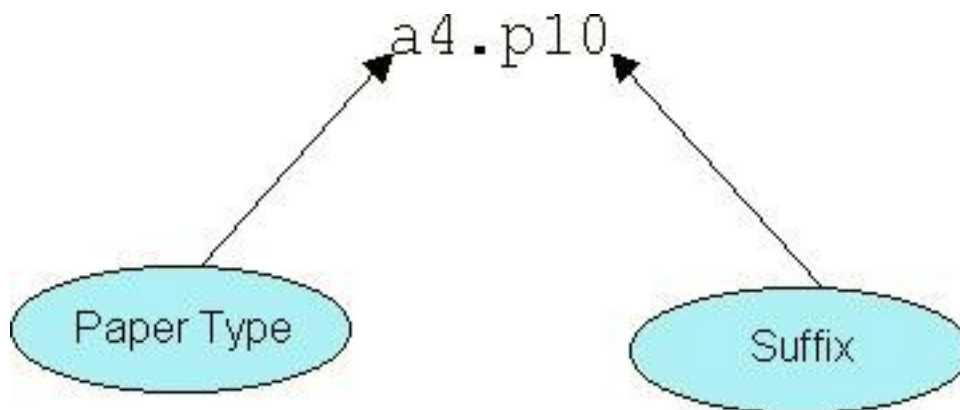
### 9.1 Print Jobs

A print job consists of a set of data which is to be output and the specification of how to output it. The data can be ASCII text, Postscript, Graphics or anything else that is understood by the output device. The specification controls things like how many copies to produce, what formatting to apply, which output devices to use and so on.

### 9.2 Formtypes (Paper Types & Suffices)

There are many different types of media that can be used in output devices like printers and plotters. There are also many different ways of formatting the data that is output on each media type. To support this idea GNUspool has a concept of a formtype.

A form type consists of two parts, a "paper type" corresponding to the actual paper loaded and an optional "suffix" denoting variations of how to print on it, for example:



In order to change the paper type, the printer has to be stopped and started on the spooler. However the suffix can change automatically as different jobs are printed with different suffixes.

In many cases, there is no need for special treatment and no suffix is required at all and the form type can be something like labels or cheques.

Various options in the printer setup file control actions to be taken at various stages of the print cycle. Generally speaking, there is a setup file for each paper type and for each printer. Suffix handling is done within the setup file.

## 10 Configurability

GNUsPool can be highly configured to restrict or enhance the scope and function of user activities. The System Reference Manual gives the basic information for configuring the user interfaces. This chapter describes some of the applications of this configurability and their implementation.

Here is an example that shows how program `gspl-pq` might look when configured to take advantage of function keys and show a different set of information:

Seq	Title	Form	AtPg	Pgs	Cps	Printer
0	Monthly Item Report	a4.p12		370	1	
1	Month end Report	a4.p12		29K	1	
2	addresslabel	label		45	3	kn*
3	invoice	a4.p12		112	1	kn*
4	picklist	listing.p10		1444	1	kn*

Printer	Form	State:Ms	Jobno	User	Flags
deskjet	a4.ps	idle			
avon:demo	scrap	idle			

-- 4 more printers below --

F1-----F2-----F3-----F4-----F5-----F6-----F7-----F8-----F9-----

Job: help list form copies printer options view Summary Quit

Ptr: help list form start halt STOP NOW -- -- Quit

This configuration could be specific to a particular user or activity. In this case it is taken from a real configuration belonging to a particular user when sending jobs to printers to a particular office. The screen display has been changed as follows:

- The `Jobno`, `User` and `Priority` fields in the job list have been removed.
- The `device` and `Lim` fields in the printer list have also been removed.
- The `Title` field is widened to display more text.
- The footer is expanded to include function key reminders.
- The blank line under the Printer field column headings has been removed to allow more printers to be seen.

This is what the standard configuration of `gspl-pq` looked like when invoked on another terminal, at the same time:

Seq	Jobno	User	Title	Form	AtPg	Pgs	Cps	Pri	Printer
1	torres:243	jmc	Monthly Item R	a4.p12		370	1	164	
2	torres:243	jmc	Month end Repo	a4.p12		29K	1	153	
3	2 3476	jmc	addresslabel	label		45	3	155	kn*
4	torres:243	jmc	invoice	a4.p12		112	1	150	kn*
5	torres:243	jmc	picklist	listing.p10		1444	1	100	kn*

Printer	Device	Form	State:Ms Lim	Jobno	User	Flags
deskjet	lp0	a4.ps	idle			
avon:demo	<dh>	scrap	idle			
-- 4 more printers below --						

Xi-Text spq (c) Xi Software Ltd 2000 (? for help)

As well as changing the format, the view may be restricted so only particular printers and jobs for them are displayed, or possibly only jobs submitted by particular users.

The following sub-sections describe various ways in which the interface can be configured.

## 10.1 Default Options

All of the programs in the Base Product can be given or require options on the command line. Default options and values can be specified to save typing or to enforce their use. The defaults can be set up globally, per user, by current working directory or by activity.

### 10.1.1 Setting Up Defaults

Default options for each command line utility, like `gspl-pr` and `gspl-qlist`, is specified using an environment variable. The environment variable for each programs use a variable of the same name as the program, in upper case and with non-alphabetic characters replaced by “\_” (underscore). This holds a string of one or more options and any associated arguments.

Each program additionally uses a further environment variable to which may select an alternative message file. There is a rather more restricted set as programs, particularly non-interactive ones, share message files.

The program descriptions in Chapter 6 of the System Reference Manual list the keywords at the start of each entry. For example for `gspl-pq`, the description shows a keyword of `GSPL_PQ` for options and `SPQCONF` for an alternative help file.

The chapter on Configurability in the System Reference Manual explains the theory. Here are some examples of how defaults can be used for `gspl-pq` :

```
SPQCONF=/usr/local/share/gnuspool/help/whouse.help
```

This tells `gspl-pq` to load the file `whouse.help` in the specified directory instead of `spq.help` from the `progs` directory. It is a good idea to use meaningful file names, in this case `whouse.help` could be the configuration for "Warehouse Jobs".

If all of these "Warehouse jobs" are going to specific printers in the warehouses the `-q` option can be used to restrict the view accordingly. For example:

```
GSPL_PQ=-Z -q whlist*,whlabel*
```

This just selects jobs whose printer name starts with `whlist` or `whlabel`. The `-Z` option excludes jobs which have no printer specified.

## 10.2 Setting views of the Job and Printer lists

There are facilities for selecting which jobs and printers to display. Whilst these can be used for security purposes they are also just as useful for selecting logical views of the print spooling system.

Users can have logical views imposed upon them or be allowed to select their own. A user can only affect a job or printer if they can see it.

### 10.2.1 Selecting Jobs and Printers by Printer Name

Programs `gspl-pq`, `splist` and `xspq` have options for printer name selection. The first two use Keywords for setting default values. For example to select only jobs for printer `Ljet`:

Program	Keyword / Resources
<code>gspl-pq</code>	<code>GSPL_PQ=-Z -q Ljet</code>
<code>gspl-qlist</code>	<code>GSPL_QLIST=-Z -q Ljet</code>

### 10.2.2 Selecting Jobs by Owner

Programs `gspl-pq` and `gspl-qlist` have options for selection by user name. The first two use Keywords for setting default values. For example to select only jobs owned by user `fred`:

Program	Keyword / Resources
<code>gspl-pq</code>	<code>GSPL_PQ=-ufred</code>
<code>gspl-qlist</code>	<code>GSPL_QLIST=-u fred</code>

### 10.2.3 Selecting Jobs by Title

The output from `gspl-qlist` and the job lists in `gspl-pq` can be reduced to show only jobs whose Title field match a pattern. The pattern is specified in the same way as it is for printer names and users. From the command line the Title may be specified by the options:

```
gspl-qlist -t selection
gspl-pq -t selection
```

A new selection can be specified from within `gspl-pq` by pressing the `=` key from either the printer or job list on the main screen. This brings up the Program options screen, which has the title parameters on the third line down. In `xspq` the equivalent window is opened using `View options` from the `File` menu.

The jobs may be selected by a simple string, a pattern or a list of strings and patterns for text in the Title. Patterns and strings in a list are separated by commas. It may be necessary to put quotation marks around the Title specification when invoked from a shell command.

The wild-card options for pattern matching are:

<code>*</code>	Matches anything
<code>?</code>	Matches one character
<code>[a-m]</code>	Matches one character in list or range
<code>[!n-z]</code>	Matches one character not in list or range

They may be given as a comma-separated list of alternatives, including the use of shell-style wild-cards. For example

<code>Sales*</code>	The string <code>Sales</code> selects all jobs whose Title starts with the word <code>Sales</code> . It does not select any jobs which have one or more characters before the word <code>Sales</code> , or which are spelt with a lower case <code>s</code> at the beginning.
<code>Sales*,*Rental</code>	Selects jobs starting with the string <code>Sales</code> or ending in the string <code>Rental</code> .
<code>*Sale*</code>	More than one asterisk can be used. This example selects any Title containing the string <code>Sale</code> . It does not have to be a separate word, i.e. the words <code>Sales</code> and <code>Salesman</code> will also match.
<code>[S,s]ales</code>	Use the list or range wild card to cope with similar patterns like words with lower case and capital letters.

The wild cards can be used in combinations like this:

`*[R,r]eport*` will select all the jobs with the word `Report` or `report` in their title.

To invoke `gspl-pq` with the job list restricted using the `-t` option from the command line could look like this:

```
gspl-pq -t "[M,m]onth*[R,r]ep"
```

This invokes `gspl-pq` with the job list restricted to "Monthly Reports" whose titles match the pattern. These will include titles like:

```
Monthly Sales Report
End of month defect reps
```

It does not match titles where the pattern `rep` or `Rep` comes before the pattern `Month` or `month`, such as

```
Sales report at month end
```

To cope with this extra requirement the selection would have to be a list of two

patterns looking something like

```
gspl-pq -t "[M,m]onth*[R,r]ep*,*[R,r]ep*[M,m]onth"
```

### 10.2.4 A real example

The jobs in the example screen for `gspl-pq` at the beginning of this Chapter were selected by invoking `gspl-pq` with the command:

```
gspl-pq -Z -q kn* -u wally
```

This could have been set up as the default in a `.xtext` file using a line like this:

```
GSPL_PQ=-Z -q kn* -u wally
```

Alternatively it could have been set up in an environment variable using the commands (assuming the Bourne or Korn shell are in use):

```
GSPL_PQ="-Z -q par* -u wally"
export GSPL_PQ
```

If these commands are being executed in a general shell script, the current user could be specified by replacing `wally` with `$LOGNAME`. For example:

```
GSPL_PQ="-Z -q par* -u $LOGNAME"
export GSPL_PQ
```

The `$LOGNAME` is evaluated by the shell not the GNUspool programs.

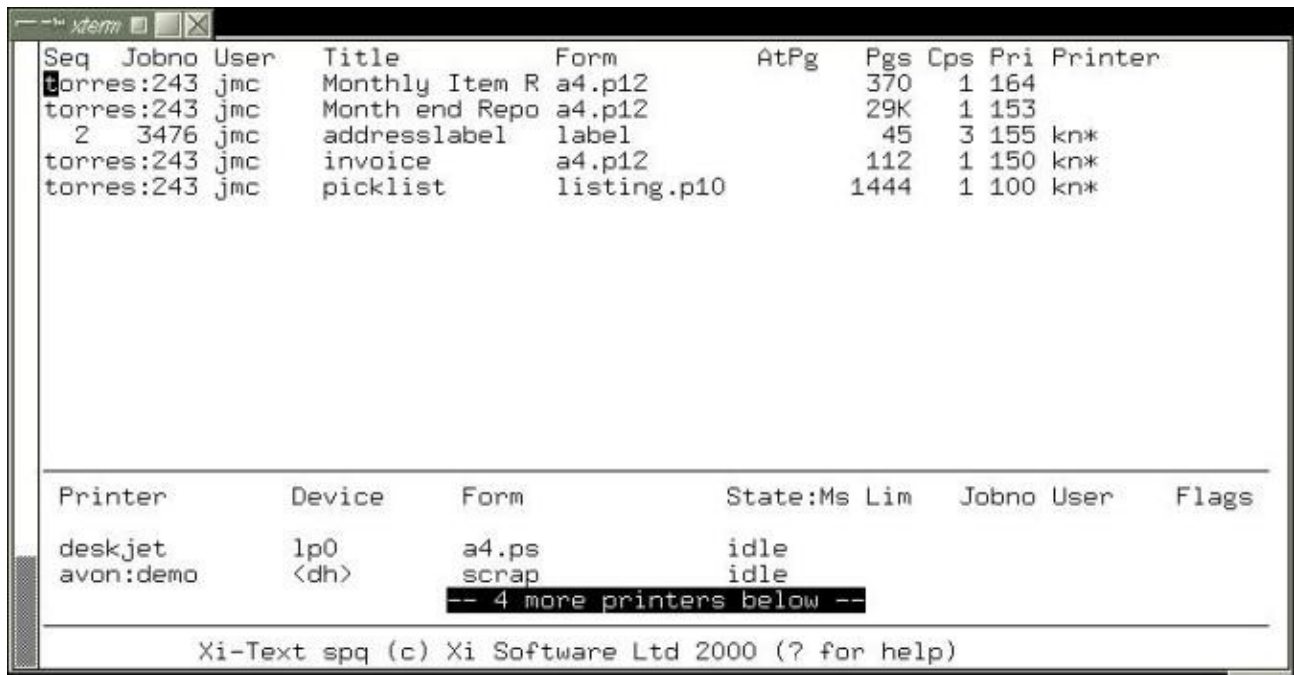
## 10.3 Job and Printer list Formats

The job and printer list formats are described in the System Reference Manual. To quickly create a new layout the format can be re-specified and tested from within `gspl-pq` or `gspl-xpq`. New formats can be saved at any time, for use or further development later.

To illustrate how to achieve the format given at the start of this chapter, let us show the steps.

Firstly create a working directory and change to it. Then start `gspl-pq` to get the standard screen:





The screenshot shows a terminal window with two tables. The first table lists jobs with columns: Seq, Jobno, User, Title, Form, AtPg, Pgs, Cps, Pri, and Printer. The second table lists printer status with columns: Printer, Device, Form, State:Ms Lim, Jobno, User, and Flags. A message at the bottom indicates that 4 more printers are below the visible ones.

Seq	Jobno	User	Title	Form	AtPg	Pgs	Cps	Pri	Printer
torres:243	jmc	Monthly Item R	a4.p12		370	1	164		
torres:243	jmc	Month end Repo	a4.p12		29K	1	153		
2	3476	jmc	addresslabel	label		45	3	155	kn*
torres:243	jmc	invoice	a4.p12		112	1	150	kn*	
torres:243	jmc	picklist	listing.p10		1444	1	100	kn*	

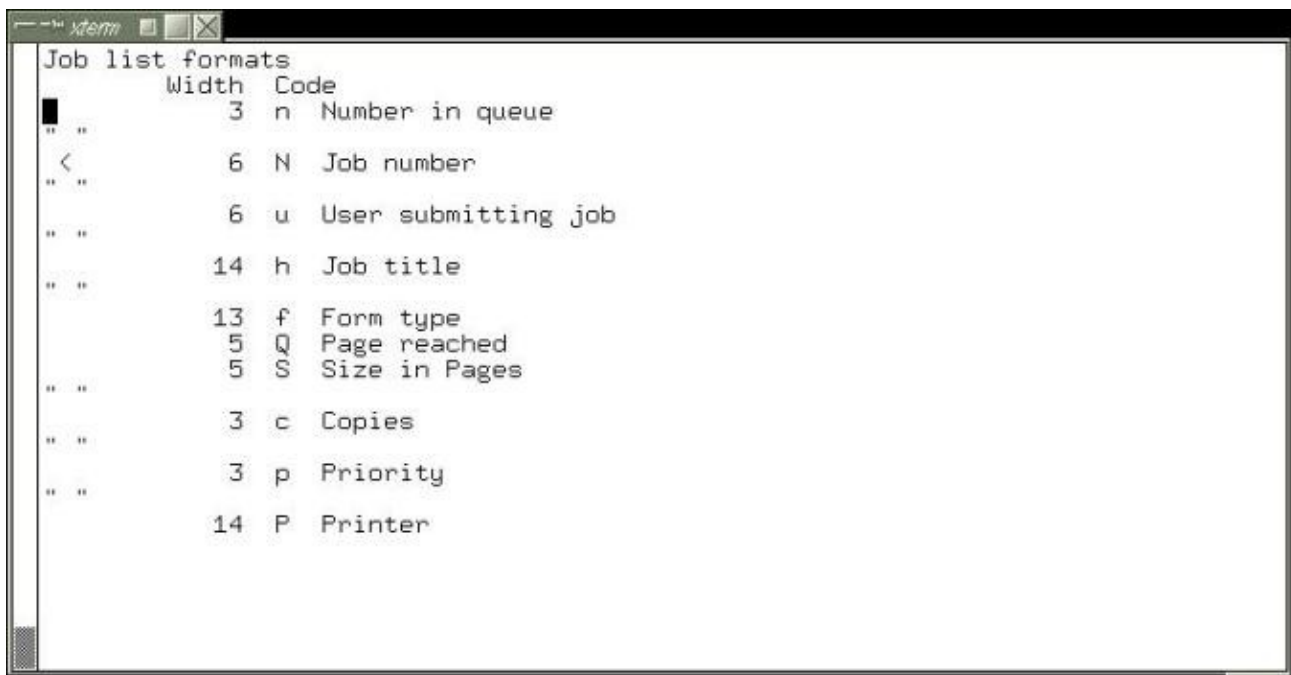
  

Printer	Device	Form	State:Ms Lim	Jobno	User	Flags
deskjet	lp0	a4.ps	idle			
avon:demo	<dh>	scrap	idle			

-- 4 more printers below --

Xi-Text spq (c) Xi Software Ltd 2000 (? for help)

From the job list enter the "," command. This opens the Job list formats screen, showing the current specification:



The screenshot shows the 'Job list formats' screen. It displays a list of job attributes with their corresponding width and code letter. The attributes are: Number in queue (width 3, code n), Job number (width 6, code N), User submitting job (width 6, code u), Job title (width 14, code h), Form type (width 13, code f), Page reached (width 5, code Q), Size in Pages (width 5, code S), Copies (width 3, code c), Priority (width 3, code p), and Printer (width 14, code P).

Width	Code	Attribute
3	n	Number in queue
6	N	Job number
6	u	User submitting job
14	h	Job title
13	f	Form type
5	Q	Page reached
5	S	Size in Pages
3	c	Copies
3	p	Priority
14	P	Printer

Each row in the display corresponds to a column in the job list, showing the width and the code letter which selects the job attribute to be displayed. Many of the fields are separated by spaces.

First let us delete the job number. To do this, move the cursor to the job number line, and press D. The display will look like the following:

```

Job list formats
  Width  Code
    3   n  Number in queue
  " "
    6   u  User submitting job
  " "
   14   h  Job title
  " "
   13   f  Form type
    5   Q  Page reached
    5   S  Size in Pages
  " "
    3   c  Copies
  " "
    3   p  Priority
  " "
   14   P  Printer

```

We don't want two separators, so press **D** again to delete the second separator, **D** again to delete the user name and **D** again to delete the third separator. The screen should now look like this:

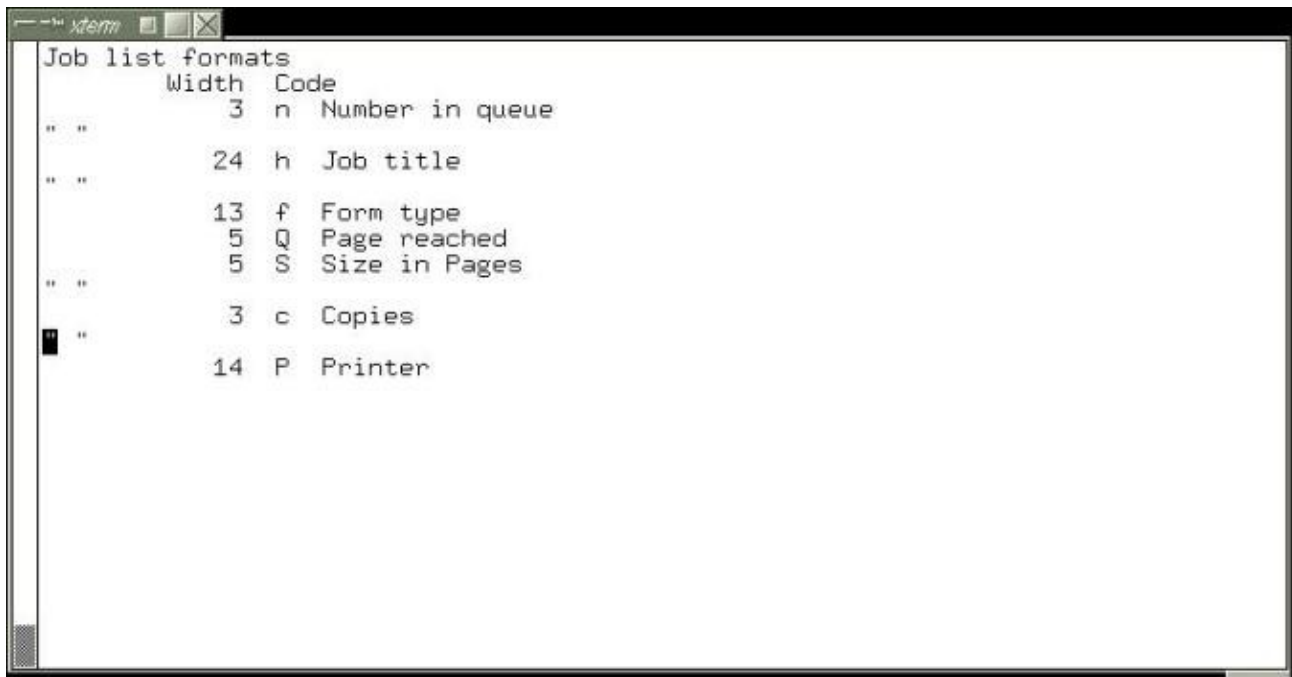
```

Job list formats
  Width  Code
    3   n  Number in queue
  " "
   14   h  Job title
  " "
   13   f  Form type
    5   Q  Page reached
    5   S  Size in Pages
  " "
    3   c  Copies
  " "
    3   p  Priority
  " "
   14   P  Printer

```

Next we can increase the width of the job title field. To do this type **w** and key in a new value, say **24**.

Finally, delete the priority and following space by moving to the field and typing **D** twice. The display should now show:

A screenshot of a terminal window titled 'xterm'. The window displays a list of job list formats. The first line is 'Job list formats'. Below it, there are several lines of text, each representing a format. The text is as follows:

```
Job list formats
Width  Code
  3    n  Number in queue
" "
 24    h  Job title
" "
 13    f  Form type
  5    Q  Page reached
" "
  5    S  Size in Pages
" "
  3    c  Copies
" "
 14    P  Printer
```

Type **q** to quit the screen. The following prompt will appear

Save format codes?

Type **y** to get the prompt

Save in current directory?

Type **y** to get the prompt

What file:

Type in a file name, e.g. **myhelp**, press RETURN and the formats are saved for next time you enter **gspl-pq**.

The display will now look like this:

Seq	Title	Form	AtPg	Pgs	Cps	Printer
0	Monthly Item Report	a4.p12		370	1	
1	Month end Report	a4.p12		29K	1	
2	addresslabel	label		45	3	kn*
3	invoice	a4.p12		112	1	kn*
4	picklist	listing.p10		1444	1	kn*

Printer	Device	Form	State:Ms Lim	Jobno	User	Flags
deskjet	lp0	a4.ps	idle			
avon:demo	<dh>	scrap	idle			
-- 4 more printers below --						

Xi-Text spq (c) Xi Software Ltd 2000 (? for help)

This has now completed the first task, that of setting the job list display.

To reset the printer list format, move to it, and go through the same steps, starting with the "," command, as for jobs. However you will have to specify a different name for the message file, as the current file is not overwritten.

The configuration was saved in a new help file, which was then edited to give the finished result.

The lines in the middle and at the bottom of the screen were also edited using a text editor. By default the specification for the lines in the middle of the screen is this:

```
D1:=====
D2:p
D3:
```

and the specification for the footer lines at the bottom is

```
T1:=====
T2:          GNUspool %P (c) Xi Software Ltd 2000 (? for help)
```

To lighten the dividing line between the jobs and printers the = signs were edited to be ordinary dashes, -. The blank line between the column headings and the printers was removed by deleting the D3: line. The result is:

```
D1:-----
D2:p
```

To put the Function key reminders at the bottom of the screen replace the two original lines with four that look like this.

```
T1:-----F1----F2----F3-----F4-----F5-----F6-----F7-----F8-----F9
T2: job: help list form copies printer options View Summary Quit
T4: ptr: help list form start halt STOP NOW -- Quit
```

Setting up the function keys to actually do something is a separate exercise described

later in this chapter.

### 10.3.1 Adding New fields

New fields, to display other job or printer attributes, and extra separators may be added to the display, in a similar manner to the change and delete options.

The relevant commands are:

i	Insert new field before current row
a	Insert new field after current row
'	Insert new separator before current row
"	Insert new separator after current row
w	Change field width
c	Change code (job or printer attribute to be selected)
<	Toggle left overflow flag (if set, over-long fields may overflow into previous field)
>	Toggle right overflow flag (if set, over-long fields may overflow into the next field).

Asking for help when setting the code will list possible fields. The width is initialised to a default setting for whichever field is selected.

## 10.4 Help & Error Messages

Almost all of the help and error messages output by GNUs pool programs are held in the help files. These messages can be edited with any ordinary text editor.

Each message can be made more informative or shorter. The terminology can be changed to reflect local standards or the whole file can be re-written in a different language. For example, in the `gspl-pq` job list, pressing a key with no associated command produces this error message:

```
Unknown command - expecting job queue control
```

The message is defined in the help file, in an error line which looks like this:

```
E3000:Unknown command - expecting job queue control
```

When one of the programs needs a message it looks down the help file for all lines beginning with a particular code. In this case `E3000` is the relevant error message.

To enhance the message it could be edited and or have additional lines appended. This version offers some constructive advice:

```
E3000:Unknown Key - Expecting a job related command
E3000:You probably pressed the wrong key by mistake
E3000:Enter a ? or press F1 to list commands & keys.
```

## 10.5 Keys & Commands

The keys and commands of the interactive tools `gspl-pq` and `gspl-user` can be re-

configured. One or more keys is mapped to each command by an entry in the relevant help file which looks like this:

```
K400:?
K401:^L
K402:\e
K403:\s,\t
K404:\r
K405:Q,q
K406:k,\kUP
K407:j,\kDOWN
```

or this:

```
1K500:A
1K501:O
1K502:I
1K503:U
1K504:C
1K505:o
1K510:c
```

The entries which start with a letter K are global and those that start with a number apply to a specific sub-screen or option. The component before the `:` specifies the command. The component after the `:` is a comma separated list of key definitions.

### 10.5.1 Specifying Different Keys

An entry for a command may have one or more keys defined. The default key for requesting help is a question mark. In the `gspl-pq` help file it is specified like this.

```
K400:?
```

If the *terminfo* file defines the sequences for the various function keys correctly (not always the case!) the F1 key could be set up for getting help instead of the question mark. Change the entry to look like this:

```
K400:\kF1
```

It is possible that not all of your terminals will support function keys properly. In this case both keys can be specified, like this:

```
K400:?,\kF1
```

### 10.5.2 Disabling Commands

If there is no key defined for a command that command cannot be invoked. Deleting, or preferably converting to comment, the entry in a help file for a key effectively disables that command.

For example the delete job command for `gspl-pq` is specified by the help file entry:

```
1K500:A
```

To disable this command comment the entry out by prefixing it with a `#` like this:

```
# 1K500:A
```

Having disabled a command it is important to edit the help messages to reflect the change. The help associated with delete command for the job list will be adjacent to the key definitions in the help file.

In the case of the delete command the only line that needs changing is from top level help that displays the available commands. This is the line

```
H1:A Abort job
```

If this line contained information about other commands it would need editing. Since it only relates to the delete command it can be commented out:

```
# H1:A Abort job
```

### 10.5.3 Customising Commands

The function of a particular command can be changed by substituting a macro. Macros are described in the chapter on Extensibility. First the existing command must be disabled as described above. Then a macro is set up which is invoked by the same key or keys as the original command.

For example the Delete command could be replaced by a macro which silently unqueues the job to an archive directory. This could be useful for accident prone users.

## 11 Extensibility

There are various mechanisms for enhancing or extending the functionality of GNUspool, 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.

Chapter 8 of the System Reference Manual describes the facilities which are built into the basic product and the Motif GUI option as standard. This chapter gives some example applications of these facilities and how to set them up.

### 11.1 Printer Error Handling

The spooler can call a custom built script or a program whenever a locally defined printer goes into error. To define an error handler a one line entry must be added to the `/usr/local/etc/gnuspool.conf` file. If the program is called `ptr_alert` and is held in the `/usr/local/libexec` directory the line would look like:

```
SPPTRMSGSGS=/usr/local/libexec/ptr_alert
```

The full path for the program must always be specified. This is because it is invoked by the spooler daemon, `spshed`, which does not usually have an appropriate `PATH` environment variable.

Any time that a local printer goes offline or into error the spooler invokes a copy of the program with the following 3 arguments:

1. Name of the printer.
2. Device name.
3. Description or comment.

The program could be an interface to an Alert Management Package or paging system. An example we have on a Sun machine running Solaris 2.5 uses the machine's sound card to give an audible warning. The warnings are spoken messages held in sound files.

The script looks something like this:

```
#
# Set up the path because there is none supplied
# by the spshed daemon.
#

PATH=/usr/sbin:/usr/bin
export PATH

#
# Put a message on the console.
#

echo "\nPrinter $1 needs attention!" > /dev/console
echo "===== \n" > /dev/console

#
```



```
# Play audio files.
#
audioplay -v 85 -p speaker /usr/local/libexec/Alert_begin
audioplay -v 85 -p speaker /usr/local/libexec/Alert/$1
audioplay -v 85 -p speaker /usr/local/libexec/Alert_end
```

The name for the middle file is derived from the printer name. There is a separate file for the name of each printer. If printer Ljet went into error the message might be:

```
<sound effect> Printer Laserjet needs attention
```

This script could be expanded to loop, repeating the message every so often, until the error has been cleared. It would check the printer status using [gspl-plist](#) or [gspl-pstat](#).

## 11.2 Scripts

The command line programs for GNUs pool enable all job, printer and user information to be queried and / or modified. These can be built into new commands using shell scripts or used within user applications.

Here are two examples which use the command line utilities to view and manage the printer audit trail files.

### 11.2.1 Printer Log Files

Printer Definitions include an option to keep an audit trail of all jobs printed. This is specified in the setup file by the keyword `logfile`. A different audit trail can be kept for each papertype. The logfile specification looks like this:

```
logfile=.log
```

In this case the audit trail will be written to a file called `.log` in the Printer Definition directory. An absolute pathname can be specified for a different directory.

#### 11.2.1.1 Viewing a Log File

Here is a shell script ,which given the printer's name, will open the currently active log file for viewing. It uses the `splist` command to get the formtype and from there find the active log file.

This example assumes that all printers have unique names. If they do not the script must be enhanced to take device names as well.

```
#
# Change to the Printer Definition directory
# (the path may be different on your machine)
#

cd /usr/local/share/gnuspools/ptrs/$1

#
# Get the formtype for the printer and extract the
# papertype - which is the same as the setup filename.
#
```

```

PAPER=`gspl-plist -F "%f" $1 | cut -d "." -f1`

# Check to see if there is a formtype specific setup file
# and if not use the file named "default" instead. If neither
# file exist then give an error message and quit.
#

if [ ! -a $PAPER ] then if [ -a default ]
then PAPER=default
else
    echo Bad Printer Definition for $1
    echo -----
    echo Missing Setup Files for formtypes: $PAPER and default
    echo
    echo press return to continue
    read reply
    exit 0
fi
fi

# Extract the name of the logfile from the setup file
# and open it with a suitable editor or listing program.
#

LOGFILE=`grep "logfile=" $PAPER | cut -d "=" -f2`

view $LOGFILE

exit 0

```

Only users with sufficient permissions will be able to access the printer setup and log files. The script is suitable for use as a macro from within [gspl-pq](#) and [gspl-xpq](#).

### 11.2.1.2 Archiving / Purging Log Files

To prevent log files getting too big you may want to archive and/or purge them regularly. Before doing this the printer should be halted. This will stop any data being written to the file during the purge/archive process. If the original file is renamed, it also ensures that the printer daemon picks up the correct file when re-started.

A script could be produced to halt the printer if running, archive/purge the file then re-start the printer if it had been running. The script should allow the current print job to finish before manipulating the log file(s). The procedure might be something like this:

```

#!/bin/ksh -vx
# use gspl-plist to get the printer state and test for it being
# halted, in error or offline.

if [ `gspl-plist -F "%t" $1 | egrep '(halted|offline|error)'` ]
then
    RESTART=NO
else
    RESTART=YES

```

```

# Use sphalt to halt printer at end of job and use splist
# to wait for it to finish.

    gspl-phalt $1
    while [ `gspl-plist-F"%t"$1|egrep'(halted|offline|error)`` ]
    do sleep 5
    done
fi

#
# **** Do log file operations here. ****
#

# Use gspl-start to re-start the printer if required.

if [ $RESTART = "YES" ]
then
    gspl-start $1
    echo starting $1 $RESTART
fi

```

This script could be run automatically at regular intervals or set up as a Macro in [gspl-pq](#) or [gspl-xpq](#) for System Administrators.

## 11.3 Macros

Amongst the different organisations which use GNUspool there are a wide variety of functional requirements. To cater for some of the more individual needs a mechanism has been provided for programming new functionality and incorporating it into the standard queue management tools. These "programs" are called Macros.

Any program which takes a job id number, printer name or set of user names as an argument can be installed as a macro. They can be any executable program, whether a shell script or binary.

### 11.3.1 Publishing Documents for Printing on Demand

GNUspool has a built in document viewer for plain text files. It can also invoke third party viewers for files using printer languages like postscript. This enables users to view a file before printing to see if they really want a hardcopy.

It is possible to submit a job to GNUspool with a specification that prevents it from being printed. Users can view the job, then route it to a printer near them if they want a copy. This could be used as a mechanism for distributing documents.

The following script will take a copy of the original job and submit it for printing at the requesters printer. By taking a copy, conflicts between two or more users asking for a print at the same time are avoided. An environment variable called MYPRINTER must be set up in each users environment to specify their printer.

```

#!/bin/ksh
#

```

```
# Setup unique file name prefixes

CID=$$C
JID=$$J

# Get the title and formtype of the job

GSPL_PR=`gspl-qlist -F '-h "%h" -f %f' $1`
export GSPL_PR

# Unqueue a copy of the job without deleting the
# original. Ignore the command file and use an
# gspl-pr command to submit a new copy.

gspl-qdel -k -u -J $JID -S $CID $1
gspl-pr -P $MYPRINTER ${JID}*$1

# Tidy up all files used.

rm ${CID}*$1 ${JID}*$1
```

The environment variables `CID` and `JID` are used to provide unique filenames for the job and command files. `gspl-qdel` produces files whose names begin with a specified prefix and end with the job number. The prefixes are specified using the `-J` and `-S` options for Job and Command files respectively. The number may have several leading zeros.

Using the process id number in the prefix avoids clashes between two users taking a copy of the job at the same time. Using an `*` pattern between the prefix and significant digits of the job number avoids having to work out how many leading zeros there are.

`GSPL_PR` is the environment variable for default options to `gspl-pr`. A single `gspl-qlist` command is used to set up the formtype and title in `GSPL_PR`.

### 11.3.1.1 Enhancements

This script handles the basic mechanics of copying the job to a pre-defined printer. There are various ways in which it could be enhanced. For example:

1. The user could be invited to select a printer for output. The script should offer a choice of the most suitable printers and check the reply. The formtype and printer restrictions are important parameters for identifying suitable printers.
2. Logging copy requests. Each request for a copy could be logged in a database. Alternatively the originator could be sent an e-mail to notify them of each copy requested.
3. Page selection can be incorporated.
4. A special printer could be set up to work with the original document. This printer could be a program not a hardware device. When a job is submitted to the printer, instead of printing the job it would read the distribution list and e-mail each recipient with details.

5. The printer setup file should contain the `retain` keyword. This is better than having to remember to submit jobs with retention specified.

## 12 Advice & Trouble Shooting

When encountering a problem the following steps should be taken:

1. Save what ever information is possible without endangering the system. Having the exact text of any error messages can be very helpful. Please let us have the last few entries in the system log file, which by default is to be found in `/usr/local/var/gnuspool/spshed_reps`.
2. Make the system safe.
3. Collect as much information as possible. This includes identifying everything involved, (software, hardware and human) what they were doing, when and where.
4. If GNUspool was working and now it is not, look for what has changed. There is always something, it just might not seem relevant.
5. When altering the system to see what happens only change one thing at a time, document it and save the results.

How much information to collect is always a compromise between how much time is available and how severe the problem is. If a problem is hard to recreate or it is intermittent then it is worth the time to collect every possible piece of information.

Before contacting Xi Software Support or your distributor for help you should have the following information ready:

- Machine Model and Operating System
- Version numbers for the copy of GNUspool.
- A copy of any error messages, especially the last few lines in the system log file `/usr/local/var/gnuspool/spshed_reps`.

The following sections describe some of the areas for investigation.

### 12.1 Files & Directories

One of the most common causes of problems for most products and platforms is changes to files, directory structures, ownerships or permissions. The effects of a missing, duplicate or modified file can be instantly fatal or very subtle.

Directories, files, permissions and ownerships should be as follows.

#### 12.1.1 User Program Directory

The permissions of this directory and all of its parent directories must be sufficient for execution of the programs held in it. It is likely that many more files than just the GNUspool user programs will be contained in this directory. The following `ls` command will display the required information for the relevant files:

```
ls -l gspl-*
```

Only the permissions, owner and file name are important. These should be:

Permissions	Owner	File
-rwsr-xr-x	gnuspool	gspl-rpr
-rwsr-xr-x	gnuspool	gspl-lpq
-rwsr-xr-x	gnuspool	gspl-lprm
-rwsr-xr-x	gnuspool	gspl-padd
-rwsr-xr-x	gnuspool	gspl-pchange
-rwsr-xr-x	gnuspool	gspl-charge
-rwsr-xr-x	gnuspool	gspl-conn
-rwsr-xr-x	gnuspool	gspl-pdel
-rwsr-xr-x	gnuspool	gspl-disconn
-rwsr-xr-x	gnuspool	gspl-phalt
-rwsr-xr-x	gnuspool	gspl-plist
-rwsr-xr-x	gnuspool	gspl-nok
-rwsr-xr-x	gnuspool	gspl-ok
-rwsr-xr-x	gnuspool	gspl-pq
-rwsr-xr-x	gnuspool	gspl-pr
-rwsr-xr-x	gnuspool	gspl-start
-rwsr-xr-x	gnuspool	gspl-pstat
-rwsr-xr-x	gnuspool	gspl-uchange
-rwsr-xr-x	gnuspool	gspl-ulist
-rwsr-xr-x	gnuspool	gspl-user
-rwsr-xr-x	gnuspool	gspl-qchange
-rwsr-xr-x	gnuspool	gspl-qdel
-rwsr-xr-x	gnuspool	gspl-sqlist
-rwsr-xr-x	gnuspool	gspl-stop
-rwsr-xr-x	gnuspool	gspl-xpq
-rwsr-xr-x	gnuspool	gspl-xuser

### 12.1.2 The spd Internal Directory

The spd directory must be owned by user gnuspool and be accessible only by the users gnuspool and root. The permissions, owner etc look like this:

Permissions	Owner	File
drwx-----	gnuspool	gnuspool

The contents of the spd directory must all belong to **gnuspool**, but the group ownership is not important. Permissions vary for each file and are very important. The files are:

Permissions	Owner	File
-r-----	gnuspool	SPnnnnnnnnn
-rw-----	gnuspool	ERnnnnnnnnn

```

-rw----- gnuspool PFnnnnnnnnn
-rw----- gnuspool SPDEnnnnnnnnn
-rw----- gnuspool spcharges23
-rw----- gnuspool spmm_jobd
-rw----- gnuspool spmm_jobi
-rw----- gnuspool spmm_ptr
-rw----- gnuspool spmm_xfer
-rw----- gnuspool spshed_jfile
-rw----- gnuspool spshed_reps
-rw----- gnuspool spshed_pfile
-rw----- gnuspool Spufile0.0
drwx----- gnuspool xtlpc
drwx----- gnuspool xtlpd

```

The number on the end of the `spcharges` and `spufile` is the major version number for the installed copy of GNUspool.

If any of these files have the wrong owner or permissions the spooler should be stopped using `gspl-stop -y` before correcting them. Please do not try to change the permissions with GNUspool running.

### 12.1.3 The libexec/gnuspool Internal Directory

The `progs` directory may be owned by user `gnuspool` or possibly user `batch` if Xi-Batch has also been installed, depending on which order they were installed.

Some of the GNUspool files contained in this directory are owned by `gnuspool` and others by `root`. Having the correct owner and permissions is essential to the operation of the scheduler. They are:

Permissions	Owner	File
-rwxr-sr-x	gnuspool	dossppwrite
-rw-r--r--	gnuspool	int-config
-rwxr-xr-x	gnuspool	longlist
-rwxr-xr-x	gnuspool	psbanner
-rwxr-xr-x	gnuspool	ptr_alert
-rwxr-xr-x	gnuspool	remove
-rw-r--r--	gnuspool	rest.help
-rwxr-xr-x	gnuspool	sendudp
-rwxr-xr-x	gnuspool	shortlist
-rwsr-xr-x	gnuspool	spd
-rwxr-xr-x	gnuspool	spdinit
-rwsr-xr-x	root	spexec
-rwsr-xr-x	gnuspool	spjobdump
-rw-r--r--	gnuspool	splpd.help
-rwsr-xr-x	root	spmdisp
-rwsr-xr-x	root	sppwchk



-rw-r--r--	gnuspool	gspl-pq.help
-rwsr-xr-x	root	spshed
-rw-r--r--	gnuspool	gspl-user.help
-rwxr-sr-x	gnuspool	spwrite
-rwsr-xr-x	root	xilp
-rw-r--r--	gnuspool	xmspq.help
-rw-r--r--	gnuspool	xmspuser.help
-rwxr-xr-x	gnuspool	xtelnet
-rwsr-xr-x	root	xtlpc
-rw-r--r--	gnuspool	xtlpc-ctrl
-rwsr-xr-x	root	xtlpd
-rw-r--r--	gnuspool	xtlpd-ctrl
-rwsr-xr-x	root	xtlpq
-rwsr-xr-x	root	xtlprm
-rwsr-xr-x	root	xtnet serv

The files `xmspq.help` and `xmspuser.help` will only be present if the GTK+ or Motif options is installed. The `.help` files can be owned by a user other than `gnuspool` but they must be readable by all users.

The spooler must be stopped with `gspl-stop -y` before fixing owner or permissions of the executable files `spshed` and `spd`. The others can be fixed with the spooler running if it is absolutely essential, but you should check with Xi Software Support first. Using the commands `chmod`, `chgrp` and `chown` should be safe, but any other operation may invalidate the license.

The `.help` files can be corrected without stopping the spooler.

#### 12.1.4 The ptrs Internal Directory

The printers directory should have permissions and ownership looking like this:

Permissions	Owner	File
<code>drwxr-xr-x</code>	<code>gnuspool</code>	<code>ptrs</code>

The Printer Definition directories inside it should have the same permissions and ownerships like this:

Permissions	Owner	File
<code>drwxr-xr-x</code>	<code>gnuspool</code>	<code>ljet</code>
<code>drwxr-xr-x</code>	<code>gnuspool</code>	<code>colour</code>

They may have a different owner, but in this case user `gnuspool` must have sufficient permissions to access the files contained in them. The permissions can be removed for users other than `gnuspool` and `root` if this is required for security.

By default the files in the Printer Definition directories will be owned by `gnuspool` with read permissions for all users. For example:

Permissions	Owner	File
-rw-r--r--	gnuspool	default

## 12.2 IPC Facilities

GNUs pool uses one message queue to communicate with the spooler process, `spshed`. Three shared memory segments (or memory-mapped files) are used to hold records of jobs and printers. These records are written out to the files `spshed_jfile` and `spshed_pfile` respectively, after changes occur. A further shared memory segment or memory-mapped file 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 semaphore set controls access to the shared memory segments.

On versions of GNUs pool subsequent to November 2004, use of shared memory is replaced by memory-mapped files and use of semaphores is replaced by file locking.

### 12.2.1 Looking at IPC Facilities

The IPC facilities can be recognised by running `ipcs`. The items in question are owned by `gnuspool` with a KEY of `0x58691xxx`. Using the command `ipcs -o` should produce an output listing resembling this:

```
# ipcs -o
IPC status from <running system> as of Thu Jan 4 12:27:52 1999
T      ID      KEY      MODE      OWNER      GROUP CBYTES  QNUM
Message Queues:
q       0 0x5869b000 -Rrw----- batch      bin        0        0
q       1 0x58691000 -Rrw----- gnuspool    staff       0        0

T      ID      KEY      MODE      OWNER      GROUP NATTCH
Shared Memory:
m       0 0x50018c83  --rw-r--r-- root       root        1
m       1 0x5869b003  --rw----- batch      bin         4
m       2 0x5869b002  --rw----- batch      bin         4
m       3 0x5869b100  --rw----- batch      bin         4
m       4 0x58691002  --rw----- gnuspool    staff       3
m       5 0x58691003  --rw----- gnuspool    staff       3
m       6 0x58691004  --rw----- gnuspool    staff       3
Semaphores:
s       0 0x5869b001  --ra-ra-ra- batch      bin
s       1 0x5869b003  --ra----- batch      bin
s       2 0x58691001  --ra----- gnuspool    staff
```

In this example there are two third party software products using the IPC facilities, as well a shared memory area in use by the operating system. The lines of interest are the ones showing entries owned by user `gnuspool`. More recent versions of both products will not use shared memory or semaphores.

These are included to show that many users may be seen in the output from `ipcs`. If several products are using IPC facilities there will be more entries displayed than will fit on a screen. The entries appear in the order they were created, as time goes by existing entries will be removed and new ones created. Hence the entries will no

longer be in easy to find blocks.

An easy way to see only the entries owned by `gnuspool` would be to pipe the output from the `ipcs` command through `grep`. For example:

```
# ipcs -o | grep gnuspool
q      1 0x58691000 -Rrw----- gnuspool  staff      0      0
m      4 0x58691002 --rw----- gnuspool  staff      3
m      5 0x58691003 --rw----- gnuspool  staff      3
m      6 0x58691004 --rw----- gnuspool  staff      3
s      2 0x58691001 --ra----- gnuspool  staff
```

The type of each entry is shown in the first column. Message Queues have a letter "q" in the first column, shared memory areas have a letter "m" and semaphores a letter "s".

### 12.2.2 Problem - Cannot Start GNUspool

When GNUspool is halted normally it removes all of the IPC facilities which it is using. If terminated abnormally the spshed processes will still try to tidy up if possible.

The spooler cannot be restarted until all old GNUspool processes have been killed and their IPC entries removed. An IPC entry cannot be deleted if there is still a process using or attached to it. This can be a program, like `xtnet serv`, `gspl-pq` or `gspl-plist`, not just spshed.

Possible Symptoms:

- Invoking `spshed` to start the spooler fails silently.
- Using `gspl-start` to start the scheduler fails with an error message about the IPC facilities, I/O or about files other than ones which are held in the spooler directory.

Investigation:

- If invoking `spshed`, run `gspl-start` instead to get an error message.
- Make sure that the spooler is stopped.
- Look for `spshed`, `xtnet serv` and any user programs with a command like `ps`.
- Check the IPC facilities with the `ipcs -o` command.

Fix:

- Kill any `spshed`, `xtnet serv` and user processes like `gspl-pq`. Do this first with just "kill", not with "kill -9".
- Check the IPC facilities, for entries owned by gnuspool, with the `ipcs -o` command again. Remove any that remain using the `ipcrm` command, or more simply by using the `rmipc -d` command (found in the distribution directory, with the product subdirectory).
- With newer versions of GNUspool, which use memory-mapped files, remove any outstanding undeleted ones with:

```
rm -f /usr/local/var/gnuspool/spmm*
```

- Run `gspl-start` to bring the spooler up.

### 12.2.3 Deleting IPC entries owned by gnuspool

The `ipcrm` command may be used to delete Message Queues, Shared Memory Segments and Semaphores. It uses the same letters to identify Queues, Memory Segments and Semaphores as the `ipcs` command. If running `ipcs -o` gives a listing like this:

```
# ipcs -o | grep gnuspool
q      1 0x58691000 -Rrw----- gnuspool  staff      0      0
m      4 0x58691002 --rw----- gnuspool  staff      3
m      5 0x58691003 --rw----- gnuspool  staff      3
m      6 0x58691004 --rw----- gnuspool  staff      3
s      2 0x58691001 --ra----- gnuspool  staff
```

then suitable `ipcrm` commands to remove each entry would be:

```
ipcrm -q 1
ipcrm -m 4
ipcrm -m 5
ipcrm -m 6
ipcrm -s 2
```

The flag in `ipcrm` indicates what type of IPC entry to delete, using the same letter as in the first column of the output from `ipcs -o`. A message queue is represented with a letter "q" and deleted with a `-q` option.

The number following the flag is an ID number for the queue, memory segment or semaphore. It is unique within the type of IPC, hence there can only be one Message Queue 0, but there can also be one Shared Memory Segment 0 and one Semaphore 0.

Most operating systems will allow more than one entry to be deleted with a single invocation of `ipcrm`. To delete all 5 entries in one go the command would be:

```
ipcrm -q 1 -m 4 -m 5 -m 6 -s 2
```

Be very careful not to delete the wrong IPC entry with `ipcrm`. It is very easy to make an error with system fatal results.

You may prefer to use the `ripc` utility in the installation kit, which diagnoses problems as well as optionally deleting the shared memory facilities.

### 12.2.4 Processes

Some Unix installations have a very low figure specified for the maximum number of processes which may simultaneously be run by a user other than the super-user, `root`. The user `gnuspool` is subject to the same restrictions on numbers of processes as all other users apart from `root`. This may have an adverse effect on the number of printers and associated processes which GNUspool can run.

If this occurs, then GNUspool will become unable to spawn new processes, such as start printers, or send emails or messages. In all cases appropriate messages should

be written to the system log file, normally located in `/usr/local/var/gnuspool/spshed_reps`.

In such cases, please refer to your operating system manual for details of how to tune the limit on the number of processes per user.

As a rough guide to the number of processes allow 6 processes for GNUspool with all printers halted, plus:

- 1 process for each directly connected printer, plus transients when it is started up, or is running alignment process.
- 1 process for each filter command.
- 2 processes for each terminal server printer.

## 12.3 Messages about key clashes entering gspl-pq or gspl-user

If you receive a message like this:

```
State 5 setup error - clash on character ^H with previously-given
value 420 and new value 530
```

It means that your help message file contains 2 definitions for the given character, in this case backspace (^H). You should look in the help message file (in the case of `gspl-pq` this will be by default `/usr/local/share/gnuspoolhelp/spq.help` and `gspl-user.help`) for lines of the form

```
K420: . . .
```

and

```
5K530: . . .
```

The "420", "5" and "530" are all parameters given in the message. You may find that the same character value is defined more than once; remember that (in this example) backspace could conceivably be specified from:

```
^H \b      An explicit character definition
\kERASE    Your terminal key settings for erase character etc
\kLEFT     The left cursor key as defined in terminfo or termcap.
```

The installation procedure attempts to cover cases on the terminal on which you installed GNUspool, but you may run into these difficulties if you use different terminals or key settings.

To overcome the difficulties you may want to rearrange the help message file, or possibly temporarily reassign your terminal keys. Don't forget that you can arrange to automatically select different help message files depending upon the terminal you are using.

## 12.4 Warning messages about unknown key name

If you receive a message on entry to `gspl-pq` or `gspl-user` such as

Unknown key name 'khome' - ignored

then this means that your help message file contains a reference to a key not defined in your *terminfo* or *termcap* file, in this case the HOME key. The effect is harmless; however to get rid of the message you should either adjust the file (*spq.help* or *spuser.help*) to remove the reference, in this case to `\kHOME`, or to replace it with the correct character sequence, alternatively you should include the definition in your *terminfo* or *termcap* file.