

CORE Computing Services

CORE Physics Services

SHIFT User Guide

Jean-Philippe Baud, Fabrizio Cane, Felix Hassine, Frédéric Hemmer

Gordon Lee, Les Robertson, Ben Segal, Antoine Trannoy

Version 1.3
8th March 1993

CERN Geneva, Switzerland

Contents

1 Introduction **1**

2 Overview of the SHIFT Architecture **3**

2.1 Scalability 4

2.2 Software Portablity 5

2.3 Software Components in SHIFT 5

2.3.1 Disk Pool Manager 5

2.3.2 Tape Copy Scheduler 6

2.3.3 Remote File I/O System 6

2.3.4 Magnetic Tape Support 7

2.3.5 Remote Tape Copy Utility, RTCOPY 7

2.3.6 Network Queueing System 8

2.3.7 Accounting Facilities 9

3 The SHIFT Configuration at CERN **10**

3.1 OPAL CPU and File Servers 10

3.2 ALEPH CPU and File Server 10
3.3 SMC CPU Servers 11
3.4 SHIFT Fileservers 11
3.5 Backbone Network - UltraNet 11
3.6 CORE Infrastructure - Tape Servers and Home Directories 12

4 Getting Started with SHIFT 14

4.1 Getting an Account 14
4.2 Passwords 14
4.3 Login and Default Shell Environment 15
4.4 Getting Online Help and Documentation 15
4.5 Printing files on SHIFT 16

5 The SHIFT System Environment 17

5.1 `System Disks` 17
5.2 /usr/local 17
5.3 Home Directories 18
5.4 POOL Filesystems 19

6 The SHIFT User Environment 20

6.1 Home directories 20
6.2 Shell Startup Files 20
6.3 CORE Service Classes 21

6.4 SuperComputer Access 21

6.5 CORE Environment Variables 21

6.5.1 ARCHITECTURE Variable 21

6.5.2 SERVICE Variable 22

6.5.3 VENDOR Variable 22

6.6 Command Search Path 22

6.7 Registration and Accounting 23

6.8 Network Access 24

6.8.1 File transfer 24

6.9 CERNLIB - CERN Program Library 25

7 Batch Job Submission using NQS **26**

7.1 NQS - An Overview 26

7.2 NQS Commands 27

7.2.1 NQS Information Commands 28

7.3 Some Notes on using NQS on SHIFT 28

8 Using SHIFT Software **29**

8.1 Disk Pools and Files 29

8.2 Using shift Files with FORTRAN Programs 32

8.3 File Permissions and Authorisation 32

8.4 Tape-Disk Copying 35

8.5 Tape Staging 36

8.5.1 stagein 37

8.5.2 stageout 37

Chapter 1

Introduction

The term `shift`, the `Scalable Heterogeneous Integrated Computing Facility`, is applied to three distinct, though related, entities:

- an `architecture` whose components include disk servers, cpu servers and tape servers, together with some special-purpose software;
- a `cern project` - a collaboration between the OPAL experiment and CN Division to build a pilot implementation of the architecture and use it for physics data analysis;
- a `service`, including hardware, software and people, installed in the `cern` Computer Centre and operated by CN Division. `shift` is now part of CORE[1] which is a group of RISC based services managed by a single team.

This `User Guide` describes the fundamentals of the `shift` architecture and the current configuration of the implementation in the `cern` Computer Center. It also gives a brief description of the special `shift` facilities with a number of examples of how to use them. The separate `Reference Manual` contains detailed specifications of the various `shift` commands and functions.

The operating system used by `shift` is Unix^{TM}, in the flavours best supported on the various hardware components. This manual assumes some familiarity with Unix. Furthermore, the SHIFT software has been designed following the Unix philosophy of providing small, simple functions, which can be used in building many different applications.

[1] CORE - Centrally Operated RISC Environment

The SHIFT development was a collaborative effort between members of CN division and the OPAL experiment. The project manager is `Les Robertson` and others who made a contribution include `Jean-Philippe Baud, Christain Boissat, Fabrizio Cane, Felix Hassine, Frederic Hemmer, Erik Jagel, Ashok Kumar, Gordon Lee, Bernd Panzer-Steindel, Ben Segal, Antoine Trannoy, and Igor Zacharov.`

Chapter 2

Overview of the SHIFT Architecture

The designers of `shift` were motivated by the appearance on the market of inexpensive processors and storage systems, using technology developed for personal workstations, and which had performance characteristics comparable with those of traditional mainframes.

The goal was to define an architecture which could be used for general purpose High Energy Physics (`hep`) computing, could be implemented to provide systems with an excellent price/performance ratio when compared with mainframe solutions, and could be scaled up to provide very large[1] integrated facilities, or down to provide a system suitable for a small physics department.

Some important characteristics of `offline hep` processing relevant to the the design choices are:

- the volume of data which must be held on online storage (up to the order of 10^{12} bytes);
- the need for access to magnetic tapes, used to store fuller information about "events" (a few terabytes);
- difficulty in finding vectorisable algorithms for a significant fraction of the processing requirements (hard to exploit supercomputers);
- inherent parallelism in much of the processing (events are largely independent);

[1] compared with the capacity of current `hep` Computer Centres.

- synchronisation of read/write file access is performed at an application level (essentially: data files are not modified, rather new copies are created, with access organised through high-level packages such as FATMEN[2]);

- use of standard `hep` packages[3] for accessing data from FORTRAN programs. This makes it possible to replace the underlying basic `i/o` system by means of application-level libraries.

The SHIFT system consists of sets of `cpu servers, disk servers` and `tape servers`. Files are stored in `pools`, each pool comprising one or more Unix file systems residing on one or more `shift` nodes. Access to the disk pools and files is coordinated by the `Disk Pool Manager`.

2.1 Scalability

A prime goal of the SHIFT project was to build facilities which could scale in capacity from small systems up to several times that of the current CERN computer center. To achieve this, an architecture was chosen which encouraged a separation of functionality. Each component of the system has a specific function, CPU server, disk server, tape server and this modular approach allows the whole facility to be extended in a flexible manner.

The servers are interconnected by the `backplane,` a very fast network medium used for optimized special purpose data transfers. TCP/IP protocols are supported across the backplane and connection to CERN's general purpose network infrastructure is by means of an `IP router.` This provides access to workstations distributed throughout CERN and at remote institutes.

The components of SHIFT are controlled by distributed software which is responsible for managing disk space, staging data between tape and disk, batch job scheduling and accounting.

---

[2] `FATMEN Distributed File and Tape Management System`, J.Shiers, J.Boucrot, L.Palermo, V.Innocente and J.Goldberg, `cern` Program Library Writeup Q123

[3] such as `ZEBRA: Dynamic Data Structure and Memory Manager`, R.Brun, M.Goossens and J.Zoll, `cern` Program Library Writeup Q100

2.2 Software Portablity

Software portablity was an important aspect of the SHIFT development for two reasons:

- It allowed flexibility in the choice of hardware platform for each system component.
- It is important to be able to benefit from the continuous and rapid progress being made in hardware technology.

Addition of further system types to the existing configuration is regularly reviewed and no major difficulties are foreseen in incorporating any Unix based systems.

2.3 Software Components in SHIFT

The main software components of SHIFT are as follows.

2.3.1 Disk Pool Manager

The SHIFT filebase comprises many Unix filesystems located on different SHIFT server systems. In order that users see a unified data file space, the notion of a `pool` was created. A `pool` is a group of one or several Unix filesystems and it is at the `pool` level that file allocation is made by the user. Pools can be much larger than conventional Unix filesystems even where logical volumes are available. Pools may also be assigned attributes. For example, a pool used for staging space can be subject to a defined garbage collection algorithm.

The pools in SHIFT are all managed by the `Disk Pool Manager.` It is responsible for disk space allocation when creating new files/directories and it may be used to locate and delete existing files.

The interface to the `Disk Pool Manager.` is via Unix user commands. The `sfget` command allocates a file of a given size within a specified pool. The command returns a full path name for the file based on the convention that all SHIFT file systems are mounted globally with NFS on the mount point

```
/shift/<host name>
```

If the file requested already exists within the pool, `sfget` simply returns the path name without allocating space. Other commands are provided to list, remove and manage files.

In addition, a user-callable garbage collector has been implemented which maintains defined levels of free space within a pool. This is useful for physics data staging where data are copied from tape to disk before being accessed by user programs.

2.3.2 Tape Copy Scheduler

The tape copy scheduler organises the copying of data between `shift` disk files and magnetic tapes. On request from a user through a `tpread` or `tpwrite` command, it selects an appropriate tape server depending on the device type required, location of the tape and current tape activity. It then initiates a tape copy using the `cptpdsk` or `cpdsktp` program on the tape server node. The tape copy scheduler informs the user when the operation is complete, queues concurrent requests for the same tape, and deals with error recovery. The tpread and tpwrite commands are described in the SHIFT Reference Manual.

2.3.3 Remote File I/O System

The Remote File I/O system (RFIO) provides an efficient way of accessing remote files on SHIFT. Remote file access is also possible using `nfs`[4], but RFIO takes account of the network characteristics and the mode of use of the files to minimize overheads and maximize throughput. RFIO maintains portability by using only the BSD socket interface to TCP, and thus operates over UltraNet, Ethernet, FDDI or other media. RFIO transmits I/O calls from client processes to remote RFIO daemons running on all SHIFT hosts.

RFIO is implemented with both C and FORTRAN interfaces. In C, the system presents the same interface as local Unix I/O calls: `rfio_open` opens a file like `open(2)`, `rfio_read` reads data from a file like `read(2)` etc.

[4]SUN Microsystem's `Network File System`.

Most High Energy Physics programs are written in FORTRAN, and usually interface their I/O via one or two intermediate library packages. RFIO has been incorporated into these, so its usage is completely transparent to the users of these programs. RFIO performance is a key factor in SHIFT and a major effort was made to reduce the operating system overheads.

2.3.4 Magnetic Tape Support

A portable Unix Tape Subsystem was designed to satisfy all SHIFT's requirements in the area of cartridge tape access. The subsystem runs on all SHIFT hosts to which tape devices are connected.

Unix systems usually offer a primitive tape interface which is ill adapted to a multiuser environment. Four basic functions are typically provided:

```
open(2), read(2), write(2), close(2)
```

Several `ioctl(2)` commands are also provided but there is no operator interface, label processing, or interface to a tape management system. The SHIFT Tape Subsystem offers dynamic configuration of tape units, reservation and allocation of the units, automatic label checking, an operator interface, a status display and an interface to the CERN/Rutherford Tape Management System. It is written entirely as user code and requires no modification of manufacturers' driver code. It currently supports StorageTek's 4280 SCSI tape drive (an IBM 3480 compatible), Exabyte 8200/8500 drives and DAT (on HP only).

Tape file labelling is provided by a set of user callable routines. In practice, most tape I/O is done by using a tape staging utility, RTCOPY which hides details of these routines from the user.

2.3.5 Remote Tape Copy Utility, RTCOPY

To provide tape access for every SHIFT CPU and disk server, a tape copy utility RTCOPY was developed which allows tape access across the network. Internally RTCOPY uses RFIO software to maximize the data transfer speed and thus minimize the tape unit allocation time. RTCOPY intelligently selects an appropriate tape server, by polling all known tape servers to query the status of

their tape unit(s). RTCOPY supplies any missing tape identification parameters by querying the Tape Management System as needed. RTCOPY then initiates the tape copy, informs the user when the operation is complete, and deals with error recovery.

2.3.6 Network Queueing System

The Network Queueing System, `nqs`, is a distributed batch scheduler originally developed for NASA and which is now in the public domain.

NQS is used in SHIFT for job submission and scheduling across a network of Unix batch workers. At CERN, the public domain version of NQS has been ported to numerous workstation platforms and useful enhancements added. These include limits on the number of jobs run for any user at one time, an interactive global run limit, the ability to move requests from one queue to another and the ability to hold and release requests dynamically. In addition CERN has implemented in NQS the ability to have the destination server chosen automatically, based on relative work loads across the set of destination machines.

Users submit jobs to a central pipe queue which in turn chooses a destination batch queue or `initiator` on the least loaded machine that meets the jobs' resource requirements. If all initiators are busy, jobs are held in the central pipe queue and only released when one becomes free. A script running above NQS holds or releases waiting jobs with a priority based on their owner's past and current usage of the SHIFT service.

An NQS command summary is given below.

qcat	Display output files of NQS running requests.
qcmplx	Display status of NQS queue complexes.
qdel	Delete or signal NQS requests.
qhold	Hold NQS requests.
qjob	Display status of NQS networked queues.
qlimit	Show supported batch limits and shell strategy.
qrls	Release NQS requests
qstat	Display status of NQS requests and queues.
qsub	Submit an NQS batch request.
qmgr	NQS queue manager program

2.3.7 Accounting Facilities

Several utilities have been developed to provide resource consumption reports and to monitor usage of the systems. At weekly intervals, a SHIFT accounting report is returned to the central CERN database. This report shows CPU usage on all SHIFT systems both by individual user and by user group.

Chapter 3

The SHIFT Configuration at CERN

3.1 OPAL CPU and File Servers

The CPU power for OPAL within SHIFT is provided by four `Silicon Graphics` Power Series systems, each with system has 4 MIPS3000 processors rated at 33 MHz. The systems have 64 MBytes of memory, an ethernet interface, and a Power channel which provides 2 single ended SCSI channels. Additional differential SCSI channels are provided by Interphase `Jaguar` boards which connect via the VME card cage in the SGI systems. A VME interface board is also used to connect to Ultranet.

The OPAL systems are named SHIFT1, SHIFT2, SHIFT3 with a total CPU capacity of 72 CERN units. In addition, the SHD01 system which is owned by CN division and used as a combined file/CPU server provides a further 24 CERN units.

3.2 ALEPH CPU and File Server

The ALEPH collaboration use a single `Silicon Graphics` Power Series system with 4 MIPS3000 processors rated at 40 MHz. The system has 64 MBytes of memory, an ethernet interface, and a Power channel which provides 2 single ended SCSI channels. Additional differential SCSI channels are provided by Interphase em Jaguar boards which connect via the VME card cage in the SGI systems. A VME interface board is also used to connect to Ultranet.

The ALEPH system is named SHIFT6 and delivers a CPU capacity of 32 CERN

units.

3.3 SMC CPU Servers

The Spin-Muon Collaboration use two SUN systems: a twin processor SUN630 and a SPARCstation-10. Both systems are networked to ethernet and SHIFT4 is connected to UltraNet. The SUN systems are named SHIFT4 and SHIFT5 and deliver a CPU capacity of 18 CERN units.

3.4 SHIFT Fileservers

Two `Silicon Graphics` CRIMSON systems are used solely as file servers. These are single processor MIPS4000 systems rated at 50 MHz which can support up to to six differential SCSI channels in addition to an UltraNet VME board.

The SHIFT filebase is distributed across the `Silicon Graphics` systems and is made up of SEAGATE WREN disks. Most are either WREN-7 of 1.1 MB capacity or WREN-8 of 1.5 MB capacity. Recently, WREN-9 units have been purchased with a formatted capacity of 1.8 MB. The disks are rack mounted as much as possible and connected to the system units via Differential SCSI interface boards, each interface providing two SCSI channels. The differential interfaces allow the disk trays to be located up to 20m from the Power Series systems which facilitates the organisation of the equipment within the computer center. The disks are grouped in trays of either four or six disks and each tray is built as a UNIX logical volume with an associated filesystem.

A detailed description of the current SHIFT configuration is maintained in the document `Systems Administration for CORE.` Copies of this document are available on request from Gordon Lee in CN division.

3.5 Backbone Network - UltraNet

`UltraNet` is a special purpose network running over co-axial cable or optical fiber. Machines are connected to a central hub and hubs can be interconnected to form a

star topology. For the SHIFT systems, the UltraNet connection is via a VME interface board.

The TCP/IP protocols are supported by Ultra and by performing a high level of protocol processing on the PP's, the CPU consumption for high data rate transfers is kept to a minimum. `UltraNet` is designed to be effective for `stream-type` transfers and this maps well on to High Energy Physics applications where accesses are sequential and use large record lengths. The coaxial cables used in UltraNet can support data transfer rates of 256 Mbit/second. In the SHIFT environment, the average data transfer rate is between 4 and 8 MBytes/second.

3.6 CORE Infrastructure - Tape Servers and Home Directories

The SHIFT tape staging facilities form an important part of both SHIFT and CSF giving access to IBM 3480 and Exabyte cartridges. The 3480 cartridge media offers high data transfer rates, up to 2 MBytes/second and excellent levels of data integrity. The 3480 cartridges have a capacity of 200 MBytes but the high density versions, 3490 can hold up to 800 MBytes of data. Exabyte cartridges are also used at CERN but to a smaller extent. These have a capacity of either 2 or 5 Gigabytes but with a data transfer rate substantially lower than that of the 3480 cartridge.

The 3480 and Exabyte units are located in the CERN tape vault and are connected to SUN-4/330 systems. The connection is made a SCSI extension product making use of single-ended to differential SCSI converter boards. 3480 cartridges held in the central IBM robot are accessed via tape units connected to the CRAY UNICOS system. The SHIFT tape facilities currently stage a total of about 1500 tapes per week, with about half being mounted automatically through the IBM robot.

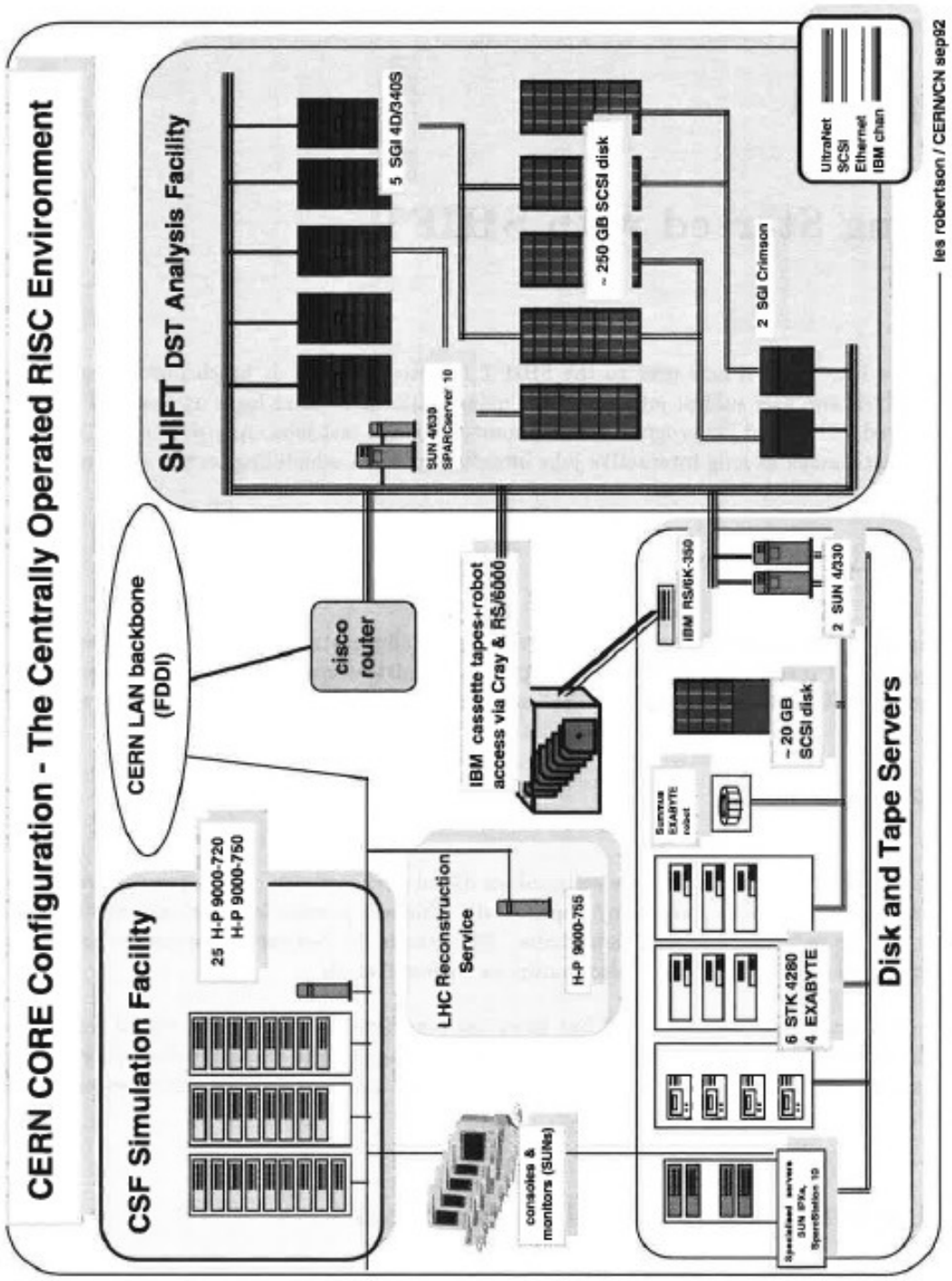


Figure 3.1: SHIFT Configuration at CERN

Chapter 4

Getting Started with SHIFT

This chapter introduces a new user to the SHIFT facilities. SHIFT is batch processing facility for physics analysis and user submit jobs to SHIFT using NQS. Interactive login on the SHIFT machines is also allowed and is used for program development and small test jobs. Any production work should be run in batch mode as long interactive jobs interfere with job scheduling on the systems.

4.1 Getting an Account

SHIFT accounts are allocated through the normal group administrators and you you first check that your group has CPU resources on SHIFT. The group administrator uses USERREG exec on CERNVM and enter the SHIFT panel. Please note that only `one` group code is allowed per UNIX username.

4.2 Passwords

Upon registration for SHIFT, users are assigned an initial password. To change your password use the command `passwd` (a link to `/usr/bin/yppasswd`). This will prompt for your current NIS password and then ask for two copies of your new choice. NIS stands for Network Information Service and is the registry used on SHIFT. (NIS is also known as `Yellow Pages`).

There is no password ageing on SHIFT but users are recommended to use

`non-trivial` passwords. At least 6 characters should be used and at least one non alphanumeric character should be included in the password. Avoid any dictionary word and any simple name associated with you or your login identifier.

4.3 Login and Default Shell Environment

To login to SHIFT, users should `telnet` or `rlogin` to an appropriate system: SHIFT1 in the case of OPAL, and SHIFT6 in the case of ALEPH. User accounts are setup with default `.cshrc`, `.profile` and `.login` files. All groups are also setup with a pub`gg` directory with default `group_cshrc` and `group_profile` files.

The default shell is generally the `csh` but the in case of ALEPH, `tcsh` is used. Users who have their own workstations may want to start up an xterm on their own display from csf. To do this, type

```
xhost shift1
```

on your own workstation. Then login into your SHIFT node and type

```
/usr/bin/X11/xterm  -display yourhost:0 &
```

4.4 Getting Online Help and Documentation

- All man pages are installed on SHIFT. To get a full description of a command, type

  ```
  man command.
  ```

- To get a list of commands related to a particular keyword, type

  ```
  apropos keyword.
  ```

- To discover where commands are located, type

  ```
  whereis command
  ```

4.5 Printing files on SHIFT

The UNIX `lpr/lpd` line printer spooling system is installed on the SHIFT. Printers known to the system are defined in the file `/etc/printcap`

To print a file, type the following command

```
lpr -P <printer> <filename>
```

Chapter 5

The SHIFT System Environment

5.1 System Disks

Each `shift` node has a system disk with traditional UNIX filesystems, `/` (root) and `/usr`, mounted from separate disk partitions. Each node has a backup system disk which can be used for booting in the event of a disk failure. The backup disk is cloned periodically from the primary system disk so as to propagate system changes.

5.2 /usr/local

By convention, the `/usr/local` filesystem contains all software and libraries that have been installed at CERN. This includes the CERN library, the SHIFT software, and a number of utility packages. There is a single copy of `/usr/local` which is shared across the `shift` nodes using `nfs` (Network File System) `/usr/local` has several sub-directories:

dimension

`src:` source code

`lib:` libraries

`bin:` executable programs and scripts

`etc:` Systems administration scripts and startup files

5.3 Home Directories

A user's home directory on SHIFT is defined by an entry in the `/etc/passwd` file and it is created when the user is registered. SHIFT home directories have the generic form:

`/u/gg/username`,

where `gg` is the 2-character group code and `username` is the user's login identifier.

The physical disk space used for the `/u` directory is made up of several separate filesystems, which are mounted as `/u1, /u2` as shown in the Table below:

Filesystem	Machine	Size	Usage
/u1	COREHOME	4.8GB	Home directories for CSF/SHIFT users
/u2	SHIFT1	2.8GB	SHIFT home directories for OPAL
/u3	SHIFT3	2.3GB	SHIFT home directories for ALEPH

In order to maintain a coherent naming convention, a series of symbolic links is maintained in the `/u` directory in all CORE systems. For example:

```
/u/c3  --->  /u1/c3
/u/ct  --->  /u1/ct
/u/ws  --->  /u2/ws
/u/xu  --->  /u3/xu
```

This shows that the home directory space for Group C3 is part of the `/u1` filesystem residing on COREHOME whilst that of Group WS is part of `/u2` located on SHIFT1. Home directories allocated on the COREHOME system are subject to disk quotas which control space allocation. To have your quota increased, contact one of the SHIFT systems adminitrators.

For client systems, the /u directory may be a local directory or may be mounted via NFS from one of the central servers such as COREHOME.

5.4 POOL Filesystems

In the `shift` terminology, a `disk pool` is a group of filesystems located on one or several disk servers. Each of these filesystems is mounted by `nfs` on every `shift` node using a standard naming convention, the `nfs_pathname`:

`/shift/node/file_system_name/group/user/user_supplied_partial_pathname`

where `node` is the machine on which the filesystem is physically located.

Chapter 6

The SHIFT User Environment

6.1 Home directories

A user's home directory on SHIFT has the generic form:

`/u/gg/username`,

where `gg` is the 2-character group code and `username` is the user's login identifier.

The organisation of the home directories is described in more detail in the previous on the SHIFT System Environment.

6.2 Shell Startup Files

Upon registration, a SHIFT user is supplied with SHELL startup files. For interactive work in SHIFT, the C-shell /bin/csh or /bin/tcsh are used almost universally and C-shell initialisation is performed at 3 levels for both .login and .cshrc:

```
user    /u/gg/user/.login
group   /u/gg/pubgg/group_login
system  /usr/local/etc/system_login

user    /u/gg/user/.cshrc
group   /u/gg/pubgg/group_cshrc
```

```
system  /usr/local/etc/system_cshrc
```

The .login and .cshrc files in a user's home directory first look for a system level file followed by the group level. The group level files are administered by responsible persons within the particular group/experiment.

6.3 CORE Service Classes

CORE comprises a number of distinct but related services. Some are shared like CSF whereas other a private to a particular experiment. A user must be registered for each `service class` within CORE that he wishes to use.

When a user accesses a CORE system, checks are carried to ensure that the user is authorised for the particular service to which the system belongs. On sucessful login, an ENVIRONMENT variable `SERVICE` is set to indicate the name of the CORE service being used (See Section 2.6 below).

6.4 SuperComputer Access

Some CORE systems are classed as `supercomputers` and access to these is restricted for nationals of certain countries. In an similar manner to the service checks, supercomputer access rights are checked at login.

6.5 CORE Environment Variables

The system level shell initialisation file defines four ENVIRONMENT variables which are listed below together with possible values. Note that the variables and their values are in UPPER case.

6.5.1 ARCHITECTURE Variable

This defines the machine architecture.

ARCHITECTURE	
HPPA	HP 9000 Precision Architecture
MIPS3000	Silicon Graphics MIPS 3000
MIPS4000	Silicon Graphics MIPS 4000
SUN4	SUN Sparc
ALPHA	DEC Alpha

6.5.2 SERVICE Variable

This defines the CORE service class.

SERVICE	
SHIFTOPAL	SHIFT analysis for OPAL
SHIFTALEPH	SHIFT analysis for ALEPH
SHIFTL3	SHIFT analysis for L3
SHIFTDELPHI	SHIFT analysis for DELPHI
CSF	CSF simulation service
ATLAS	ATLAS Reconstruction Service
CMS	CMS Reconstruction Service
CPLEAR	CPLEAR analysis system

6.5.3 VENDOR Variable

This defines the machine vendor.

VENDOR	
HP	Hewlett Packard
DEC	Digital Equipment
SGI	Silicon Graphics
IBM	IBM
SUN	SUN

6.6 Command Search Path

On UNIX systems, the command search path is defined by the PATH environment variable. The PATH variable consists of a list of directory files which contain

executable programs and shell script. The current setting of the PATH variable may be displayed by the command

```
echo $PATH
```

On `shift` several directories have been added to the path in addition to the standard directories such as `/bin` and `/usr/bin`.

- `/usr/bin/X11` X-Window application programs
- `/usr/local/bin` UNIX utility programs installed locally
- `/cern/pro/bin` CERNLIB executables

6.7 Registration and Accounting

User registration for `shift` accounts is handled by `cern` group administrators using the USERREG script on CERNVM. The registration procedure ensures that the user is allocated a unique UID corresponding to his login identifier. If the user already has an account on one of the central UNIX systems, then the same login identifier will be used in `shift`. The UID code determines file ownership on UNIX systems and coherence across different systems at `cern` is essential.

Once a user is registered for `shift`, he is able to login on any on the `shift` systems and user home directories are visible across the whole `shift` domain. Accounts on `shift` nodes are managed using the `Network Information System` or Yellow Pages. The system `COREYP` is the Yellow Pages Master Server, where updates are made and other nodes act as slave servers.

A list of registered `shift` users is generated each night and transferred to CERNVM for correlation with the central database.

System accounting reports are produced each day and a shell script correlates the user accounting data with the `cern` group codes. Each week, a summary accounting report is transferred to the accounting database on CERNVM. Detailed accounting information is maintained for a certain period on each node of the `shift` configuration.

6.8 Network Access

Three different Network media are used in the `shift` project: Ethernet, FDDI Ring, and UltraNet. Initially, all nodes have Ethernet and UltraNet connections and the FDDI connections may be added where required. Each network interface on a `shift` node must be given a separate `Internet` address.

The primary interface of each node, currently Ethernet, is identified by the hostname and has an Internet number in the officially registered Class B Network for CERN, network number `128.141`. Machines on the `cern` Class B network have officially registered names and may be visible from networks outside the `cern` site. Class C networks are internal `cern` networks which are unknown to the outside world. Class C nodes are reached from the Class B network via Gateway machines.

Internet addresses for the secondary interfaces are generated by adding prefixes or suffices to the hostnames as follows: dimension

`hostname-u:`
: The name used for UltraNet access and is a `subnet` of the CERN Class B network.

`hostname-uip:`
: Used for special UltraNet applications which create their own IP packets. It is a separate Class C network.

`f-hostname:`
: Currently, the FDDI ring connections use a separate Class C network. This will change in the future and FDDI connections will become the primary interface to the CERN Class B network.

6.8.1 File transfer

When using the file transfer commands such as `ftp` and `rcp,` the transfer medium can be selected by choosing the appropriate nodename. For example, when logged in to `shift1`, the command

```
rcp shd01-u:data.f data.f
```

will transfer the file data.f using UltraNet whereas

```
rcp shd01:data.f data.f
```

will make the transfer using Ethernet.

6.9 CERNLIB - CERN Program Library

The location of the CERN Program Library on `shift` systems is given by the 3 environment variables:

```
CERN       = /cern
CERN_LEVEL = pro
CERN_ROOT  = /cern/pro
```

On the `shift` Silicon Graphics nodes, the library is located in `/usr/local/cern`, and the file `/cern` is a symbolic link to this directory.

Chapter 7

Batch Job Submission using NQS

7.1 NQS - An Overview

The Network Queueing System, `nqs` has been installed on the SHIFT systems and is used for batch job submission. The SHIFT implementation of the public domain version of `nqs` is described here, together with an overview of the `shift` batch environment. For a detailed description of the `nqs` commands `qstat, qsub, qlimit` and `qdel`, the reader is referred the on-line man pages.

An `nqs` job is a series of UNIX commands combined in a SHELL script. `nqs` works from job queues and the actual queues names and resource limits are different for each SHIFT user group. The `nqs` queue manager, `qmgr` is used by the system administrator to change queue parameters and to set resource limits.

In general, `nqs` distinguishes between 2 types of queue: `pipe` queues and `batch` queues. Pipe queues are a mechanism to distribute jobs and balance the workload evenly over the `batch` queues on the destination servers. Users submit to the pipe queue and load balancing software in `nqs` finds an empty server on which to run the job. In the SHIFT environment, several `pipe` queues are created, each with different CPU resource limits and the jobs from the different queues run at different priorities.

`nqs` commands are located in the directory `/usr/bin` and in `/usr/local/bin.` Users submit batch requests with the `qsub` command. Batch requests are SHELL scripts which can be executed independently of any user intervention. A job should be submitted to the queue most suited to its resource needs, principally based on

required cpu time. If no queue is specified, a job runs in the default queue, which has very low priority. A job's status can be checked with the command `qusage`. A job may be stopped or removed from a queue with the `qdel` command.

All `nqs` commands work across machines within the `nqs` domain. [1] A user need not log into the destination batch machine to submit or delete jobs or to check a queue's status. Instead, he may execute commands from any system in the `nqs` domain.

7.2 NQS Commands

NQS provides a total of nine user commands:

```
qcat:         display output files of NQS running requests.
qcmplx:       display status of NQS queue complexes.
qdel:         delete or signal NQS requests.
qhold:        hold NQS requests.
qjob:         display status of NQS networked queues.
qlimit:       show supported batch limits and shell strategy.
qrls:         release NQS requests
qstat:        display status of NQS requests and queues.
qsub:         submit an NQS batch request.
```

In addition, 3 information commands have been provided in the CERN version of NQS.

```
qorder:       display queue order
qusage:       display NQS job execution time
qresource:    display queue resource limits
```

To get more information about a particular command, type:

```
man  command
```

[1] The NQS domain is a list of machines in the NQS configuration. It is currently limited to 256 systems.

7.2.1 NQS Information Commands

dimension

`qorder:` The command `qorder` displays the latest ordering of the queue. This is useful where NQS queue administrators change the execution order of jobs in relation to budget levels, number of jobs running for this user etc.

`qusage:` `qusage` displays the cumulative execution time in seconds for all running NQS jobs.

`qresource:`
`qresource` is a user command to display the available resources for the different batch queues.

7.3 Some Notes on using NQS on SHIFT

- `nqs` will work for you across machines, but only if you have the same user `and` group ids on all machines.
- With care, all files should have the same relative pathname on all machines (`e.g.` /cern/pro/lib) so job scripts with suitable vendor-specific switches for compiling and linking should be able to run on `any` hardware platform within `shift`.
- If a batch machine crashes or undergoes a scheduled shutdown, then at reboot, all jobs that had been running are restarted from scratch. For this reason, a user's job script should take care that any files created by the job do not already exist from an earlier attempt to run the job. (Either delete them if they exist, or have the job work in a unique directory.)

Chapter 8

Using SHIFT Software

This chapter is intended as a tutorial introduction to the SHIFT software. A separate `SHIFT Reference manual` is also available which describe SHIFT commands and the C and FORTRAN interfaces to SHIFT in more detail.

8.1 Disk Pools and Files

A `disk pool` is a set of Unix file systems on one or more `shift` disk servers. Each of these file systems is mounted by `nfs` on every `shift` server, using a convention for the names of the mount points which ensures that files may be referred to by the same Unix pathname on all cpu, disk and tape servers in the `shift` configuration. This name is referred to as the `nfs_pathname`.

The `sfget` command is used to allocate a file. This is a call to the `Disk Pool Manager` (`dpm`), which selects a suitable file system within the pool specified by the user, creates any necessary directory structure within the file system, creates an empty file, and echoes its full `nfs_pathname`.

The `nfs_pathname` has the following form:

`/shift/node/file_system_name/group/user/user_supplied_partial_pathname`

e.g. `/shift/shift1/data1/c3/les/tape23`

The `node` is the name of the `shift` server on which the file system is physically located. The `file_system_name` identifies the file system on the node (it is by

convention the name of the "special" file which identifies the file system in the /dev directory). The `group` and `user` define the owner of the file (see the discussion on file permissions in Section 8.3). The `user_supplied_partial_pathname` is the name of the file as supplied by the user in the sfget command. It may be a single term (e.g. tape23), or it may include directories (e.g. period21/run7/file26). In the latter case the system will create the directories if necessary. Thus, the `nfs_pathname` returned by:

```
sfget -p opaldst period21/run7/file26
```

might have the form:

```
/shift/shd01/data3/c3/les/period21/run7/file26
```

This name may be used by the user in subsequent commands, such as `tpread` or a call to a FORTRAN program. The user does not need to remember the full pathname between sessions, as a subsequent sfget call using the same `user_supplied_partial_pathname` will locate the file and return the full pathname again. The sfget command also sets the Unix command status to indicate if the file was created (the status is set to a value of 1) or simply located (value 0). This may be useful in shell scripts (the status of the last Unix command may be tested through the $? [Bourne shell] or $status [C shell] variables).

If the user only wishes to locate the file if it is present, but not to create it, he may use the `-k` option of sfget.

```
sfget -p opaldst -k period21/run8/file41
```

This example will return a status of 0, and echo the `nfs_pathname` if the file exists in the specified pool, but merely return a status of 1 if it does not exist.

A user may list all of his files in a particular pool by means of the `sfsh` command.

e.g. `sfsh -p opaldst ls period21/run7`

sfsh effectively performs a `cd` (change working directory) to the user's directory in each of the file systems in the pool in turn, issuing the `ls` command in each of these directories. The sfsh command is actually much more general, and will issue any command specified by the user in each of the file systems in the pool. Be careful, however, as the shell expands some characters in the command line using the working directory at the time the sfsh command is issued. Thus, if it is wished

to issue the command `ls *` it is necessary to protect the * from expansion by the shell by enclosing it in single quotes as follows:

```
sfsh -p opaldst 'ls *'
```

Files are removed from `shift` pools by means of the `sfrm` command.

e.g. `sfrm -p opaldst period21/run7/file26`

In addition to the `-p` option, specifying the disk pool to be used, all of the `dpm` commands support the `-u` option, which allows the caller to specify the user (and implied group) associated with the file. The default is the user who issues the command.

Here is some more information about disk pools.

- A file system belongs to at most one disk pool.
- A pool can be `temporary` or `permanent`. If a pool is temporary, a garbage collector is run periodically to maintain the amount of free space on each of the component file systems above a specified `free space threshold`. This involves deleting files on the basis of size and time of last access. Files are deleted from permanent pools only on explicit request by a user.
- A pool can be `public` or `private`. If it is private only a defined set of users will be able to allocate space within the pool.
- When a file is created by sfget the user specifies the pool which should be used. If no pool is specified a default pool named `public` will be used. The disk pool management system selects a suitable file system within the pool using criteria like occupancy level, performance characteristics, etc.
- The user may supply an estimate of the size of the file which will be created (`-s` option of the sfget command), to ensure that a file system with sufficient free space is selected. The default is 200 MBytes, and so there must be at least this amount of free space in one of the file systems in the pool. If `-s 0` is specified this check will be omitted.

8.2 Using shift Files with FORTRAN Programs

Once a file has been created or located by the the Disk Pool Manager, and the user knows the full pathname (the `nfs_pathname`), the file may be used exactly like any other file. Normally, however, the `shift` Remote File I/O System (`rfio`) will be used to access these files, in order to obtain the best possible performance. The `rfio` routines are used by the ZEBRA and FATMEN packages. In order to connect a file to the `rfio` routines, the `assign` command must be used. Assign sets up a connection for both the `rfio` routines and for FORTRAN. The above example becomes:

```
assign `sfget sample45` 10
gendata
```

8.3 File Permissions and Authorisation

This section sounds rather complicated. If you do not wish to be concerned with the details, skip to the summary.

There are two distinct access control facilities:

- The right of a user to allocate space in a disk pool. This is specified in a configuration file used by the `dpm`. The `dpm` will create files on behalf of a user only if he is authorised for the particular pool.
- Unix File Permissions. These are checked by the Unix file system software whenever a file is accessed (create, read, write, remove). The `dpm` will set permissions in directories and files which it creates, as described later in the present section. The user may change these permissions using normal Unix functions. Access to a file depends on the permissions set at the time of access, according to the usual Unix rules.

When the `dpm` creates directories and files, it performs checks and sets ownership and permissions as follows. The `client` is the user who issues the command.

group directory

Checks: The client must belong to the group defined by the name of the directory.
Owner: gid of the group name, uid of root.
Permissions: rwxr_xr_x

user directory
Checks: gid of the client must match that of the parent (group) directory, and the name of the directory must correspond to a username in that group.
Owner: gid of the parent directory, uid of the user defined by the name of the directory.
Permissions: rwxr_xr_x.

lower level directories
Checks: gid of the client must match that of the parent directory.
Owner: gid and uid of the parent directory.
Permissions: rwxr_xr_x.

the file
Checks: gid of the client must match that of the parent directory.
Owner: gid of the parent directory, uid of the client.
Permissions: rwxrwxrwx minus the current user mask of the client.

`Summary:`

All files are world `visible`: anyone may read any of the directories.

To create a file, a client must belong to the `group` in whose sub-tree the file will be created. He may, however, create a file in a directory which belongs to another `user` in that group. This allows several users in the same group, working on a common topic, to maintain their files in a single directory.

The client's `umask` defines access to the file itself, and so read and write access to the file may be restricted.

The user whose name is associated with the user-level directory owns all inferior directories, and he alone is able to manipulate the directory structure manually. All other people may manipulate it only through `dpm` commands.

Before removing a file, the `dpm` checks that the client has write permission to the file `or` is the owner of the parent directory `or` is the garbage collector. When a file or directory is removed, its parent directory is also removed if it is now empty and

is below the user directory level. No permission checks are made when directories are removed, as this is seen as a system function, rather than being performed on behalf of the client. The directories will be recreated automatically when required.

Example

Let us assume that the client is user les in group c3. There is another user in group c3 called pubc3 (probably this is a "dummy" user which has been created specifically to own files which are common to the group). The user mask is set to 22, which means that files will be created with permissions: rwxr_xr_x.

```
shift1 [1] umask
22
shift1 [2] sfget -u pubc3 performance/disk/rs6000/test22
/shift/shift1/data0/c3/pubc3/performance/disk/rs6000/test22
shift1 [3] sfget -u pubc3 performance/disk/sgi340/test7
/shift/shd01/data4/c3/pubc3/performance/disk/sgi340/test7
shift1 [4] sfsh -u pubc3 'ls -ls performance/disk/*'

/shift/shift1/data0/c3/pubc3
ls -ls performance/disk/*

performance/disk/rs6000:
total 0
   0 -rwxr-xr-x   1 les      c3             0 Jan 25 08:58 test22

/shift/shd01/data4/c3/pubc3
ls -ls performance/disk/*

performance/disk/sgi340:
total 0
   0 -rwxr-xr-x   1 les      c3             0 Jan 25 08:59 test7
shift1 [5] sfrm -u pubc3 performance/disk/rs6000/test22
shift1 [6] sfrm -u pubc3 performance/disk/sgi340/test7
```

8.4 Tape-Disk Copying

The `shift` system provides two commands to copy data between tape and disk: `tpread` (tape to disk) and `tpwrite` (disk to tape). These commands select a `shift` tape server which has a device of the correct type available, and initiate a process on the server to mount the tape and perform the copy operation. The remote process uses the basic `shift` tape copy commands, `cptpdsk` and `cpdsktp`.Normally the user will not invoke these commands directly himself.

When using the tpread or tpwrite commands, the user must specify the tape vsn or vid and the pathname of the file. The type of tape unit and its location are specified by means of the `-g device_group_name` option. By default this is `CART` meaning an IBM 3480-compatible tape unit in the manually-operated section of the `cern` Computer Centre tape vault. A tape in the `cern` cartridge tape robot would be selected by `-g SMCF`.

The following example copies data from tape into a file in a `shift` disk pool. First we obtain the pathname of the file from sfget. The example assumes that we are using the C shell, and first stores the pathname allocated by sfget in a shell variable called NFSNAME. This variable is then used in the subsequent tpread, which copies the first file from a cartridge tape into the disk file.

```
set NFSNAME=`sfget -p opaldst testtape`
tpread -v xu4125 $NFSNAME
```

This could of course be achieved by the single command:

```
tpread -v xu4125 `sfget -p opaldst testtape`
```

Similarly, data can be copied from disk to tape using the sequence:

```
set NFSNAME=`sfget -p opaldst testtape`
tpwrite -v fz7425 -q 1 $NFSNAME
```

The next example demonstrates how to copy the first four files from a tape into a single disk file (in this case in the PUBLIC pool).

```
tpread -v qw8256 -q 1-4 `sfget testtape`
```

The files may be copied into separate files by simply supplying a list of pathnames.

```
tpread -v qw8256 -q 1-4 `sfget tfile1` `sfget tfile2` `sfget tfile3`
`sfget tfile4`
```

Multiple files may be created on tape with a single tpwrite command in the same way.

```
tpwrite -v mv1423 -q 1-4 `sfget tfile1` `sfget tfile2` `sfget tfile3`
`sfget tfile4`
```

There are many other parameters to the tpread and tpwrite commands, concerned with label formats, record formats, etc.

Although the above examples all use `sfget` to generate a pathname, the tpread and tpwrite commands are entirely independent of the Disk Pool Manager, and may be used to copy data to and from any disk files accessible to the `shift` system.

8.5 Tape Staging

The idea of tape staging, with a history at `cern` dating back to the CDC7600 in the early 70's, is that a file which has a permanent home on tape is copied to disk while it is being used, the disk copy being thrown away when the space is needed for new data. Similarly, a program creates a new file on disk, and copies it to tape when the program has terminated correctly. The disk copy may then be removed immediately, or may be left in the stage area, to be removed automatically by a garbage collector.

`shift` will not support a general tape staging mechanism, but it includes a number of features which can be used to build such facilities, and a simple stage script is provided - See `SHIFT Reference Manual`. This may be used as an example for more complicated stage routines tailored to the user's needs. The source of the script is maintained in the library `/usr/local/bin`. It is basically a one-line interface to `sfget`, `tpread/tpwrite` and `assign`. When used in conjunction with a `shift temporary` disk pool it provides a straight-forward staging facility.

There are two stage commands: `stagein`, which checks to see if the file is already present on disk and copies it in from tape if necessary, and `stageout` which copies a file from disk to tape. Both of these commands are implemented by a single shell script, `stage`.

8.5.1 stagein

- issues an `sfget` to locate or create the file;
- if the file was created, issues a `tpread` to copy the data from tape to disk;
- if requested by the user (-U option) issues an `assign` command to connect the file to a FORTRAN unit number.

8.5.2 stageout

- issues an `sfget` to locate the file to be staged out;
- issues a `tpwrite` to copy the file to tape;

The parameters of both stagein and stageout are those of the sfget, tpread and tpwrite commands, with the restriction that multiple disk files are not supported.

`Examples`

1. `stagein -v ux3456 -g SMCF -q 4 -U 11 testfile.4`

   will stage in file 4 from a tape in the `cern` cartridge robot, to the PUBLIC disk pool and assign the disk file to FORTRAN unit 11.

2. `stageout -v uz1654 -g SMCF -q 6 -p opaldst run26`

   locates the file `run26` in the pool `opaldst` and copies it to tape.

3. `stagein -v rm3458 -g CART -q 3- batch6.ntuples`

   will stage tape data from file 3 to the end of the tape into a single disk file.

4. Notice that `stageout` copies the file immediately. If a job creates a file and wishes to copy it to tape at the end, it is probably more appropriate to use the `sfget`, `assign` and `tpwrite` commands directly.

   ```
   ....
   ....
   OUTFILE=`sfget tempfile`
   assign $OUTFILE 10
   ( get a file in the public pool, and assign it to FORTRAN unit
   10)
   ```

```
....
....
simul ( run the program, which writes to unit 10)
....
....
tpwrite -v wg7634 -g CART $OUTFILE
sfrm $OUTFILE
(write out the file, and remove the disk copy)
```