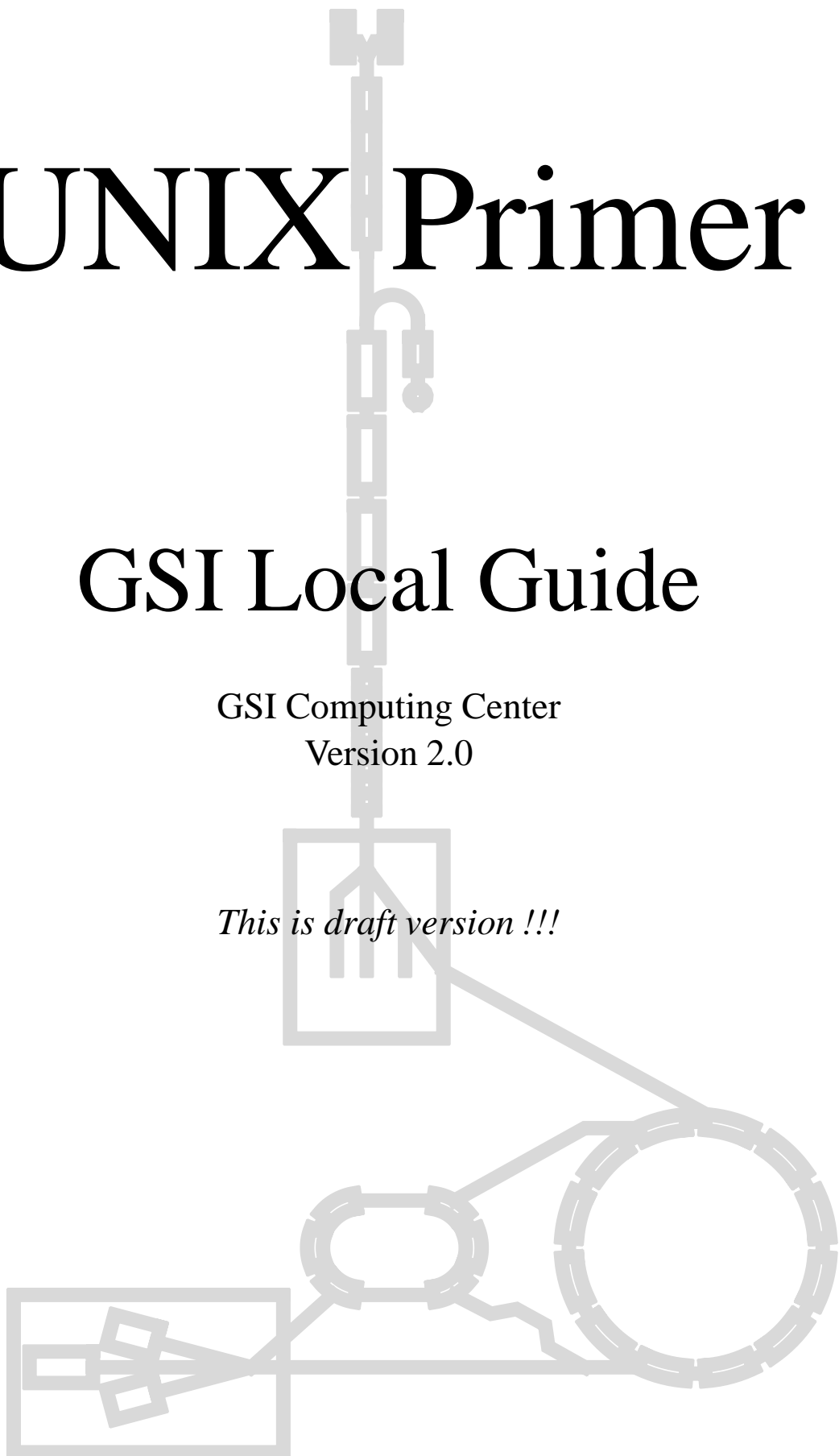


UNIX Primer

GSI Local Guide

GSI Computing Center
Version 2.0

This is draft version !!!



Preface:

More than one year ago, we published our first version of the Unix primer, which has been used in the meantime by many people at GSI and even in the outside HEP community. Nowadays, as more and more physicists have access to a Unix computer either via a X-terminal or use their own workstation, and as the installed computing power has increased by a large factor, we have revised the first version of our Unix primer.

We tried to reflect the changes in the installed hardware, like the installation of the 11 machine AIX cluster, and the installation of new software products, as the batch system for job submission, new backup and restore products and the graphics system IDL. Almost all chapters have been revised, and some have undergone substantial changes like the introduction, the section about experimental data and tape handling and the chapter about the editors, where more editors are described in detail.

Although many topics are still missing or could be improved, we decided to publish the second edition of the Unix primer now in order to give a guide to the rapidly increasing Unix user community at GSI.

As for the first edition, many people again have contributed to this document: Wolfgang Ahner, Eliete Bertulani, Michael Dahlinger, Matthias Feyerabend, Ingo Giese, Horst Göringer, Eva Hocks, Peter Malzacher, Udo Meyer, Kerstin Schiebel, Kay Winkler and Heiko Weber.

Preface for Version 1.0:

In early summer 1991 the GSI Computing Center started a Unix Pilot Project investigating the hardware and software possibilities of centrally operated unix workstation systems. A few machines from DEC, HP and IBM have been installed in the central computer room and have since been used as development platforms by the computer centers application software-, telecommunication-, and systems groups. We feel that now the moment has come to publish our experience with the pilot project. At the same time we want to make the machines available to a larger number of users at GSI and - hence - have chosen the form of this booklet, the

Unix Primer - GSI Local Guide

as our publication medium. We do not consider the present version as complete but rather as a working document, which will be updated at regular intervals. As you can easily see when reading, the primer is the product of many authors. A common style is only evolving. However, as it gives the answer to many "frequently asked questions" we have chosen to publish version 1.0 in the state it is in now. We hope that nevertheless the **Primer** will be helpful to many users. Any readers are welcome to comment and make suggestions. We will gratefully accept mail, printed copies with corrections or personal visits at the help desk (alias Benutzerberatung).

The following people have made contributions to version 1.0 of the primer: Michael Dahlinger, Hans Döbbling, Horst Göringer, Richard Herrmann, Eva Hocks, Frank Kraske, Peter Malzacher, Udo Meyer, Thomas Schwab, Heiko Weber.

This document has been produced with the \LaTeX macro package. The style used (gsiman.sty) has been derived from CERN's cernman style. We acknowledge the support of M.Goossens and A.Samarin [4] from CERN/CN.

Michael Dahlinger has coordinated this project and has taken care of the final editing.

Printed copies of this document are available from the help desk. Printed on copy via
World Wide Web :

<http://www.gsi.de/computing/unix/primer/primer.html>

A copy of this document can also be obtained via anonymous ftp on internet node **ftp.gsi.de** as
/dist/doc/primer.ps.

Unix Primer, Version 2 formatted at GSI: 17th July 1995

If You Need Help

User Registration

New users-IDs may be obtained by NN (Room 2.243, Tel. 2552). Registration forms are also available at the help desk.

Operating - Hardware Problems

For hardware and networking problems, repairs and printer maintenance, please contact the central operating Tel. 2515, machine-room, Südbau, ground-floor, Monday – Friday 6:00 to 23:00.

Help Desk - Software Problems

There is a help desk (**Benutzerberatung**) for all general computing related questions. Telephone 2555, Office 2.244, Südbau, first floor. The opening hours are 9:30 to 11:30 and 14:00 to 16:30 except Tuesdays (a.m. closed because of the DV & EE Palaver) and Fridays (until 16:00). If the help desk cannot help you, they will refer you to the appropriate specialist.

UNIX related staff ¹

If you need help, please contact the help desk first.

The DV & EE department at GSI supports the GSI local networks and various computing environments (MVS on an IBM mainframe, VMS on VAX computers, Open VMS on DEC AXP computers, UNIX on IBM RS/6000 (AIX), HP 9000 (HP-UX) and DEC 9000 (Ultrix) workstations. The following people have at least part of their time UNIX related responsibilities.

Name	Phone	
Wolfgang Ahner	2548	Integrated Network and System Management
Gerhard Aker	2497	Local Networks
Michael Dahlinger	2546	Data Analysis, CERN Software
Hans Essel	2491	OS/9, Lynx, CVC
Matthias Feyerabend	2519	Backup, Archive
Ingo Giese	2547	Graphics, CATIA
Horst Göringer	2553	Experiment Data Handling, Tapes, Data Analysis
Eva Hocks	2701	System Management (AIX, HP-UX, Ultrix), DCE
Peter Malzacher	2551	Application Software, Mathematical Libraries
Udo Meyer	2525	Operating, External Networks
Nikolaus Kurz	2979	Lynx, CVC
Kerstin Schiebel	2703	Textprocessing
Wolfgang Schiebel	2498	Local Networks
H.G. Schmidt	2498	Local Networks
Heiko Weber	2556	System and Data Management AIX, DCE
Kay Winkler	2551	Application Software, WWW

¹ None of us is as smart as all of us.

Manual Conventions:

Throughout this manual the following typographical conventions are used:

If You See This ...	It Means ...
<p>\$ <u>what you type</u></p> <p>\$ <u>command parameter</u></p>	<p><u>underlined</u> text is used in dialog examples to distinguish what you type from what the computer displays</p> <p><i>italic</i> text is used to denote parameters which must be specified on commands</p>
<p>Esc q</p> <p>Ctrl - c</p> <p>Alt - d</p> <p>Tab</p> <p>Backspace</p> <p>Del</p>	<p>you press the escape key on the keyboard followed by key q. The escape key usually sits on the top left of a PC-like keyboard or on the top center coinciding with F11 on a VT200-like keyboard</p> <p>you press and hold-down the control key and simultaneously press the c key.</p> <p>you press and hold-down the alt key and simultaneously press the d key.</p> <p>The alt key - if it exists - is next to the space bar on both VT200-like and PC-like keyboards</p> <p>you press the Tab key</p> <p>you press the Backspace key</p> <p>you press the Delete key</p>
<p>Mb1</p> <p>Mb2</p> <p>Mb3</p>	<p>the left mouse button</p> <p>the center mouse button</p> <p>the right mouse button</p>

Table of Contents

1	How to get started	1-1
1.1	How to obtain an Account	1-1
1.2	Login	1-1
1.2.1	Connection via XDMCP	1-1
1.2.2	Connection via Telnet	1-3
1.2.3	The Login Procedure	1-4
1.2.4	Locking your Terminal Screen	1-4
1.2.5	Logout	1-5
1.3	Accessing Unix via X-Windows	1-5
1.3.1	X-server	1-5
1.3.2	Window Manager	1-5
1.3.3	Creating Terminal Windows	1-5
1.4	Unix Shells	1-6
1.5	Desktop Environments	1-6
1.5.1	HP Visual User Environment	1-6
1.6	Some initial Hints	1-7
1.6.1	The first Commands	1-7
1.6.2	Handling Motif Windows	1-7
1.7	GSI Customization	1-8
1.7.1	Environment Variables	1-8
1.7.2	The Profile Files	1-10
1.7.3	Command Line Editing	1-10
2	UNIX Commands	2-1
2.1	Constructing a command line	2-1
2.1.1	Redirection of Input and Output	2-1
2.1.2	Pipelines	2-1
2.2	Regular Expressions	2-1
2.3	Quick Reference of Commands	2-2
2.3.1	Managing Directories	2-2
2.3.2	Managing Files	2-2
2.3.3	Managing Jobs	2-2
2.3.4	On-line Help	2-2
2.3.5	System Information	2-2
2.3.6	Utility Programs	2-2
2.3.7	Directory Identifiers	2-3
2.3.8	Special Characters	2-3
2.4	Shell Scripts	2-3

3	Files and Directories	3-1
3.1	The UNIX File System	3-1
3.1.1	Naming Directories and Files	3-1
3.1.2	Rules for Naming and Accessing Files	3-1
3.2	Working with Directories	3-1
3.2.1	Displaying the contents of a directory: <code>ls</code>	3-1
3.2.2	Changing the Working Directory: <code>cd</code>	3-1
3.2.3	Determining Your Working Directory: <code>pwd</code>	3-2
3.2.4	Creating a New Directory: <code>mkdir</code>	3-2
3.2.5	Removing an Existing Directory: <code>rmdir</code>	3-2
3.2.6	Renaming a Directory: <code>mv</code>	3-2
3.3	Working With Files	3-3
3.3.1	Displaying the Contents of a File: <code>cat</code>	3-3
3.3.2	Renaming a File: <code>mv</code>	3-3
3.3.3	Copying a File: <code>cp</code>	3-3
3.3.4	Deleting a File: <code>rm</code>	3-3
3.4	File and Directory Permissions	3-3
3.4.1	Determining Permission: <code>ls -l</code>	3-3
3.4.2	Changing Permission: <code>chmod</code>	3-4
3.5	Filesystems and Links	3-4
3.5.1	Filesystems	3-4
3.5.2	Links	3-5
3.6	<code>/u</code> , <code>/d</code> , <code>/s</code> and <code>/tmp</code>	3-5
4	Basic Services	4-1
4.1	Help and Documentation	4-1
4.1.1	How to get Help	4-1
4.1.2	The <code>man</code> Command	4-1
4.1.3	The <code>info</code> Command	4-1
4.1.4	The VUE Help button	4-1
4.2	World Wide Web at GSI -	4-1
4.2.1	Navigation though the World Wide Web	4-2
4.2.2	Information about GSI	4-3
4.3	Mail Facilities	4-4
4.3.1	Simplified E-Mail Addresses	4-4
4.3.2	The Standard Mailer	4-4
4.3.3	The <code>e1m</code> Mailer	4-5
4.3.4	The <code>xmh</code> Mail Interface	4-5
4.4	Print Services	4-6
4.5	Local Networking Tools	4-7
4.5.1	Opening Remote X-Windows	4-7
4.5.2	Remote Processing	4-8
4.5.3	File transfer	4-9
4.6	IBM Mainframe Access	4-11

4.6.1	X-Window Users	4-11
4.6.2	Other X Windows environments	4-11
4.6.3	Alphanumeric sessions	4-11
4.7	Backup and Archive Services	4-12
4.7.1	Automatic Backup with DSM	4-12
4.7.2	Starting and Ending DSM	4-12
4.7.3	File specifications	4-12
4.7.4	Backing Up Files	4-13
4.7.5	Archiving Files	4-13
4.7.6	Restoring Backup Versions	4-13
4.7.7	Retrieving and Deleting Archived Files	4-13
4.7.8	Using the Utilities Menu Options	4-14
4.7.9	Using the View Menu Options	4-14
4.7.10	Displaying Online Help	4-14
4.8	Batch Jobs on the RS/6000 Cluster	4-14
4.8.1	LoadLeveler using NQS Scripts	4-15
4.9	Tape Handling Tools	4-15
4.9.1	Tape Allocation System	4-15
4.9.2	Tape Copy Station	4-16
5	Editors	5-1
5.1	vi Editor	5-1
5.1.1	Operating Modes	5-1
5.1.2	Starting vi	5-1
5.1.3	Exiting vi	5-1
5.1.4	vi Command Mode	5-1
5.1.5	ex Command Mode	5-2
5.1.6	Basic vi Keystrokes	5-2
5.1.7	The .exrc File	5-3
5.1.8	The EXINIT Variable	5-3
5.1.9	More about vi	5-3
5.2	GNU Emacs	5-3
5.2.1	Emacs Commands	5-3
5.2.2	Starting Emacs	5-4
5.2.3	Exiting Emacs	5-4
5.2.4	Emacs Screen	5-4
5.2.5	Emacs modes	5-4
5.2.6	Basic Emacs Keystrokes	5-5
5.2.7	More information about Emacs	5-6
5.3	edt+	5-6
5.4	sofedit Editor	5-7
5.5	xedit	5-10
5.6	Other Editors	5-11
5.6.1	ed Editor	5-11
5.6.2	LPEX	5-11
5.6.3	INed Editor	5-12
5.6.4	uni-XEDIT editor	5-12
5.6.5	Vuepad	5-12
5.6.6	Notepad	5-12

6	Text processing	6-1
6.1	T _E X and L ^A T _E X text processing	6-1
6.1.1	Advantages of L ^A T _E X	6-1
6.1.2	Disadvantages of L ^A T _E X	6-1
6.1.3	How does L ^A T _E X work?	6-1
6.1.4	T _E X glossary	6-2
6.1.5	Documentation in PostScript form	6-4
6.1.6	Suggested Reading	6-4
6.1.7	Running L ^A T _E X	6-4
6.1.8	Using PostScript fonts	6-6
6.1.9	Merging Graphics and Text	6-7
6.1.10	Other Stylefiles	6-10
6.1.11	L ^A T _E X2 _ε —the new L ^A T _E X release	6-12
7	Program Development	7-1
7.1	Overview	7-1
7.2	Compiling and linking a program	7-1
7.3	Correcting errors in a program	7-2
7.4	Building and Maintaining a Program	7-2
7.4.1	make	7-2
8	Applications and Utilities	8-1
8.1	Mathematical packages	8-1
8.1.1	Mathematica	8-1
8.1.2	AXIOM	8-1
8.2	Mathematical Libraries	8-1
8.2.1	NAG FORTRAN Library	8-1
8.2.2	The ESSL Library	8-4
8.2.3	LAPACK	8-4
8.3	Graphical Tools	8-4
8.3.1	Interactive Graphics with IDL	8-4
8.3.2	Handling of Image Files	8-5
8.3.3	Image Format Conversion	8-7
8.3.4	PHIGS	8-7
8.3.5	xv: Interactive Image Display for the X Window System	8-9
8.3.6	xpick: Pick images from an X11-screen	8-9
8.4	Screendump on XScreens	8-10
8.4.1	The interactive way: xv	8-10
8.4.2	The commandline way: xwpick	8-11
8.4.3	The classic way: xwd	8-11

9	Experiment Data Processing	9-1
9.1	Experiment Data Handling	9-1
9.1.1	Experiment Data in Unix	9-1
9.1.2	The Central Experiment Data Server	9-2
9.1.3	Experiment Data Naming Conventions in MVS	9-2
9.1.4	A Client-Server Package for Access to the Experiment Data Server	9-2
9.1.5	Moving Experiment Data with FTP	9-3
9.1.6	Mounting the Server with NFS	9-4
9.2	CERN Software	9-5
9.2.1	paw	9-5
9.2.2	PIAF, Parallel Interactive Analysis Facility	9-6
9.2.3	cmz	9-6
9.2.4	GEANT	9-6
9.2.5	Organization of the CERN program library	9-7
9.2.6	Usage of CERN-library programs	9-7
9.3	Unix Clients for the Event Servers at GSI	9-11
9.3.1	Making Event Server Clients	9-12
9.3.2	Starting the Clients	9-12
9.3.3	Handling Shared Memory	9-13
9.4	SATANGD on Unix	9-14
A	Telnet support for MVS (3270) terminals	A-1
A.1	Appendix	A-1
B	Logging in from MVS via Telnet	B-1
C	How to use the Tape Stacker	C-1
D	GNU Software	D-1
E	Motif Windows	E-1
F	Unix Hardware at GSI Computing Center	F-1
F.1	Managing your X-Terminal by XDM	F-2
G	Introduction to Internet Services	G-1
G.1	About Internet	G-1
G.1.1	FTP	G-1
G.1.2	Internet addresses	G-1
G.2	Internet Services	G-1
G.2.1	Overview	G-1
G.2.2	archie	G-1
G.2.3	NetNews	G-2
Index		G-4

List of Figures

1.1	Mwm Root Menu1, activated with the left mouse button	1-2
1.2	Mwm Root Menu1, together with the X11-Applications submenu	1-2
1.3	Mwm Root Menu2, activated with the middle mouse button	1-2
1.4	Mwm Root Menu1, activated with the right mouse button	1-2
3.1	Part of a typical UNIX file system	3-2
4.1	LoadLeveler GUI xloadl	4-15
5.1	sofedit window	5-8
5.2	sofedit search dialog window	5-10
6.1	Data flow for the files used by L ^A T _E X	6-3
6.2	Large Text with L ^A T _E X and PostScript	6-7
6.3	Encapsulated PostScript example 1	6-8
6.4	Encapsulated PostScript example 2	6-8
6.5	A single centered figure	6-10
6.6	The same figure as 6.5, but in draft mode	6-10
6.7	Distorting a picture with epsfig	6-11
6.8	Example 1 - gsifoil.sty	6-12
6.9	Example 2 - gsibrief.sty	6-12
6.10	Example 3 - fancyheadings.sty	6-13
6.11	Example 4 - cernmins.sty	6-13
8.1	Mathematica display	8-1
8.2	AXIOM hyperdoc entry panel	8-2
8.3	IDL display	8-5
9.1	PAW and its components	9-6
9.2	File system structure of the CERN program library	9-8
9.3	Help Menues in PAW for Clients of Event Server	9-13
9.4	List of Shared Memory Segments on an AIX node	9-14
F.1	Unix Hardware at GSI Computing Center	F-1

List of Tables

1.1	List of centrally available Unix hosts	1-2
1.2	Command Line Editing Functions	1-11
4.1	List of centrally available tape drives	4-16
5.1	Moving the cursor with sofedit	5-8
5.2	Selecting items with sofedit	5-9
5.3	Cut, copy and paste text with sofedit	5-9

5.4	Delete and Kill text with <code>sofedit</code>	5-10
5.5	Aligning text with <code>sofedit</code>	5-10
6.1	List of predefined colors for use with <code>dpcolor</code>	6-11
9.1	Naming Conventions for Experiment data in MVS	9-3
9.2	CERN libraries installed on <code>/cern/version/lib</code> subdirectories	9-8
A.1	List of available Printers	A-2
E.1	Summary of Operations with Motif Windows and Icons	E-2

Chapter 1: How to get started

Currently the GSI DV&EE department supports the Unix platforms

- IBM RS/6000 running with the operating system AIX,
- HP 9000-700 running with HP-UX, and
- DEC 5000 running with ULTRIX (with limited support only).

Users may get in contact with Unix on three different levels:

1. The simplest access is when starting a Unix session with the `telnet` command from a **text oriented** terminal, such as a DEC VT100 or VT200, for example.
2. With the appropriate hardware platform - an X-terminal, PC, or workstation - users can work in a comfortable **X-Windows** environment. The X-Windows system is a network-based graphics windowing system, which can be seen as a layer above the operating system.
3. You can work in an even more comfortable (X-Windows) environment, if you use a **session manager**, which controls your complete session with all applications running in it. So it is very easy, for example, to restore the previous session or to organize your working environment in several workspaces. A comfortable session manager is available with the Visual User Environment (VUE) of HP.

1.1 How to obtain an Account

If you want to work with Unix, please contact

Editha Mitz, room 2.243, Tel. 552,
email address: E.Mitz@gsi.de

Currently the GSI DV&EE department runs a big RS/6000 cluster, a HP 9000-700 cluster, and a DEC 5000 workstation. The actual Unix host names are listed in table 1.1 on page 1-2. For more details on the currently available hardware see F.1 on page F-1.

An account 'for Unix' is always granted for all machines of both, the IBM and HP clusters. However, we urgently recommend to **login only on clri6a, rzri6f, rzhp9a, or rzhp9b** (our login servers). The other cluster machines are primarily intended for batch usage controlled by the **Load Leveler** (see section 4.8 on 4-14).

As Ultrix is only supported with some restrictions, login access to the node **rzds5a** is only granted on special request.

1.2 Login

The steps to be performed in the login procedure depend on your working place. Vendor dependencies are negligible. However, the procedure may be different when logging in on 'dumb' text oriented terminals, or on devices supporting X-windows such as workstations, PCs, or X-terminals, especially when working in environments under control of a session manager.

On text oriented terminals, e.g. a VT100, you connect to a(nother) host with the `telnet` command. On an X-terminal, the first connection after booting should be established with XDMCP, but is also possible via telnet window (see below).

1.2.1 Connection via XDMCP

From X-terminals, access to Unix is recommended via XDMCP (X Display Manager Command Protocol). It provides a mechanism by which X-terminals can request login services from a network host.

After successful booting an X-terminal, a menu for selection of the login hosts appears. On Tektronix X-terminals (XPxxx) it has the title **TekHostMenu**. On top it contains a row with buttons to select the access method. Currently supported are

1. XDMCP (default) (see section ?? on page ??).
2. TELNET (see section 1.2.2 on page 1-3).

Normally the **XDMCP** button (left) is activated. If not, move the mouse cursor there and press the left mouse button to activate XDMCP. Now select a node for login.

A login menu with fields for your account name and your password appears. You should keep in mind that Unix is **case sensitive** - so be careful when entering account name and password!

Connection to AIX node

After successful authorization (with account name and password) on one of the AIX login servers (see table 1.1 on page 1-2), the Motif Window Manager `mwm` from that host is started on your X-terminal. The Window Manager allows you to work with multiple windows on

Nodes	hardware	comment
rzri6f	IBM RS/6000-970	login server
clri6a,	IBM RS/6000-580	login server
clri6b, clri6c, ..., clri6g	IBM RS/6000-580	batch server
clri6h, clri6j, clri6k	IBM RS/6000-590	batch server
rzhp9a, rzhp9b	HP 9000-720	login server
rzds5a	DEC 5000-200	only limited support

Table 1.1: List of centrally available Unix hosts



Figure 1.1: Mwm Root Menu1, activated with the left mouse button

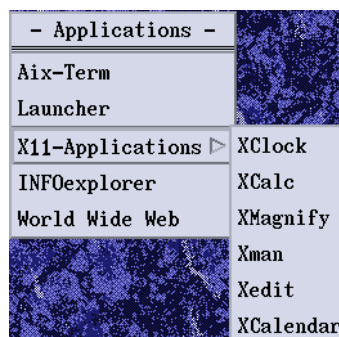


Figure 1.2: Mwm Root Menu1, together with the X11-Applications submenu

your physical screen. The functionality of the mwm is described in section 1.6.2 on page 1-7.

Caution: Do not include interactive responses in your profile file, if you login via XDMCP! In this connection methode, your profile files are executed in batch and cannot request input from your terminal screen! Especially the display address is known by XDMCP and need neither be set nor exported.

Your screen will show by default the X11 application xclock and an aixterm window. When you click the terminal background with one of the three mouse buttons, a related application menu will appear. Each of the three buttons provide different menus to invoke some useful applications or mwm functions:

1. With the left mouse button you get an application menu (see figure 1.1 on page 1-2) containing buttons for
 - a (new) AIX **terminal window** on the current host (see section 1.3.3 on page 1-5),
 - the **Launcher** menu providing some basic functions as *Lock Screen* (see section 1.2.4 on page 1-4) or the local *mwm* (see section 1.2.2 on page 1-3),



Figure 1.3: Mwm Root Menu2, activated with the middle mouse button

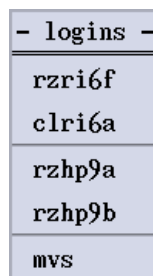


Figure 1.4: Mwm Root Menu1, activated with the right mouse button

- an **X11 Applications** submenu (see figure 1.2 on page 1-2) containing buttons for some X-Windows based standard applications such as a clock (*XClock*), a calendar *XCalendar*, a calculator (*XCalc*), etc,
 - online access to the complete AIX manual information (**INFOexplorer**, see section 4.1.1 on page 4-1), and
 - access to the GSI home page of the **World Wide Web** (see section ?? on page ??).
2. The middle mouse button provides a menu with some Motif Window Manager functions such as reordering the window stack or leaving the *mwm* (see figure 1.3 on page 1-2).
 3. With the right mouse button you get a Login menu containing a selection list with the current login servers at GSI (see figure 1.4 on page 1-2).

You can customize these menus for your special purposes with a suitable `.mwmrc` file in your home directory. A useful sample file - the GSI default described above - can be copied from `/usr/local/doc/.mwmrc`.

Connection to HP-UX node

When you selected one of the HP-UX workstations, after successful authorization, a VUE session (Visual User Environment) starts. A session manager becomes active which has full control of your environment and provides - among others - also a Motif Window Manager. When you log in with HP VUE for the first time, many menus and icons appear for immediate graphics access with mouse and click. For more information, see the section 1.5.1 on page 1-6.

At a following login you get again the default startup environment, or you can continue with the previous environment which was active at the last logout - depending on your actions when logging out.

1.2.2 Connection via Telnet

Login from one node to another node via `telnet` works on each hardware platform, on text oriented terminals and on X-terminals, on PCs, and on workstations, provided the required (TCP/IP) software is properly installed.

If you want to access an HP-UX workstation, e.g. `rzhp9a`, enter the command

```
telnet rzhp9a
```

Some messages show up with the login prompt:

```
Trying...148.181.64.51
Connected to RZHP9A
Escape character is '^['.
```

```
HP-UX rzhp9a A.09.01 E 9000/735 (ttys7)
```

```
login:
```

Telnet Windows

On X-terminals, telnet connections may be established not only with the `telnet` command, but also with **telnet windows** running in principle a VT100 terminal emulation program.

There are two ways to get a telnet window on the Tektronix X-terminals :

1) activate the **TELNET** button in the **TekHostMenu** (see above) with your mouse and select a host from the selection list.

A telnet window appears. It is labeled on top with **telnet** and prompts you, for example, when you selected the node `clri6a`, with

```
clri6a
```

```
IBM AIX Version 3 for RISC System/6000
(C) Copyrights by IBM and by others
1982, 1991.
login:
```

When you connect via telnet immediately after your X-terminal was booted, no window manager is active so far. Which is the telnet window and the **TekHostMenu** have no frames and cannot be moved or iconized (see section 1.3.2 on page 1-5). You have to activate manually the local Motif Window Manager (*mwm*) available with your X-terminal.

When logged in at any AIX or HP-UX host, invoke the command

```
launcher &
```

A menu with title 'Launcher' appears (see section 1.2.1 on page 1-1). Move the mouse cursor to the field **MWM** and press the left mouse button to activate. After a few seconds, all X11 applications on the screen will get a Motif frame (see section 1.6.2 on page 1-7).

2) With the launcher menu, you may also get a telnet window. Activating the field **Host Connections →**, a submenu appears containing a field labeled **Telnet**. Keep the left mouse button pressed and move it to this field - the result will be a new telnet window. It prompts you with

```
telnet>
```

Now you can login to a host using the **open** statement:

```
telnet> open hostname
```

1.2.3 The Login Procedure

When you got connected to a Unix host by any of the different ways described above, enter your account name and password. Please keep in mind that Unix is **case sensitive!** Upper and lower case letters have a different meaning! Besides that, in some environments it may occur that the normal backspace keys might not yet be available during the login process. Therefore, in case of problems after mistyping, reenter name or password completely.

After successful authorization the system greets you with a welcome message and provides some information, such as the existence of unread news or of new mail files. In new accounts, in foreground you may be asked for two things (see also section 1.2.1 on page 1-1):

1. the **terminal type** (see corresponding section on page 1-4), and
2. the **display address** (see corresponding section on page 1-4).

As dumb text terminals have no display address, just press in this case.

The system responds with the default prompt

```
hostname: /u/username sequencenumber$
```

which may be, for example:

```
clri6a:/u/goeri 33$
```

and is ready for command input. Now you are in an environment called **shell**, and the commands you can enter are called **shell commands**. For more information see section 1.4 on page 1-6.

You can customize your account(s) by proper settings in the login profile files thus avoiding the prompts during the login process (see section 1.7 on page 1-8).

Terminal Type

With the terminal type you specify the terminal hardware or the emulation program you use when communicating with Unix. The most important terminal types used at GSI are

vt100	DEC
vt220	DEC
3270	IBM ¹
xterm	standard X-window terminal
aixterm	AIX terminal emulation
hpterm	HP-UX terminal emulation
dxterm	ULTRIX terminal emulation

When you work in a Unix environment without the appropriate terminal type set, you should keep in mind that not all keys of your keyboard may be available in the way you expect it! You should correct your terminal type with the `export` command (see section 1.7.1 on page 1-8). For example, if you want to correct your terminal type to vt100, enter

```
export TERM=vt100
```

On dumb terminals, there may be applications that can not work correctly because they may require hardware features not available. However, many Unix commands can be used in such an environment, e.g. to compile and run user programs, or to manage your file system.

Display Address

The display address is the address of the 'host', on which your X-server resides, that means, the address of your X-terminal, PC, or workstation, appended with `:0`. The host address has been assigned by the network group and is known as the **internet address or name**.

Examples of display addresses at GSI are:

xwtae:0	(X-terminal)
pc01:0	(PC)
vsaa:0	(VMS Vaxstation)
dsaa:0	(ULTRIX workstation)
140.181.97.168:0	(decimal representation of display address)

1.2.4 Locking your Terminal Screen

If you have to leave your working place for some time, in most environments you may lock your terminal to inhibit unauthorized access.

On X-terminals, for example, you may activate the field in the *Launcher* window (see section 1.2.1 on page 1-1). If you do it the first time after booting, you are asked for a password. This password is required again to unlock and remains valid for all future

¹ only known in AIX

invocations of the **Lock Screen** function until the next boot of the X-terminal!

DO not lock public X-terminals !

In an HP VUE environment you may use the **Lock Button** in the menu bar to lock your terminal. To unlock, you must enter your user password.

1.2.5 Logout

Closing a Terminal Window

When you are finished with your work session, you can close the windows no longer needed by

- entering the shell command **exit**
- entering **(Ctrl) - (D)**, or
- using the **Close** button of the **window menu**.
The window menu is activated by the button **-** in the upper left corner. Within this menu move the cursor to the field **Close** and press the left mouse button again.

A terminal window disappears when the primary shell is closed.

Closing a Telnet Window

When the login shell in a telnet window is closed, the connection to the host is cut. The window is still available and prompts you with `telnet>` - ready for the next login. It can be closed by

- entering **quit**,
- entering **(Ctrl) - (D)**, or
- utilizing the **Close** button of the **window menu**.

Closing a HP-VUE Session

When you work with HP VUE, it is not necessary to close any window on HP hosts for logout. You just need to press the **logout** field, the rightmost button in the menu bar at the bottom of your screen. With the **Style Manager**, you can customize the logout

- Either the current window configuration will be stored and be restored at the next login, or
- you will start with a default window configuration at the next login, or
- you will be asked, if you want to start next time with the current or the default window configuration.

1.3 Accessing Unix via X-Windows

You can skip this section, if you work with Unix via dumb terminals such as DEC VTxxx or IBM 3270. This section is only relevant for hardware platforms enabling the usage of X-windows, such as X-terminals, PCs, or workstations.

X-Windows is a standardized, vendor independent user interface, originally developed at MIT. It is a network-based graphics windowing system which allows you to work with multiple programs simultaneously, each in a separate window on your screen.

You have access to different hosts from different windows on a single physical screen. This includes also the non-Unix operating systems available at GSI such as MVS on IBM 9121-621 (see 4.6 on page 4-11) and VMS on the VAX or AXP computers.

1.3.1 X-server

Central part of the X-Window system is the X-server, also called X-Window server or display server. It controls screen, keyboard, and mouse, and the process communication requests. The X-server updates the windows on the screen on request of the clients (e.g. programs initiated by your input).

On X-terminals, the X-server is already locally available and need not be started. When you work locally at a workstation or PC, you may have to start the X-server by command. The command name depends on the X-emulation used. On a RS/6000 workstation, for example, this can be done with the command

```
startx
```

1.3.2 Window Manager

Working with multiple windows requires a window manager, which allows, for example, to change the size and position of windows. The window manager provided with **OSF/Motif** is called **mwm**, the Motif window manager. Its functionality is described in section 1.6.2 on page 1-7.

1.3.3 Creating Terminal Windows

Logged in on any centrally available Unix node at GSI (see table 1.1 on page 1-2), you can create windows with terminal emulations (**terminal windows**) on any central Unix node, VMS, and MVS system, using the command

```
xt node
```

The parameter `node` specifies the node providing the terminal window. Use `'mvs'` to select the MVS host IBM 9121-621 (see section 4.6 on page 4-11). If `node` is omitted, a new window on the current host will be created.

On Unix hosts, **`aixterm`**, **`hpterm`**, or **`dxterm`** commands, respectively, are invoked by the `xt` command with GSI-specific defaults. The terminal windows are Motif windows with vertical scroll bars on the right by default. They are labeled on top with

hostname:username

and should be used for working. X-terminal users should **not** work within telnet windows running VT100 emulations, because only a subset of keys and fonts may be available, and several tools (e.g. when requiring the use of function keys) may not work correctly. **The intended use of telnet windows is only login!**

1.4 Unix Shells

After successful completion of the login process you are in an environment called **shell**. It is another process, which has been started (**spawned**) at the end of the login process. A new shell is also started for each invocation of a terminal window.

A shell is the interface between the operating system and the user. It interprets the commands you type, and the keys you press, in order to direct the operating system to take an appropriate action.

On most platforms, there are several shells available differing in power and functionality. The most popular ones are

- the **Bourne Shell** (`sh`),
- the **Bourne Again Shell** (`bash`),
- the **C Shell** (`csh`),
- the **tc Shell** (`tcsh`),
- the **Korn Shell** (`ksh`) and the
- the **z Shell** (`zsh`)

Most Unix systems offer several shells with the Bourne shell as default. However, at the Unix platforms supported by the GSI DV&EE department, the **Korn shell is default**, and all GSI customization is only done for the Korn shell (see section 1.7 on page 1-8). It is a newer shell, developed by David Korn at Bell Laboratories,

and it is upwardly compatible with the most features of the Bourne shell.

You can invoke a new Korn shell (subshell) on top of your current shell by typing

`/bin/ksh`

You can exit this subshell by typing

`exit`

More details on the Korn shell environment and its customization can be found in 1.7 on page 1-8.

1.5 Desktop Environments

A desktop environment provides a collection of windows with menus and icons for immediate graphical access (with mouse and click) to the most common commands and applications.

When you click a menu button or icon, application programs are started providing the required functionality. Depending on their task they may have names such as file manager, style manager, help manager, etc. The default configuration of the desktop environment can be tailored by each user for his specific needs.

Each desktop contains a collection of small pictures, called **icons**. Each icon visually represents an actual file, directory, program, or object and can be activated by double click with the left mouse button. When you activate a program icon, you actually run the program. When you (double) click an icon representing a directory, the contents of this directory is displayed in a new window, either with text lines or with new icons.

Desktops also provide **menu bars** which functions can be activated by single click with the left mouse button. In general these functions are to create new files, directories, or other desktops, and help information. For more information on the functionality and on the configuration possibilities activate the **Help** menu in the specific desktop environment.

1.5.1 HP Visual User Environment

HP VUE runs on Hewlett Packard 9000-700 workstations with HP-UX. It offers components such as

1. **Login manager:** Performs configuration activities, starts the X-server, and performs the login and logout activities.
2. **Session manager:** Controls all windows and applications of complete sessions from login to logout.

3. **Workspace manager:** Controls and manages up to six different working environments ('workspaces'), which can be set up by the user.

The complete configuration of your session can be saved when logging out and restored at a new login (see section 1.2.5 on page 1-5.)

1.6 Some initial Hints

1.6.1 The first Commands

To run a command, type the command's name after the prompt and press the **Enter** key. When the command is finished, the prompt appears again. For example, if you want to know who and where you are:

```
clri6a:/u/goeri $ whoami Enter
goeri
clri6a:/u/goeri $ hostname Enter
clri6a
clri6a:/u/goeri $
```

Another important command is `cd`, which stands for change current working directory:

```
clri6a:/u/goeri $ cd uti Enter
clri6a:/u/goeri/uti $
```

Shell Scripts

The commands described in the previous examples have been executed when typed at the command prompt. They can also be written to files and executed as command lists or **shell scripts**, which corresponds to a TSO CLIST in MVS or a DCL in VMS. For more information, see section 2.4 on page 2-3.

Set Password

Creating your account for the AIX and HP-UX clusters, the system administrator gives you a start-up password. It is a one-time password and you have to change it when you log in for the first time on any AIX login server and HP-UX login server. You can change your password anytime with the command

```
passwd
```

After invocation, you are prompted for the current password to check your authorization. Then you can enter a new password. You have to type it twice, because the system compares both versions to avoid accidental mistyping.

For security reasons, the new password has to fulfill some regulations. It must have at least six characters. If you enter more than eight characters, only the first eight characters are significant. A password should be different from the account name, and a new password must be different from the previous one.

If correctly handled, passwords are never displayed. Please keep in mind that **upper case and lower case letters are treated differently** in Unix environments. If you forgot your password, please contact the Unix system administrators.

1.6.2 Handling Motif Windows

In each window, a process or program is active. In a terminal window, a **terminal emulation** running a shell is active (see 1.3.3 on page 1-5). Other windows can be created by application programs to communicate with the user.

You can have multiple windows on your screen, which can **overlap** much like sheets of paper on a desktop. All windows and menus have a frame in **Motif** style and are managed by the Motif window manager (mwm) provided with OSF/Motif.

Windows are handled by **pointer devices**, that means input devices, that, when moved across a flat surface, move the pointer symbol on the display correspondingly. Pointers usually have buttons that can be pressed to send signals, which in turn accomplish certain functions. The **mouse** is the most common example of a pointer device.

Using the Mouse Cursor

With a mouse, a lot of functions can be performed with windows:

1. You can accomplish **geometrical** operations, such as moving or resizing.
2. The contents of windows can be **scrolled**.
3. Windows can be **iconized**, that means, they can be converted to a small picture called **icon**, which is a representation of a background window, and they can be restored again into the state of an active window.

In section E on page E-1 the operations possible with Motif windows are explained in detail.

The mouse is used to indicate or activate a graphical element on the screen such as a window, icon, or command button. Normally, a mouse has three buttons labeled here as left, middle, and right button. By placing the mouse on a particular element and then performing some button action and possibly cursor motion, you can invoke a variety of commands. The types of actions you can perform are:

- Click** Press the button down and release it immediately again. A double or triple click is two or three clicks in succession, with no pause between clicks.
- Press** Push the button and hold it down.
- Release** After pressing a button down, release it by letting up on the button.
- Drag** To drag a graphical object (e.g. a window or an icon) from one location on the screen to another, place the cursor on the appropriate field of the object. Then press the mouse button and move the cursor to the required new location, such as dragging the object. Then release the button again.

Motif Widgets and Menu Types

This short paragraph provides some brief definitions of some terms which are often heard in connection with X-Windows. OSF/Motif provides a set of user interface mechanisms that are composed of data structures and procedures. They can be displayed in many ways, for example as buttons, boxes, labels, scroll bars, and so on, and are called **Motif widgets**.

However, in public domain software, often another class of widgets, called **Athena widgets**, is used. Their implementation and graphical visualization differs somewhat from the Motif widgets.

The Motif widget set provides three types of menus:

1. **Pulldown Menus:** are displayed as horizontal menu bars with (normally) several fields called labels or buttons. When you activate a label, it will display a vertical
2. **Popup Menu.** A popup menu also offers several labels to select. It disappears again when released after execution.
3. **Option Menus** are very similar to popup menus and offer a one-of-many selection. However, when the selection is complete, the selected label remains

visible in the menu, and only the not selected alternatives disappear.

Active Windows

The window which currently accepts input is called the **active** window and differs from all other windows by a differently colored frame. The active window sits on top of the window stack by default.. The other windows, sometimes referred to as **background** windows, currently do not accept input.

The processes running in the background windows, however, are still active (in background!) and are even able to produce output. To make a background window to the active one, just move the mouse to any part of it and click with the left mouse button.

1.7 GSI Customization

1.7.1 Environment Variables

There are parameters in the shell that define parts of your working environment and can be set individually at the command prompt, in shell scripts, or in user specific profile files.

- Shell parameters that are **local** to your login shell and not passed to any subshell or subprocess are called **shell variables**.
- Shell parameters that are **global** are called **environment variables**. They are valid in the current shell, where they are set, and in all subshells. They are not valid, however, in 'higher' shells, from where the current shell is invoked as subshell. To be global, shell parameters must be **exported** (see below).

There is one restriction you should keep in mind, when you define environment variables in a shell script (see 2.4 on page 2-3). You must invoke the script with a **preceding dot**, if the environment variables shall be valid also in the current shell:

```
. myscript
```

Otherwise the environment variables are valid only within your script and all subprocesses invoked by that script.

A set of environment variables is already defined by the operating system and can be changed by each user for his personal working environment. In addition, you can also define new environment variables, of course. The

profile files as an interface for initialization at login are described below.

A list of the currently valid environment variables may be obtained with the command

```
printenv
```

In the following examples, for simplicity the command prompt is assumed to be \$.

Setting Environment Variables

Your default printer, for example, is defined by the environment variable **LPDEST**. In section 4.4 on page 4-6 you can get a list of the available printers and their names and characteristics. To change the default value of LPDEST from pshpad to pshpa, enter the command

```
$ export LPDEST=pshpa
```

Getting Values of Environment Variables

The contents of environment variables can be displayed by the `echo` command. To get the value, the name of the environment variable must be preceded by a \$-sign:

```
$ echo $LPDEST  
pshpa
```

The value pshpa is shown according to your redefinition in the example above.

A common mistake is to forget the \$-sign. Not the value, but the character string is printed:

```
$ echo LPDEST  
LPDEST
```

Again you should keep in mind that Unix is case sensitive, that means environment variables such as lpdest or LPdest, for example, are not defined normally.

A More advanced Example

If you want to have your current host name and working directory available in an environment variable named MyEnv you can proceed as follows:

```
$ export NODE='hostname'  
$ export MyEnv=$NODE:$PWD  
$ echo 'My environment: $MyEnv'  
My environment: rzri6f:/u/goeri
```

With the first statement, a new global variable NODE is defined. NODE gets the output of the command `hostname` as value, which is achieved by enclosing the command name in single backquotes. Then the values of NODE and of the environment variable PWD, which is already provided by the system and contains always the current working directory, are put together in MyEnv. Finally, to control the success of this action, the contents of MyEnv is printed in your terminal window, prefixed by some text.

A Summary of some useful Environment Variables

In the following, some useful environment variables provided by the system are listed in alphabetical order:

DISPLAY: See section 1.2.3 on page 1-4.

EDITOR: The default editor is `vi`.

ENV: The name of a shell script that is executed each time when a new shell is invoked. This shell script is used, for example, to define common alias names, which should be available through the whole environment. A common default in the Korn shell environments is

```
~/ .kshrc
```

The tilde '~' specifies your home directory. The file ~/ .kshrc is also in use at GSI for your private shell customization. However, in order to provide a mechanism for common and platform independent shell customization, at GSI, the default for ENV is set to

```
/usr/local/bin/.kshrc
```

This common shell script, in turn, invokes your 'private' shell script ~/ .kshrc, if existing. Therefore you should not overwrite the value of the environment variable ENV, because the common GSI shell customization would be unavailable for you.

HOME: The default directory after login. You switch to the home directory, when you specify the command `cd` (change directory) without options. It is set by default to

```
/u/username
```

LPDEST: The default printer (default: pshp9ad).

PATH: Defines the search path for the shell when looking for commands in the system file structure, which is different in different Unix flavors. For example, in HP-UX the PATH variable is set by default to

```
/bin:/usr/bin:/usr/contrib/bin:
/usr/local/bin:/usr/bin/X11:
/etc:$HOME/bin:.
```

The directories in the path are separated by colons (:). The search order is from left to right. The environment variable HOME contains the value of the **home directory**, the default directory after login (see above).

The last directory in the PATH variable, indicated by '.', specifies the current working directory. If you have a lot of own commands and wish to put your current working directory to the first position in your search path, specify

```
$ export PATH=.:$PATH
```

PS1: The default command prompt in your shell. At GSI, it is set to

```
hostname: /u/username sequencenumber$
```

or, to give an example:

```
rzri6f:/u/goeri 33$
```

PWD: The current working directory. It may be indicated with '.' in file commands.

TERM: The terminal type for which output should be prepared. Depending on the Unix flavor, **aixterm**, **hpterm**, or **dxterm** are assumed as default for AIX, HP-UX, or ULTRIX systems, respectively. If necessary, you should overwrite it with **vt100**, **vt200**, **3270**, and so on, depending on your terminal type.

VISUAL: Is set to the value **emacs** to enable command line editing with emacs syntax (see section 1.7.3 on page 1-10).

1.7.2 The Profile Files

Using the Korn shell, there are normally four files for the customization of the environment:

```
/etc/profile
.profile
/usr/local/bin/.kshrc
.kshrc
```

```
/etc/profile
~/.profile
/usr/local/bin/.kshrc
~/.kshrc
```

The files in the users home directory are available for private customization. The two other files are available for common customization on all GSI platforms and can only be modified by the corresponding system manager. The file `/usr/local/bin/.kshrc` can only be utilized, if the ENV environment variable is set appropriately (see the description above).

On the HP workstations, the organization of profile files is a little bit different in normal Korn shell environments and in HP VUE environments. The standard Korn shell file `~/.profile` is meaningless for HP VUE users. Instead a file named `~/.vueprofile` is used, which in turn has no meaning without HP VUE. In order to have only one location containing the common customization settings for both environments, a new file named `~/.profile-common` has been created at GSI. It is invoked by `~/.profile` and `~/.vueprofile`.

1.7.3 Command Line Editing

Command line editing in the (Korn) shell can be done with emacs syntax (see section 5 on page 5-1). The available functions are summarized in table 1.2 on page 1-11 and can be classified as follows:

- A mask for the next command to be executed can be obtained with `Ctrl-P`, `Ctrl-N`, or `Ctrl-R` *string*.
- The cursor is moved within the command line with `Ctrl-b`, `Ctrl-f`, `Ctrl-a`, or `Ctrl-e`.
- At any position, characters can be inserted.
- With the backspace key `␣`, with `Ctrl-h`, or `Ctrl-d`, characters can be erased.

Incomplete file names can be completed. For example, if you type

```
.pr ESC ESC
```

the system completes the filename up to the end or to the first ambiguity. It probably shows

```
.profile
```

When you look for the file `.profile-common` (on an HP workstation), type in the next letter

Keys	Functions
Ctrl - P	get previous command from history file
Ctrl - n	get next command from history file
Ctrl - r <i>string</i>	(at least one Ctrl - P in advance required) recall the last command, which contains <i>string</i> anywhere in the command line
Ctrl - b	move cursor backwards in command line
Ctrl - f	move cursor forwards in command line
Ctrl - a	jump to begin of command line
Ctrl - e	jump to end of command line
←	delete before cursor
Ctrl - d	delete under cursor
Ctrl - h	delete under cursor
ESC ESC	complete a filename

Table 1.2: Command Line Editing Functions

```
.profile_ ESC ESC
```

and the system will expand the file name to

```
.profile-common
```

The command history is accessed from the file
~/.sh_history.

Chapter 2: UNIX Commands

2.1 Constructing a command line

In general, a command line consists of three parts, although not every command requires all three parts:

general Unix Command
Command_Name Command_Option(s) File(s)

There is not much to say about the command's name, except that most UNIX commands have short names. Command options are usually designated by a hyphen (or minus sign), followed by a single letter also called a **switch**. Sometimes you can type more than one letter after a single minus sign to indicate multiple options; sometimes you cannot. In a few instances, command options are designated by plus signs instead of minus signs. Many commands allow one or more input files to be named. Output files are generally, but not always, designated by an output option switch like `-o`. Another method of designating an output file will be discussed later in this chapter. The various options and filenames that follow the command are referred to, collectively, as **arguments**.

An Example

Consider the `ls` command, discussed in Chapter 3. The UNIX `man` pages, an online help facility in the UNIX environment, show 21 possible options for this command:

ls command options		
Name	Command Options	File(s)
ls	[-RadCxmlnogrtucpFbqisf]	[name...]

In case of this command, there are 22 different options - 21 switches plus one no switch - you can use. The brackets which are not to be typed on the command line, indicate that all option switches are optional, never required. More than one option can be typed after a single minus sign though. Finally the word name indicates that you can type at least one directory or file name after the options. See the UNIX online help command `man` for further information.

2.1.1 Redirection of Input and Output

Unix regards the terminal's keyboard as its **standard input: `stdin(0)`** and the terminal's screen as **standard output: `stdout(1)`**. However, with most UNIX commands it is possible to **redirect** the input and output of a command. The symbols used in a command line to request redirection are the **less sign (<)** and the **greater sign (>)**.

Redirection of Input

In case you want to use a file as input to your program, type

```
prog < file
```

Redirection of Output

In case you want to redirect the output of your program to a file, type

```
prog > file
```

To append the output to an existing file rather than to overwrite it, use the redirection symbol `>>` :

```
prog >> file
```

To suppress output redirect it to the null device like :

```
prog > /dev/null
```

2.1.2 Pipelines

In addition to redirection of input and output, UNIX can connect two processes with a **pipe**, so that the output of the one process becomes the input for another. The symbol for a pipe is the **vertical bar (|)**. It is possible to set up multiple pipelines. Commands that appear in pipe statements may include all the usual options and file designations.

```
man ls | lp
```

This command pipes the output of the `man ls` command to the printer via the `lp` command.

2.2 Regular Expressions

When file and directory names are used you can specify some special characters as pattern that the shell matches against the file names in a directory. These special **pattern-matching characters** are :

- * Matches any string, including the null string
- ? Matches any one character

- [. . .] Matches any one of the characters enclosed in square brackets
- [! . . .] Matches any character other than one of the characters that follow the exclamation mark within square brackets.

Inside square brackets, a pair of characters separated by a - (hyphen) specifies a set of all characters lexically within the inclusive range of that pair, so that [a-dy] is equivalent to [abcdy].

Using pattern-matching characters in file names on the command line has some restrictions. If the first character of a file name is a . (dot), it can be matched only by a pattern that begins with a dot. For example, *file matches the files myfile and yourfile, but not .myfile or .yourfile. Use the pattern .*file to match these files.

The character @ at the end of the pattern is ignored during the matching. However, the @ is appended to the corresponding component of the matched file names to allow hidden directories to be referenced directly.

If the pattern does not match any file names, the pattern itself is returned as the result of the match.

2.3 Quick Reference of Commands

This reference summarizes frequently used UNIX commands, special characters used to identify directories, and special characters used on the command line. More detailed information on each command, including a complete list of options, can be obtained with the man command.

2.3.1 Managing Directories



pwd	display the path name of the working directory
cd <i>dir</i>	change working directory to <i>dir</i>
mkdir <i>dir</i>	create directory called <i>dir</i>
rmdir <i>dir</i>	remove (delete) directory called <i>dir</i> . <i>dir</i> must be empty

2.3.2 Managing Files

ls	list contents of working directory
ls <i>file</i>	list <i>file</i> if it exists in working directory
ls <i>dir</i>	list contents of the directory <i>dir</i>
ls -l	list additional information on directory contents
ls -a	list all files including hidden files
cp <i>file1 file2</i>	copy <i>file1</i> to <i>file2</i> (overwrites <i>file2</i>)

cp <i>file dir</i>	copy <i>file</i> into directory <i>dir</i>
mv <i>file dir</i>	move <i>file</i> into directory <i>dir</i>
mv <i>file1 file2</i>	move <i>file1</i> to <i>file2</i> (overwrites <i>file2</i>)
rm <i>file</i>	remove (delete) file
rm -i <i>file</i>	ask for confirmation before removing (deleting) <i>file</i>
more <i>file</i>	displays contents of <i>file</i> , one screen at a time
cat <i>file</i>	displays contents of <i>file</i>
chmod <i>arg file</i>	change read/write/execute permission of <i>file</i>
chmod <i>arg dir</i>	change read/write/execute permission of <i>dir</i>

2.3.3 Managing Jobs

 - 	stop current job
ps	list process by process identifier
ps -fu <i>user</i>	full listing of all processes of user <i>user</i>
kill <i>PID</i>	stop process with process identifier <i>PID</i>
jobs	list your jobs by job number
kill % <i>jobnr</i>	kill job nr. <i>jobnr</i>

2.3.4 On-line Help

man <i>command</i>	display manual entry for <i>command</i>
man -k <i>keyword</i>	list manual pages that pertain to <i>keyword</i>
learn	on-line tutorial (AIX only)
info	on-line tutorial, help pages and manuals, stored on CD-ROM. Can only be used with X-Windows Terminals.

2.3.5 System Information

who	list users logged onto system
who am i	displays your logon ID
finger <i>user</i>	displays information on <i>user</i>
passwd	change password

2.3.6 Utility Programs

sort <i>file</i>	sort contents of <i>file</i> , send result to standard output
grep <i>pattern file</i>	look for <i>pattern</i> in <i>file</i>
uniq <i>file1 file2</i>	delete repeated lines in <i>file1</i> , write new version to <i>file2</i>
wc <i>file</i>	count the number of lines, words, and characters in <i>file</i>

<code>echo <i>string</i></code>	write Paramstring to standard output, translate special characters
<code>find <i>path</i></code>	search the directory tree <i>path</i> (see man page for more details)
<code>tar <i>file</i></code>	write to or retrieve files from an archival storage media
<code>compress <i>file</i></code>	compresses the file and writes <i>file.Z</i>
<code>uncompress <i>file.Z</i></code>	restores <i>file</i> from compressed file

If the file resides in another directory than your current working directory, and if you don't want to switch to it, specify either the full (**absolute**) path name or the path name **relative** to your current working directory. Let's assume, for example, your script resides in /u/goeri/util, and your current working directory is /u/goeri. You can enter either

```
util/myscript
```

using the relative path name (with no / at the beginning), or

```
/u/goeri/util/myscript
```

2.3.7 Directory Identifiers

<code>~</code>	your home directory
<code>.</code>	the working directory
<code>..</code>	the parent directory (one level up within hierarchy)
<code>/</code>	root directory

specifying the absolute path name (with / at the beginning).

When you define **environment variables** (see section 1.7.1 on page 1-8) in your script, you should invoke it with a preceding dot, if the environment variable shall be valid in the current shell:

```
. myscript
```

2.3.8 Special Characters

<code>*</code>	match any character
<code><</code>	redirect standard input
<code>></code>	redirect standard output
<code> </code>	send standard output of first command to standard input of second command (pipe)
<code>&</code>	put job in background

If not, the environment variables are valid only within your script and all subprocesses invoked from there.

2.4 Shell Scripts

The commands described in the previous examples can be executed when typed at the command prompt, but they can also be written to files and executed as command lists or **shell scripts**, which correspond to TSO CLIST in MVS or DCL in VMS. For example, if you have a command list in a file named `myscript` in the current working directory, then just enter

```
myscript
```

at the command prompt. The command list will be executed. If not, you should check two things :

1. Is the file containing your shell script **marked as executable**? See section 3 on page 3-1 for more information!
2. Check with the `ls` command if the required file really exists in your current working directory!

Chapter 3: Files and Directories

3.1 The UNIX File System

UNIX has a structured filesystem that contains three kinds of files:

- directories** which store the names of other files including other directories;
- ordinary files** which store text, source programs, and object code; and
- special files** which correspond to peripheral devices.

3.1.1 Naming Directories and Files

The **root** directory is identified by a single character: slash (/). To name one of the major directories directly under root, type slash (/) to represent root, followed by the directory's own name, as in /usr. The slash in front of usr tells you that usr is a subdirectory of root. An example of a typical Unix file system is shown in figure 3.1 on page 3-2. At least a UNIX System will have:

```
/      root directory
/usr   usr directory
/bin   binary directory
/dev   device directory
/etc   miscellaneous directory
/tmp   temporary directory
/u     user directory
```

For big amounts of data at GSI :

```
/d     data staging directory
/s     scratch directory
```

To identify the user's home directory, type another slash after /u, followed by the account name, as in /u/otto. The first slash refers to the root directory, u identifies the **parent directory**, and otto is a **subdirectory**. At the end of this Chapter you can find more information on how the /u, /d and /s filesystems are organized at GSI.

3.1.2 Rules for Naming and Accessing Files

The rules for naming and accessing files and directories are closely related to the structure of the UNIX file system:

- The root directory is identified by a slash (/).

- A simple filename can be any combination of 1 - 14 characters **other than** slashes (/), asterisks (*), question marks (?), quotation marks (") or ('), square brackets ([or (]), dollar sign (\$) or control characters.
- A path name is a sequence of directory names, possibly followed by a simple filename, with each pair of names separated by a slash (/).

To avoid misinterpretation, the safest characters to use for simple filenames are letters of the alphabet, numbers, periods (.), hyphens (-), and underline (_). Note: in UNIX, upper and lower case are **not** the same !

The directory permanently assigned to you is called your **home directory**; this is the directory to which you log on. Any directory to which you may move after logging on (including your home directory) will be called your **current directory**, or **working directory**, as long as you remain in that directory. The directory which is one level above your current directory in the file system is called your **parent directory**. UNIX provides shorthand symbols to indicate your current directory (.) and your parent directory (..). If a path name used to access a file begins with a slash (/), then the search for the file begins at the **root directory**. Such a path name is called an **absolute path name** or **full path name**. If a path name begins with a simple filename, then the search for the file begins at your current directory. Such a path name is called a **relative path name**.

3.2 Working with Directories

3.2.1 Displaying the contents of a directory: **ls**

To sort and display the names of all the directories and files that reside in you current directory, use the **ls** command:

```
$ ls
file1
file2
file.3
Mail
$
```

3.2.2 Changing the Working Directory: **cd**

To change your working directory, that is to move to another directory, use the **cd** command:

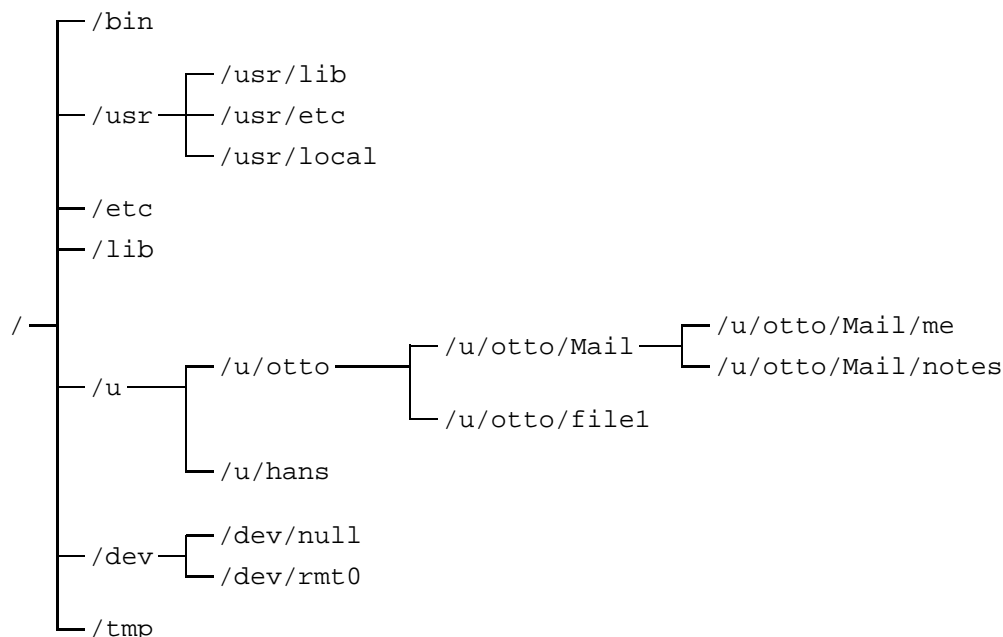


Figure 3.1: Part of a typical UNIX file system

```
$ cd /u/otto/Mail
$
```

move back to the parent directory, and then use the `rmdir` command:

3.2.3 Determining Your Working Directory: `pwd`

To find out the name of your working directory at any moment, use the `pwd` command:

```
$ pwd
/u/robin
$
```

```
$ cd /u/useless
$ pwd
/u/useless
$ rm -i *
$ cd ..
$ rmdir useless
```

If you try to remove a directory that is not empty, you will see a warning displayed. You may use the following shorter method instead of the above:

3.2.4 Creating a New Directory: `mkdir`

To create a new subdirectory within your current working directory, use the `mkdir` command:

```
$ mkdir messages
$
```

This command will create a new subdirectory called `messages`.

```
$ rm /u/useless/*
$ rmdir /u/useless
```

or:

```
$ rm -r /u/useless
```

3.2.5 Removing an Existing Directory: `rmdir`

To remove an existing directory from your working directory, move to the target directory, delete all its files,

3.2.6 Renaming a Directory: `mv`

To change the name of a directory, use the `mv` command:

```
$ mv old.name new.name
```

3.3 Working With Files

3.3.1 Displaying the Contents of a File: `cat`

To display the contents of a file, use the `cat` command. It simply displays the contents of the file or several files on the screen (standard output):

```
$ cat file.3 file.1
```

Combining Files

Another function of the `cat` command is to combine files, or concatenate files with the result stored in another file e.g.:

```
$ cat file.1 file.2 > file.3
```

Avoid storing the result in one of the original files, since this will the original file to be overwritten.

3.3.2 Renaming a File: `mv`

You can use the `mv` command to rename a file or to move it from one directory to another. To change the name of a file, enter a pair of command like this:

```
$ cat new.file
cat: cannot open new.file
$ mv old.file new.file
```

The `mv` command will change the file's name whether the new filename exists or not. The `cat` command makes sure that a file will not be replaced and be lost.

3.3.3 Copying a File: `cp`

To make a duplicate copy of a file, use the `cp` command:

```
$ cp file.one FILE.ONE
```

This command will make a copy of `file.one`. As a reminder lower and capital letters are **different** filenames.

3.3.4 Deleting a File: `rm`

To delete a file, use the `rm` command:

```
$ rm file.1
```

This form of the command will delete the file `file.1` immediately. To confirm before proceeding to delete the file, add the `-i` option:

```
$ rm -i file.1
```

3.4 File and Directory Permissions

UNIX allows you to access other files and directories in the system, but only if you have permission from the owner of those directories and files.

3.4.1 Determining Permission: `ls -l`

To determine the permission associated with a given file or directory, use the `ls -l` command to display the contents of the directory:

```
$ ls -l
total 501
-rw-r-----
1 user group 108 Oct 15 19:10 file.1
-rwxr-x---
1 user group 6452 Oct 15 17:15 program.1
drwxr-xrw-
1 user group 512 Oct 15 19:13 letters
```

The first character indicates the type of the file

```
- ordinary files
d directory
l links
```

The remaining nine characters represent three sets of three characters: one set for the individual user, one for the user's working group, and one for all other users. Spread out the characters of the display above to explain the groupings:

Type	User	Group	Others	
-	rw	r-x	---	program.1
d	rw	r-x	rw-	letters

The permissions given are for **reading**, **writing**, and **executing**. They have different meanings for ordinary files and directories. For an **ordinary file**, permissions are defined as follows:

read permission means you may look at the contents of the file

write permission means you may change the contents of the file

execute permission means you may execute the file as if it were a UNIX command.

For a **directory**, permissions are defined as follows:

read permission means you may see the names of the files in the directory

write permission means you may add files to and remove files from the directory

execute permission means you may change to the directory, search the directory, and copy files from it.

The characters used to represent these permissions are:

```
r      read permission
w      write permission
x      execute permission
s      set user (or group)ID
-      permission denied
```

3.4.2 Changing Permission: **chmod**

You can make changes to permissions by entering a **chmod** command. It allows the owner of the file to add (+) or remove from (-) existing permissions. It also allows the owner to clear existing permission and assign all permission from scratch; this is known as assigning permissions absolutely (=). The **chmod** command affects any of the three types of access for any of the three categories of UNIX users, using one-letter symbols in the following order (left to right):

```
u      owner (user)
g      File's group
o      all others
a      all (default)

+      add permission
-      remove permission
=      absolute permission

r      to read
w      to write
x      to execute
```

Caution: It is possible for you to lock yourself out of one of your own files with **chmod**. Be careful when you type it.

Example:

```
$ ls -l
-rwxr-xr-x 1 otto rz 487 Jul 30 10:21 psab
$ chmod o-x psab
$ ls -l
-rwxr-xr-- 1 otto rz 487 Jul 30 10:21 psab
```

In the above example, the first `ls -l` shows the default permissions for a script, which is executable and readable by everyone, but writable only by the owner. After the `chmod o-x` command, the execution permission for others is removed.

3.5 Filesystems and Links

This section explains why and how the physical and logical structure of filesystems can (and usually will) differ. Skip this section on the first reading if you are completely inexperienced with UNIX.

3.5.1 Filesystems

Looking at an UNIX filesystem by directories gives the logical view of how data are organized. This logical view hides the physical and implementational aspects of how the filesystem is build up.

A file resides physically on some medium like harddisk or CD-ROM. Physical filesystems are sets of files which are already hierarchically organized and reside on contiguous parts of the medium.

There are many different ways of how such a filesystem can be implemented. Some important ones are: The UNIX Filesystem (System V), The Fast Filesystem (BSD), The Epoche Filesystem (Transarc), or The Journaled Filesystem (AIX). There are also ways to access filesystems (in total or parts) that reside on remote hosts, examples are: The Network File System “NFS” (SUN), The Andrew resp. Distributed Filesystem “AFS”/“DFS” (Transarc resp. OSF).

What logically appears as *the* filesystem of the local host (everything that is visible below /) are several physical filesystems (local and remote) that are *mounted* together. Mounting means to take the top directory of a physical filesystem and put it on a directory of an already mounted filesystem. The directory on which the filesystem is mounted is called *mount point*, it should be empty, because while there is a filesystem mounted on it its contents in not accessible.

To see the physical organization of the filesystem use the **df** command on AIX, on HP use **bd**. As argument you can specify every file or directory, the **df** command will show in which physical filesystem the file or directory resides. Without argument all filesystems of that host are displayed. The information given is:

filesystem A device name if the filesystem resides at the local host, or *host:Path* for NFS filesystems.

size Total size in kilobytes.

used How much kilobytes are used (HP only).

free How much kilobytes are available for use.

percentage How much percents are used.

mountpoint Directory on which the filesystem is mounted and under which its contents logically appears.

3.5.2 Links

To make a file available under different names and/or in different directories without copying them (and thus wasting disk space and running the risk of inconsistencies) it can be *linked*. A link is simply an entry in a directory that refers to the existing file.

Be careful using links, they can be confusing. A link may look like a file but it is not. Copying **cp** or moving **mv** links produce files and not links again. To copy links use the **tar** command.

On the AIX machines at GSI the /u directory is linked to /home

3.6 /u, /d, /s and /tmp

There are several filesystems at GSI where users can store data: the standard UNIX HOME directory /u/<user>, disk space for data staging /d/<group>, scratch space /s/<group> and the local temporary /tmp filesystem.

The /u directories are automatically backed once a day, /d and /s are not backed up. The scratch space only keeps active files, files not accessed for more than 8 days are automatically deleted.

In both filesystems, /d and /s it is recommended to create user subdirectories in the groups.

The /d and /s filesystems are equivalent on the AIX and HP-UX systems and represent one unique area to store data. In addition a user can store temporary data on every system in the local /tmp filesystem. The data is automatically deleted after more than 3 days of inactivity.

Chapter 4: Basic Services

In this chapter we assume that the reader has successfully logged in, is somewhat familiar with the basic Unix commands and understands the fundamentals of the Unix file system. We explain here the use of some everyday infrastructure services such as getting help, mailing, printing, remote login, remote execution and the backup/recovery mechanism. Editors are not mentioned in this chapter but can be found in chapter 5 on page 5-1. If you are interested in text processing and page layout please refer to chapter 6.1 on page 6-1

4.1 Help and Documentation

4.1.1 How to get Help

To get online help in a Unix session, there are in principle three methods available:

- command `man`
- command `info` (on AIX/6000 and HP-UX platforms only, with graphics terminal only).
- help button in HP-VUE environment (HP-UX only)

4.1.2 The `man` Command

The `man` command is the standard Unix command to get online command descriptions. Specifying for example

```
man man
```

gives the first page of the description of the `man` command on your screen. Following pages are displayed after pressing the `Enter` key. Output can be interrupted with `Ctrl-C`.

4.1.3 The `info` Command

The complete manual information available on CD-ROM can be obtained in a comfortable environment with several screens using the `info` command. The implementations on AIX and HP-UX are similar in functionality, but different in the layout of the windows and menus. Both systems provide extensive help menus on functionality and usage and are rather self-explaining.

In principle, the following functions can be performed:

- Any manual page can be **browsed** which allows to read the manuals online.

- All or parts of the manuals can be scanned for **keyword search**. In this mode, a specific text expression can be entered to be searched for. If found, the number of occurrences will be indicated, and the corresponding manual parts are located and offered for reading.
- On the RS/6000, the AIX command descriptions can be accessed via menu offering the first letters and command categories. To access the information on **fortran** or **C** you have to use the command `info -l xlf` or `info -l xlc`

4.1.4 The VUE Help button

In the HP VUE environment you can press the help-button (marked by a question mark (?) in the lower row of the HP-VUE workspace manager) and you will get a help-window with information on the following topics, which can be selected by a mouse-click on the appropriate line:



- Tutorial on HP VUE
- man pages
- Help on Customization of HP VUE
- Fortran and other applications
- ...

4.2 World Wide Web at GSI - Getting Help and Information

At GSI we decided to provide all our information about GSI related topic through the World Wide Web. The World Wide Web, in short WWW or W3, has the great advantage that nearly every information could be distributed and reached by an easy to use manner over the whole Internet. (s. Appendix Internet)

At GSI starting the WWW browser, the user interface to the WWW, could easily be done by typing

```
www
```

Or by specifying the URL at startup, e.g.


```
www http://www.gsi.de/gsi.html
```

The **URL**, the **U**niversal **R**esource **L**ocation, is the kind of specifying the location of the desired information:

- choose the kind of service or protocol e.g. `http:` -the Hypertext Transfer Protocol

- select the server you want to get the information from: `//www.gsi.de`

- specify the location of information: `/gsi.html`

As the name URL as resource specification expresses, `http` is not the only kind of service you could reach via WWW. Other services are `ftp`, `gopher`, or `news`. So if you discover something similar to the above URL, you could easily load this by typing `www` followed by the URL.

Some examples:

```
www ftp://ftp.gsi.de/pub/
www news:comp.os.linux.announce
```

The command `www` will start a linemode WWW browser on all dump terminals and the Mosaic WWW browser for the X11 user.

4.2.1 Navigation though the World Wide Web

The linemode browser could be handled by typing the selectable numbers referenced in rectangular parenthesis or using the keys described in the last row.

As X11 is the standard Graphical User Interface, the Mosaic WWW browser will be described in detail in the following chapter.

On startup the Mosaic browser will show the GSI - Homepage:



Hypertext Markup Language and Links

As you can see, it's possible to include images, structuring the document by sections and description lists. This description language is called Hypertext Markup Language or in short html. This Hypertext Markup Language makes it possible to have something like links, which "links" information together. Links can also point to images, sound files, movies, files and more. When a link is selected that points to a resource which the WWW browser cannot handle internally, it attempts to send the data file to an external viewer. For example, GIF images are sent, by default, to the program `xv`, a commonly-used image viewer. Most sound files are sent, by default, to the `showaudio` program.

Following Links

The link could be selected in the Mosaic browser by clicking with the mouse on the undelined item or in the outlined image. A *single-click* with the left mouse button on a link causes the WWW browser to retrieve the document associated with that link, and display it in the Document View window. A *single-click* with the middle mouse button also causes Mosaic to follow the link, *and* open a new Document View window on top of the existing one. From that point, either Document View window can be used for further navigating.

Back through the Window History

Each open Document View window maintains a window history, recording every document it visits. Clicking with the left mouse button on the **Back** button on the bottom control panel travels back through the window history, as does selecting *Back* from the **Navigate** menu, or hitting *b* (a hotkey) while the mouse cursor is in the Document View area. After travelling back through the window history, subsequent hyperlinks will write over the existing window history from that point, as expected.

Forward

After moving back one or more documents through the window history, the **Forward** button will travel forward through the already-determined window history, as if the previously-selected hyperlinks are selected again. Selecting *Forward* from the **Navigate** menu or hitting the hotkey *f* while the mouse cursor is in the Document View area has the same effect.

The Window History List

Travelling back and forward through a Document View window's history can also be achieved with the *Window History* feature under the **Navigate** menu. Mosaic can jump to any document in the window history, forward or back. Simply *double-click* on any document title, or highlight a title and hit the *Go To* button. Hitting *lowercase h* with the mouse cursor in the Document View area also causes the window history list to pop-up.

Home Document

When Mosaic is first executed, it retrieves and displays the "home document." By default, this is the GSI Home Page, but you can specify an environment variable `WWW_HOME` pointing to your preferred starting point. To jump to the home document in any Document View window, click the **Home** button or use the *Home Document* feature under the **Navigate** menu. Note that Mosaic will add the home document to the window history, rather than jumping to the first node in the window history list (which is *not* necessarily the home document).

Opening New Windows and Cloning

The **New Window** button can be used to pop-up an additional Document View window on top of the existing one(s). The new Document View window will retrieve and display the home document. On the other hand, the

Clone button can be used to open a new Document View window displaying the same document currently being viewed. The *New Window* and *Clone Window* features can also be selected from the **File** menu, or with the hotkeys *n* and *c*.

Closing Windows

Any Document View window can be closed with the **Close Window** button, the *Close Window* feature under the **File** menu, or the *ESC* hotkey. Only the current Document View window will be closed. If the current window is the only window currently open, the entire application will be shut down when the window is closed.

Aborting Document Transfers

Occasionally, you may want to abort the retrieval of large documents or documents stored on distant servers which require greater transfer time. At any point in a data transfer process (hostname lookups and certain stages of direct WAIS queries excepted), you can click on the icon in the upper right corner of the window to stop the current network action.

more Information ...

A more detailed description about the Mosaic browser and WWW you will find under the **Help** menu:

Mosaic - NCSA Mosaic for X Documentation:
(<http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/mosaic-docs.html>)

HTML - A Beginner's Guide to HTML
(<http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html>)

URL - A Beginner's Guide to URLs
(<http://www.ncsa.uiuc.edu/demoweb/url-primer.html>)

4.2.2 Information about GSI

At GSI we have installed our own Information Server. It provides for you the GSI Homepage and, from this starting point, all GSI related information.

The GSI Homepage

The GSI Homepage should represent the GSI and is divided in the following sections:

- **General Information on GSI**

for Information on first connect to GSI

- **Events**

GSI - related events schedule

The next sections mirror the logical organisation structure of the GSI:

- **Accelerators** as there are:

UNILAC, SIS, ESR

- **Research** separated into:

Nuclear Physics, Atomic Physics Biophysics/Cancer Therapy, Plasma Physics, Material Sciences, Theoretical Physics

- **Infrastructure** with the subsections:

Computing, Detector Laboratory, Target Laboratory, Library

Not all sections have by now provided their information through the WWW, but this is in progress, so you will get nearly all information about GSI related special topics through the WWW.

Information at GSI Computing Center

Information about GSI Computing Center related topics, you will find starting at the Computing Homepage. You can walk from the GSI Homepage to the Infrastructure section: Computing (URL: <http://www.gsi.de/computing/dvee.html>).

Have a look especially on the **DV & EE Palaver** Schedule, where new themes in the Computing Environment will be discussed. For more information about UNIX look at the Computing Environment section: UNIX. There you will get e.g. Manpages, Printer information and status and the Software Map (work in progress).

If you have suggestions or questions have a look on the Questions and Answers item on the GSI Computing Homepage.

Searching Information

Searching in the WWW is a very serious problem, but there are a lot of ways on doing this, starting from the Miscellaneous section on the Computing Homepage to the Web Crawler. You could enter there any search term and it will be searched through most documents on the Web.

For only searching a file on the ftp servers, have a look at Archie, who has got a WWW interface.

How to put your own Documents in the Web

If you decide to put your own documents on the WWW at GSI we recommend to talk to your project chief, as there is in every group somebody who is responsible for managing this group's own WWW page.

For more information or help contact us at the user consulting (phone GSI-555).

4.3 Mail Facilities

There are several different kinds of mailers available on our UNIX systems. AIX, HP-UX, and ULTRIX provide the standard `mail` or `mailx` command. In addition there are public domain mailers `elm` and `xmh`, which allow a comfortable and easy to use mail handling.

Mail addresses are of the general form `user@node.domain`. For a few examples of valid mail addresses see the description of the `$HOME/.mailrc` alias definition file below. You can send mail to any internet-, BITNET-, and GSI-DECnet-nodes. When mailing to BITNET use addressing of the form `user@node.bitnet`, for GSI DECnet nodes use `v6000a` as gateway.

4.3.1 Simplified E-Mail Addresses

In addition to the RFC822 standard email address of the form `user@node.domain`, a registered user of the GSI computing environment can be addressed by a simplified mail address `userpart@gsi.de`. This mail alias consists of a user specific part left of the `@` and the worldwide unique domain name `gsi.de` behind the `@`.

The user specific part can be derived from the user's real name. The regular form is combined of the first letter of the first name and the complete last name, e.g. `U.Meyer@gsi.de`.

Some rules for the user specific part are

- The user name will be taken from the user registration form.
- The first name will be abbreviated to the first letter.
- If the first name consists of more than one part, each part will be abbreviated to the first letter, separated by dots.
- In case of equivocation, the next letter of the first name will be appended.
- German umlauts will be changed (ä to ae, ö to oe, ü to ue, ß to ss).

Request for new mail alias addresses or for changes of existing ones must be directed to `postmaster@gsi.de`.

4.3.2 The Standard Mailer

Example: mail on AIX:

Send a mail message:

```
mail username or mail aliasname
```

Type the message text, terminate and send your message with CTRL-d or with a last line containing a single . period in column one.

Aliases are defined in the **\$HOME/.mailrc** file. This file may contain lines like the following.

```
alias hw      H.Weber@gsi.de
alias um      rz02@mvs.gsi.de
alias rb      brun@cernvm.cern.ch
alias he      goofy@v6000a.gsi.de
alias body    user@cageir5a.bitnet
alias group   dob,goeri,schwab
```

Aliases may contain lists of addresses but not lists of aliases.

Mail a file:

```
mail user -s "next meeting" < dates.txt
```

Reading, forwarding, replying, filing, sorting and editing mail are done inside the mail utility. Invoke:

```
mail
```

then use any of the following subcommands:

```
m      send mail, invoke editor
h      display list of messages in your mailbox
?      help
d      delete current message
e      edit the current message
[n]    read message number [n]
-      read previous message
s      save current message in personal mailbox
s file save current message to file
s [n] file save message number [n] to file
c [n][file] same as s but do not delete message from incoming mailbox
r      reply to current message
a      display aliases
a hd   display alias hd
q      quit mail, discard deleted messages
x      quit mail, do not discard deleted messages
```

By default mail uses the vi editor. Insert a line into your **\$HOME/.mailrc** file to change the default editor to emacs.

```
set EDITOR=/usr/local/bin/emacs
```

If you want to work with your personal mailbox instead of the system mailbox, type

```
mail -f
```

Work with an arbitrary mail folder is started as

```
mail -f filename
```

You can forward all mail arriving for you on some node which you do not use frequently by creating a file **\$HOME/.forward**. This file should contain a line *user@othernode*, e.g.

```
otto@rzhp9a.gsi.de
```

This will automatically forward all incoming mail to *user@othernode*.

Mail on Ultrix works very much the same way.

4.3.3 The elm Mailer

elm is a screen-oriented electronic mail processing system. In interactive use, the main header index and mini-menu of commands are displayed upon initial invocation and at any point when the program is waiting for input.

You can also invoke elm by the mail button of the HP-VUE workspace manager (upper row, 4th left button).



You can easily send mail (m), reply (r) or forward (f) mail, save it to a folder (s) or delete incoming mail (d). By command a you enter the alias menu.

Aliases are stored in the file `~/elm/aliases.text`, which can be edited by any text editor. After having edited this file, the internal alias file has to be updated by the command `elmaliases`. In elm aliases may also contain a list of aliases.

4.3.4 The xmh Mail Interface

For X-terminal users, the xmh command provides a Motif (X-Windows) based user interface to the mail handling System (AIX only). Electronic mail messages may be composed, sent, received, replied to, forwarded, sorted, and stored into folders.

The file `~/mh_profile` can be edited and contains basic configuration parameters for the xmh interface. If it does not exist, you can create it by the `inc` command. You should do this before using xmh to incorporate new mail for the first time.

xmh starts with a single window, divided into four main areas:

- Six buttons with pull-down command menus.

- A collection of buttons, one for each top level folder. New users of xmh will have two folders, “drafts” and “inbox”.
- A listing, or Table of Contents, of the messages in the open folder. Initially, this will show the messages in “inbox”.
- A view of the contents of one message. Initially this is empty.

Some useful hints:

- The make the xmh icon indicate new incoming mail, start from the command line with the -flag command line option.
- To use Incorporate New Mail, the corresponding **inc** command must be selected (works only, if “inbox” is the folder being viewed).
- The header fields of outgoing mail will automatically be modified, if a file `~/Mail/components` contains the desired changes. The following examples inserts a Reply-To field with the global GSI mailing address and some other useful fields:

```

Example of file Mail/components

Reply-To:  Udo Meyer <U.Meyer@gsi.de>
To:
cc:
Subject:
-----

```

More information about xmh, its mail managing features and how to use the built-in editor may be obtained by `man xmh`.

4.4 Print Services

The GSI Unix print service is available on the login servers of the central Unix cluster. Access is possible to some printers installed at the output room of the compute center (2.223). Maintenance is done by the central operation personnel (phone 2515). Spread over the GSI further printers open to the users can be reached. We support the printing of plain text ASCII files and files in PostScript format.

Among the printers in the output room the printers **pshpa** and **psag** give ordinary black-white output on natural white paper, the printers **psaa** and **pshpa** give black-white output on plain white paper. These printers are

capable to do double-sided printing as well. Color printing is possible with the printers **psteka** on paper, **pslexa** on transparencies and the printer **p41** on both output media. Special transparencies are required, the printing costs are approximately DM 2.- per color slide. The output media size is DIN A4 in general. The printer **psaa** permits to select a DIN A3 output media size. Printing commands vary slightly from platform to platform. The basic commands are shown here in tabular form, for more details look at the examples below.

	Print	Status	Cancel	Env.Var.
IBM	<code>lpr</code> <code>lp</code>	<code>lpstat</code>	<code>cancel</code>	<code>LPDEST</code>
HP	<code>lp</code>	<code>lpstat</code>	<code>cancel</code>	<code>LPDEST</code>

The print queue names are derived from the name of the printers and appendices selecting a special output style. For instance in the case of the printer **pshpa** there are four dedicated print queues: **pshpa** and **pshpad** for single-sided and double-sided (duplex) output on natural white paper and **pshpaab** and **pshpadab** for single-sided and double-sided (duplex) output on plain white paper from the alternate bin.

The default print queue on all central Unix systems is the **pshpad** queue. The system default printer destination may be overwritten by the user setting the environment variable **LPDEST**. If you want to overwrite this system default, just add a line into your `~/profile` ksh-startup, e.g. `export LPDEST=psaad`. See Section 1.7.1 on page 1-8 for more information on shell and environment. You get a table with information about the available printers by typing `man printer`. The table is listed in the appendix A.1.

Examples for AIX:

```
lpr -Ppshpa -m doc.txt
```

prints **doc.txt** on **pshpa** and sends mail (-m) upon completion.

```
lp -dpshpa kaos.ps
```

on AIX the **lp** command is available too. Here the file **kaos.ps** is printed on the Postscript printer **pshpa**.

```
lpstat -opshpa
```

displays the status of the print queue **pshpa**. The output may look somewhat like this:

```

Queue Dev   Status Job Files   User  PP % Blks Cp Rnk
-----
pshpa laser READY
pshpa pshpa RUNNING 21 kaos.ps  op    0 0 3984 1 0

```

```
cancel 21 pshpa
```

will cancel that particular print job.

```
man lpstat | lp
```

prints the man page for command **lpstat** on the system default printer. In this case it is the printer **pshpa** and the print queue is **pshpad**.

```
man cancel | lp -dpsaa
```

This time the man page for command **cancel** is printed on the printer **psaa**.

Examples for HP-UX:

```
lp -dpsag -n2 .profile
```

prints two copies of your file `.profile` on printer **psaa**.

```
lpstat -opsag
```

displays the status of the print queue **psag**. You will get an output like this:

```

printer queue for psag
psag is ready and printing via network
psag-44 op priority ? from rzhp9a
        .profile                1723 bytes

```

```
cancel psag-44
```

That is how to cancel this particular print job.

For a comprehensive description of the commands we refer the reader to the man pages.

Larger jobs for one of the printers in the output room should not be printed during day time to grant all other users the quick access to the public print service. Typically it lasts more than 15 minutes to print 100 pages on a laser printer. You can easily schedule a print job at night time using the `at` command.

Example for scheduling a print job:

```
lp -dpsaa thesis.ps | at 23:00
```

The `at` command schedules the printing of your thesis.ps file at 23:00 (11 pm) that evening.

4.5 Local Networking Tools

On a host connected to a network you will encounter the situation that a command or file is not available on your local node. You will then want to work on a remote host (node) or transfer files between two nodes of the network.

The most common situation is to open a terminal window on a remote machine. This can be achieved by different methods. An easy and convenient way in a graphical environment (X-terminal, workstation graphical display) at GSI is to use the command `xt`, described in more detail in section ?? on page ??.

Some of the commands described below only work between *equivalent* Unix hosts. This applies to `rlogin`, `remsh`, and `rcp`. The effect users see on equivalent hosts is that when they are logged on to one host, they can work on all equivalent hosts with the above commands without further authentication. Equivalent hosts at GSI are currently the UNIX machines of the AIX Cluster (`rzri6f`, `clri6a`, ...), the HP Cluster (`rzhp9a`, `rzhp9b`) and `rzds5a`. The equivalence of hosts is maintained by the administrators of these hosts.

Other commands work on/between arbitrary hosts with almost any Operating System. To run such a command the host must be connected to the IP net and therefore be able to handle the TCP/IP protocol. These commands, namely `telnet` and `ftp`, allow much more connectivity but — of course — less transparency.

4.5.1 Opening Remote X-Windows

In order to open a remote terminal window with a display on your local graphical screen, proceed as follows:

1. You are required to have an account on the remote host.
2. You can open a remote window by typing at the prompt line:

```
xt remote-host
```

where *remote-host* can be:

<i>remote-host</i>	terminal type
mvs ¹	3270 emulation
mvsg ¹	3270 emulation with graphics
rzri6f	aixterm
clri6a	aixterm
rzhp9a	hpterm
rzhp9b	hpterm
rzds5a	decterm
any other UNIX	xterm

3. `/usr/local/bin` has to be included in your `$PATH` variable, in case it has not been already included by default. You may want to check with the UNIX command `echo $PATH`.
4. If the `DISPLAY` variable has not been set, a message will be displayed asking for the name of your display (e.g. `XWTAN:0`).

4.5.2 Remote Processing

Besides the `xt` there are other commands to work on a remote host from your local session:

`telnet` and the `r...`-commands.

With the commands

```
telnet and rlogin
```

you can establish a session on a remote host from within your local session. Whereas the commands

```
remsh and rexec
```

do not perform a login on a remote host but execute commands there for you.

telnet

To enter the `telnet` environment issue the command

```
telnet
```

which will return the prompt

```
telnet>
```

now you can enter `telnet` subcommands.

For a complete list of subcommands and flags of the `telnet` command consult its man page. Here a list of some often used subcommands:

¹remote machine: `mvs.gsi.de`, IBM mainframe

```
quit    ends the telnet command
open    establishes a connection to a remote host
close   ends that connection
help    lists the subcommands with a brief explanation
user    login after establishing a connection with
        open
```

The first subcommand could be:

```
telnet> open hostname
```

This connects your terminal or X-window to the specified host and displays the logon logo. You can log on to the host supposed you do have a local account. After logging out of the remote host you will get the

```
telnet>
```

prompt. You can open the next host or `quit` from the `telnet` program and resume the local session.

Alternatively you can specify the remote host on invocation of the `telnet` program:

```
telnet hostname
```

This automatically opens the remote host and you see immediately the logon logo. Logging out of the remote host will now end the `telnet` program. You will not get the `telnet>` prompt. You are back to your local session.

rlogin

The “remote login” command

```
rlogin hostname
```

connects your current session or X-window to a login session on the specified host. Since your local host must be equivalent (see above) to the remote host you do not need to authenticate yourself to the remote system. You will get the command line prompt of the remote system. Logging out there resumes your current local session.

Usage Notes:

You must not omit the `hostname` parameter when using `rlogin`.

If you use options with this command you have to place them **after** `hostname`!

remsh

To execute a command on a remote host enter:

```
remsh hostname command
```

This command runs a “remote shell” which executes *command*. You can *not* work interactively on the remote host, all input to the command must be specified on the local command line, all output from the command is directed to the local standard output. I/O-redirection (>, >>, <, <<) redirects in- and output to and from the remotely executed command to *local(!)* files. If you want to redirect the in- and output to *remote* files use double quotes " around the redirection operators, for example ">".

Usage Notes:

On AIX *remsh* and *rsh* are synonyms.

The *remsh* command will *not* process the login profiles, but your shell startup file (usually the *.kshrc* for *ksh*). If you omit the *command* parameter on the *remsh* command the *rlogin* will be executed instead, which — of course — *will* process the login profiles!

If you use options with this command you must place them **between** *hostname* and *command*!

Examples:

All the examples below assume, that you are logged on to a host that is equivalent to *rzr16f*:

These examples do illustrate the I/O redirection mechanism with the *remsh* command.

```
remsh rzr16f cat .profile >> .profile
```

appends your profile on the *rzr16f* to your *local* profile.

```
remsh rzr16f cat .profile ">" .profile.old
```

your remote profile will be copied to a remote file named *.profile.old*.

rexec

With the *rexec* command you can execute a command on remote UNIX hosts you do have an account on regardless if the hosts are equivalent or not. The *rexec* command works in the same way *remsh* does with the difference you will be prompted for your name and password on the remote host.

```
rexec hostname command
```

4.5.3 File transfer

The two commands

```
rcp and ftp
```

allow file transfer between nodes in a network.

ftp

The *ftp* (file transfer protocol) allows to login to a remote host and execute *ftp* subcommands without leaving your current session on the local host. The *ftp* command works between various platforms, not only between UNIX systems. Invoke *ftp* by typing

```
ftp
```

to get the prompt

```
ftp>
```

Select the remote host by

```
ftp> open hostname
```

and you will be prompted for login. when you invoke *ftp* by

```
ftp hostname
```

the *open hostname* is implicitly executed and you will be directly prompted for login. For the various options on the command line consult the man page or the *info*. After a successful login you will get the *ftp>* prompt. Issue the *ftp* subcommands which allow you to navigate through the remote filesystem, display the remote directory and transfer files between the remote and the local host in both directions. Some often used *ftp* subcommands are:

<i>quit</i>	ends the <i>ftp</i> command
<i>cd</i>	changes directory on remote host
<i>lcd</i>	changes directory on local host
<i>mkdir</i>	creates a new directory on the remote host
<i>pwd</i>	prints the path that is current on the remote host
<i>put</i>	transfers a file from local to remote
<i>get</i>	transfers a file from remote to local
<i>binary</i>	the data are transferred without conversion
<i>ascii</i>	the data are converted due to different character representation on the sending and receiving host.
<i>help</i>	displays <i>all</i> available subcommands and gives a short description of them.

You will find more information on the *ftp* command in the man pages, *info*, and TCP/IP manuals.

Usage Notes:

Some explanations on data conversion:

When you transfer a file in text mode `ascii` mode is the default. This is the mode to transfer e.g. a `TEX` source.

To transfer compiled programs or object files you have to switch to `binary` mode so the transfer will be done on a “bit by bit” mode without any translation. This is the mode to transfer e.g. a `dvi` file produced by the `TEX` program or a `Ntuple` file produced by `paw`.

Examples:

The following example illustrates how to use `ftp` within a shell script.

```
# getfiles: get a few files from MVS
#
# customize the next lines:
user=XY12          # replace this with your id
locpath=...       # specifies where to put
rempath=...       # from where to get
#
echo " "
echo "Opening FTP connection to MVS"
echo "for user" $user
echo " "
ftp -n mvs <<EOF # invoke FTP with
                # next lines as input
user $user      # specifies remote user id
lcd $locpath    # set the local path
cd $rempath     # set the remote path
get file1      # with EBCDIC/ASCII conversion
binary
get file2      # without conversion
quit           # terminate FTP
EOF            # terminate input to FTP
```

You will be prompted for your password on the remote host.

Some advanced features :

If you transfer files from hosts with other operating systems (e.g. VMS, DOS, MVS) you will appreciate some more advanced `ftp`-subcommands. For a complete description see the man-page of `ftp`. Some examples follow:

- Get or put multiple files without prompting for each file using `mget` or `mput` and `prompt`:

```
multiple file transfer

ftp> prompt
Interactive mode off.
ftp> mget *.f
```

- Convert to lower case when transfer from MVS or DOS e.g. with `case`:

```
case conversion

ftp> case
Case mapping on.
ftp> get LOGON.CLIST
local: logon.clist remote: LOGON.CLIST
```

- Get rid of parts of the remote filename (e.g. version numbers on VMS) with `nmap` and rename parts of the filename (e.g. change extension from `.FOR` to `.f` with `ntrans`). The following example is useful for transferring Fortran files from a VAX to a UNIX system:

```
transfer from VAX

ftp> case
Case mapping on
ftp> nmap $1.$2;$3 $1.$2
ftp> ntrans .for .f
ftp> mget *.for
local: hplexaml.f remote: HPLEXAM1.FOR;14
```

rcp

The `rcp` (remote copy) command copies a file or directory from one host in the network to a directory or file on another host:

`rcp [-rp] source destination`

Remember the hosts have to be equivalent (see above)! The `-r` option means *source* is a directory and is to be copied with all the files and subdirectories it contains; the `-p` option preserves the modification time and access mode.

The *source* and *destination* specifications do not only contain the name of the file but also the host where it resides. You can specify *source* and *destination* in one of the following forms:

<i>filename</i>	relative or absolute name of local file
<i>host:filename</i>	absolute name of file residing on <i>host</i>
<i>user@host:filename</i>	name of file relative to the home directory of <i>user</i> on <i>host</i> .

Usage Notes:

You can use the `rcp` command to transfer files between hosts that are *not* equivalent if and only if the owner(s) of the remote file(s) give you their permission explicitly in a file named `.rhosts` in their home directory or if the remote account(s) do(es) not require a password.

Examples:

To copy the file `some.data` from the current directory on the local host to the directory `/u/hugo/archive` on host `rzri6f` enter:

```
rcp some.data rzri6f:/u/hugo/archive
```

Suppose you want to choose a different name for the *destination*:

```
rcp some.data rzri6f:/u/hugo/archive/x.y
```

You may specify the *destination* relative to hugo's home directory:

```
rcp some.data hugo@rzri6f:archive
```

This is absolutely equivalent to the first example.

This example is to be entered on one line:

```
rcp -rp hugo@rzhp9a:measurements/data  
rzhp9b:/exp.data/march
```

`rzhp9a` and `rzhp9b` are both remote nodes. The *source* is given relative to the home directory of hugo on `rzhp9a` whereas the *destination* is an absolute pathname on `rzhp9b`. The option `-rp`: *source* is a directory and is to be copied recursively (`-r`); this implies the *destination* also has to be a directory. The file permission and the modification time are preserved (`-p`).

4.6 IBM Mainframe Access

To access mainframe applications which require fullscreen support (e.g. ISPF on the MVS system), the terminal client software on the UNIX system must provide the IBM 3270 terminal emulation.

4.6.1 X-Window Users

For users of X-Windows at GSI, a shell script is provided which allows easy access to MVS applications. To start the session enter

```
xt mvsg for a graphics session  
or
```

```
xt mvs for an alphanumeric session
```

The `xt` script (see also chapter 4.5.1) uses the `x3270` program, an X-Window based 3270 emulator. It provides 3270 terminal sessions for UNIX workstations, accessing an IBM mainframe. GDDM type graphics is supported when specified by the corresponding options at session initiation. `x3270` is installed on the IBM AIX cluster systems, but can be used via the `xt` shell script also from HP and ULTRIX cluster machines. For extended information on `x3270` options and parameters type

man x3270

on one of the central maintained AIX machines.

4.6.2 Other X Windows environments

GDDMXD is an interface between the MVS GDDM graphics system and workstations, supporting the X-Window system. The GDDM data stream created by the MVS application (e.g. SATAN, GNOM, DCF) is translated to the X-Window protocol and transmitted by TCP/IP to the X-Window server. GDDMXD can be used from any workstation or PC with X-Window support without the need to login to an UNIX account at a GSI workstation and without the need to take care for required X11 fonts.

To initiate the GDDMXD output, you have to enter the TSO command (on IBM MVS system)

GDDMXD ON

outside ISPF. The TCP/IP address of the target display is read from the MVS dataset `user_id.XWINDOWS.DISPLAY`. Options for the graphic window on the workstation or PC as screen size, geometry, color table etc. can be modified by means of the dataset `user_id.X.DEFAULTS`. During the session, graphics output must be acknowledged on the X screen by hitting any PF-key, otherwise the host application will not continue. For more information contact the GSI user help desk.

4.6.3 Alphanumeric sessions

The `telnet` command on the IBM RISC System/6000 workstations supports the emulation of the DEC VT100 terminal or IBM 3270 terminal. Public domain `tn3270` software is available to emulate IBM 3270. The NCSA-Telnet for DOS PC's includes the Tektronix graphics emulation. To start the telnet 3270 emulation on AIX, enter

```
telnet -e 3270 mvs  
or  
tn3270 mvs
```

which connects directly to the MVS system in IBM 3270 mode. The `telnet` command supports some standard 3270 terminal types with different lines and columns. The maximum screen size will be used, depending on the lines/columns values of the screen or window from which the session is initiated. The 3270 keyboard mapping is determined:

`$HOME/.3270keys` specifies the user's keyboard mapping and overrides the system defaults.

`/etc/3270keys.hft` system default keyboard mapping for use with standard workstations or consoles.

`/etc/map3270` system default keyboard mapping for use with limited function terminals.

On a color display, e.g. X11 screen, the colors and field attributes are similar to the IBM 3279 terminal.

4.7 Backup and Archive Services

Backup of disks is an important central service. It allows recovery from disasters like disk failures, head crashes, etc. as well as from unintended deletion of single files or directories.

Archiving files is similar to backing them up, in that both operations store copies of files from a workstation to central storage. However, the underlying reasons for the two operations are different. Backing up files is done to have replacement copies available if the originals are damaged or lost. Archiving files is done for two reasons:

- Long-term storage of files no longer wanted on workstation storage.
- Creation of a "Snapshot" of a particular file or group of files as they exist at a certain point of time.

4.7.1 Automatic Backup with DSM

Backup and Archive functions for the UNIX user are provided with the Distributed Storage Manager (DSM) software.

DSM clients on AIX and HP-UX workstations and a DSM server on MVS mainframe are implemented at GSI.

Automatic daily backups of user disks on AIX, HP-UX and ULTRIX are done by the system, additional backups and archives, restore of backup versions, and retrieve of archive copies can be selected by means of a graphical user interface.

There is a maximum number of three backup versions for one file. The most recent backup is called the active version, the older ones are called inactive versions. Backups of deleted files and inactive backup versions are kept for 120 days.

system disks	<ul style="list-style-type: none"> • after major updates and changes
user disks /u/...	<ul style="list-style-type: none"> • daily incremental backup
data disks /d/...	<ul style="list-style-type: none"> • no backup
scratch disks /tmp, /s/...	<ul style="list-style-type: none"> • no backup • automatic removal of files after 3 days

4.7.2 Starting and Ending DSM

To start enter the command

```
dsm
```

Any of the following options may be specified with the dsm command:

- **DOMAIN** specifies selection of the filesystem for backup/archive in the DSM window. For example:
`dsm -domain=/nfs/clri6b/urz_1` preselects a filesystem in the DSM window.
- **NODENAME** specifies selection of the UNIX client node for which DSM services are wanted. For example to restore or retrieve files on AIX from HP client enter
`dsm -nodename=rzhp9a.`
 On two nodes are DSM-clients installed:
 CLRI6B and RZHP9A.

On the appearing DSM window are three main sections:

Action bar Contains buttons for seven menus: Backup, Restore, Archive, Retrieve, Utilities, View and Help.

File systems for Backup/Archive Using the mouse you can select and deselect individual file systems to backup or archive.

File systems for Restore/Retrieve Using the mouse you can select and deselect individual file systems to restore or retrieve.

To end the DSM session select *Exit* from the *Backup* menu.

4.7.3 File specifications

On the backup, archive, restore or retrieve selection you may specify files by file name, Wildcards. "?" replaces a single character, "*" replaces all characters. The file specification has to obey the following rules:

- If the file specification contains directory names, it must contain the complete path starting at the root directory.
- The file specification must end with a file name, not a directory name.
- Only the file name portion of the file specification may contain wildcard characters such as the asterisk.

Examples of valid entries are:

- myfile
- chapter?.doc
- /nfs/clri6b/u_rz_1/matthias/*

Entries which consist only of a file name (the first two examples) are looked up in the file system selected in the DSM window. Entries which contain a complete pathname are searched for in the directory specified. This does not have to be in the file system selected in the ADSTAR Distributed Storage Manager window.

4.7.4 Backing Up Files

Daily backup of user filesystems is done automatically at night, there is no need for additional backups.

dsm offers three variations of backup services:

1. *Incremental backup:*
Backup of all user files in the file systems specified in the DSM window which were created or changed since last backup.
2. *Backup by file specification . . . :*
Backup of a particular file or a group of files whose names match a specified pattern.
3. *Backup by directory tree . . . :*
Backup of an assembled group of files from a selection of one or more directories.

4.7.5 Archiving Files

Archiving files can be done in two ways:

1. *Archive by file specification . . . :*
Archive of a particular file or a group of files whose names match a specified pattern.
2. *Archive by directory tree . . . :*
Archive of an assembled group of files from a selection of one or more directories.

4.7.6 Restoring Backup Versions

There are three ways to specify the files to restore:

1. *Restore by file specification . . . :*
Select a particular file by name or select a group of files whose names match a specified pattern.
2. *Restore by directory tree . . . :*
Select one or more directories and assemble a group of files to restore from those directories.
3. *Restore subdirectory path . . . :*
Specify a particular directory and restore all the files in that directory.
You can choose to restore the files to a directory with the same name as the source or to one with a different name.

In an **Action for files that already exist** section of the Restore Subdirectory Path window are three choices to select for files that already exist:

- Show a prompt.
- Leave the existing file intact.
- Replace the existing file.

Inactive versions of files are only selected if in the **View** Menu is specified to show active and inactive files. Otherwise only active versions are shown.

4.7.7 Retrieving and Deleting Archived Files

Retrieving files is done by specification of

- file name;
- file description given when files were archived;
- archive date;
- expiration date;

An archived file must meet all four criteria in order to be retrieved.

The Retrieve menu has a second option for deleting archived files. The same criteria as above are used to do a first selection of files. A further selection is done by mouse clicks in the file selection list.

4.7.8 Using the Utilities Menu Options

The Utilities menu offers six options. Two of them – Change password and Delete restore/retrieve file systems – can only be used by a root user.

The other options offer

- restoring or retrieving another user's files.
- displaying and updating access privileges.
- displaying server and client options.
- displaying policy information.

4.7.9 Using the View Menu Options

The View Menu controls:

- showing only active or all backup versions.
- order in which files are displayed. Sorting is can be done by:
 - directory;
 - file name;
 - file extension;
 - size;
 - modification date.
- file attributes displayed in addition to the file name. Possible are:
 - User;
 - Group;
 - Management Class;
 - Size;
 - Modification Date;

4.7.10 Displaying Online Help

Online help is available in two ways. The click on the Help button which is available in many windows displays online information about the current operation.

The selection of Help in the action bar shows another menu. In the task list all kind of possible actions are explained.

4.8 Batch Jobs on the RS/6000 Cluster

LoadLeveler is the facility for executing batch jobs on the GSI RS/6000 cluster and the SP2 system. Jobs are submitted using the `llsubmit` command, or via the graphical user interface `xloadl`. As part of the submittal, the job owner constructs a job command file, where he specifies the characteristics of the job and the resource it requires to execute.

He has to specify one of the following job classes:

class	cpu time	limit	memory limit
short	1 hour	120 MB	
medium	6 hours	120 MB	
long	24 hours	120 MB	
large	24 hours	120 MB	

Important LoadLeveler commands are:

<code>llsubmit</code> :	submits jobs to the LoadLeveler
<code>llq</code> :	examines the LoadLeveler queue
<code>llcancel</code> :	cancels jobs from the LoadLeveler queue
<code>llstatus</code> :	examines the status of machines in the LoadLeveler pool
<code>xloadl</code> :	start the GUI.

The job command file may be structured in the form of a shell script with the following rules:

- Each line that has a # as its first character and a as its second non-whitespace character is interpreted as part of the job description.
- If you omit the 'executable' keyword, the shell script itself is used as the executable. In this case it must be marked as executable (`chmod +x`).

Example of command files:

```

job1.cmd

# Simple job command file
#
# @ executable = xyz
# @ queue

```

```

job2.cmd2

# Example: specify a job class
#
# @ executable = xyz
# @ class = medium
# @ queue

```

```

job3.cmd
# Example: executable shell script
#
# @ class = long
# @ queue
xyz

```

To submit these jobs use:

```

$ llsubmit job1.cmd
$ llsubmit job2.cmd
$ chmod +x job3.cmd
$ llsubmit job3.cmd

```

The screenshot shows the IBM LoadLeveler GUI with three main panels:

- Jobs:** A table listing job details.

Id	Owner	Submitted	ST	PRI	Size	Class	Running	On
clri6a.3575.0	bass	1/24 09:29	R	50	0.0	long	clri6c	
clri6h.91.0	hartnack	1/24 11:16	R	50	0.0	long	clri6d	
clri6h.92.0	hartnack	1/24 11:17	R	50	0.0	long	clri6d	
clri6h.93.0	hartnack	1/24 11:17	R	50	0.0	long	clri6e	
clri6a.3571.0	bass	1/23 17:33	R	50	0.0	long	clri6f	
clri6e.221.0	spieles	1/23 22:51	R	50	0.0	long	clri6g	
clri6a.3574.0	bass	1/24 09:29	R	50	0.0	long	clri6h	
- Machines:** A table listing machine details.

Name	Run	Tot	State	LdAvg	Idle	Arch	OpSys
clri6a.gsi.de	6	6	Idle	2.12	0	R6000	ADX32
clri6b.gsi.de	0	0	Idle	2.26	0	R6000	ADX32
clri6c.gsi.de	0	0	Running	1.00	1462	R6000	ADX32
clri6d.gsi.de	0	0	Running	2.00	115	R6000	ADX32
clri6e.gsi.de	1	1	Running	1.00	9999	R6000	ADX32
clri6f.gsi.de	0	0	Running	2.05	0	R6000	ADX32
clri6g.gsi.de	0	0	Running	1.02	32	R6000	ADX32
- Messages:** A log of system messages.

```

1/20 13:42 Setting Job Status timeout to 120 seconds
1/20 13:42 Setting Host Status timeout to 60 seconds
1/20 15:03 File "xx.cmd" was written.
1/20 15:04 File "/u/peter/xx.cmd" was written.

```

Figure 4.1: LoadLeveler GUI xloadl

Figure 4.1 is the result of the `xloadl` command. You can click to the **File** button to get a window to create and to submit a job without knowing the details of the LoadLeveler command language. For more information see the man pages for: `LoadLeveler`, `llsubmit`, `llq`, `llstatus`, `llcancel` or Hypertext-Help of `xloadl`.

4.8.1 LoadLeveler using NQS Scripts

Scripts written for NQS that contain NQS options are acceptable to LoadLeveler. The options are mapped as closely as possible to the features provided by LoadLeveler.

Remarks:

- You cannot mix LL and NQS options in a script.
- If you do not specify a 'class' in your script it will run in the 'short' class.

```

NQS example:

#@ $-lt 00:01:10
#@ $-lm 20mb
#@ $-q long
#@ $
xyz

```

4.9 Tape Handling Tools

This section provides an overview on the tape utilities available in the Unix world of GSI. These are

- a **tape allocation system**, developed to enable normally privileged users to allocate (and free) the public available tape drives in the Unix world of GSI, and
- **tape copy stations** for 4mm DAT and 8mm Exabyte tapes, the latter containing a stacker for up to ten Exabyte volumes.
- Another tool, a client-server package, which allows Unix users to copy experiment data between MVS, Unix tape or Unix disk, is described in section 9.1.4 on page 9-2.

See also the documentation in the WWW on **Tape Handling in Unix at GSI**, which may be entered via the *Unix* page in the *Computing* home page.

4.9.1 Tape Allocation System

In native Unix systems, a normally privileged user has no possibility to lock his tape against accidental (mis)use by another user. Without additional security actions it cannot be excluded in principle that valuable data are unintentionally overwritten by other users. To overcome this situation, a suitable tape allocation system has been developed at GSI.

In order to get access to a tape device at a centrally managed Unix workstation at GSI, the tape **must** be allocated by the user with the command

```
tape alloc devicefile
```

If the required drive is allocated by another user, try later again. After successful allocation, no other user has access, until the tape device is freed again:

Node	devicefile	description
rzri6m	/dev/rmt0	TZ87 DEC CompacTape III, 10 GB/vol
rzri6m	/dev/rmt1	8mm Exabyte TTI 8510, 5 GB/vol
rzri6m	/dev/rmt2	8mm Exabyte EXB 10e, 5 GB/vol, Stacker with 10 slots
rzri6m	/dev/rmt3	4mm DAT DDS-2 compressing, 4 GB/vol
rzri6m	/dev/rmt4	4mm DAT DDS-2 compressing, 4 GB/vol
rzri6m	/dev/fd0	2.88Mbyte Diskette drive
rzhp9b	/dev/rmt/0m	4mm DAT HP6400-2000 non-compressing, 2 Gbyte/vol,

Table 4.1: List of devices at the central tape station.. All capacity numbers are valid for uncompressed data.

```
tape free devicefile
```

The allocation and free commands must be executed on the node with the required tape drive connected. On **any** node an overview of all centrally available tape drives and their status may be obtained:

```
tape [show]
```

The *show* option is the default option of the `tape` command. If you are working on a distributed Unix workstation, successful execution of the last command requires that your workstation is made equivalent to the nodes containing the tape drives. This can be done by a suitable `.rhosts` file in your home directory.

For more details look into the WWW or request the corresponding Unix man page for `tape` (AIX, HP-UX).

For a list of available tape drives, for the hosts they are connected to, and for the names of the special device files see table 4.9 on page 4-16.

The central tape station named **rzri6m** resides in the graphics room of the computing center (ground floor of the southern building) and is accessible 24 hours a day and 7 days a week. The node `rzhp9b` is located in the machine room next door and only accessible at operator times.

4.9.2 Tape Copy Station

The tape station `rzri6m` may serve as copy station for 4mm DAT DDS2 and for 8mm Exabyte tapes. With the (standard) AIX command `tcopy` complete DAT or Exabyte tape volumes can be copied with a single command. For example the command

```
tcopy \dev\rmt3 \dev\rmt4
```

duplicates complete 4mm tape volumes.

Both 4mm drives fully support the newer DDS-2 format but are also able to read from and write to the older DDS-1 volumes. However, DDS-2 volumes cannot be read with DDS-1 drives (such as `/dev/rmt/0m` on `rzhp9b`).

One of the two Exabyte drives is part of a **tape stacker** containing a cartridge holder with slots for up to ten Exabyte volumes. The slots are accessed by a small robot in sequential order from bottom to top via the special device file (`/dev/rmt2`). With the command

```
mt -t devicefile offl
```

the current Exabyte volume is unloaded and put back to the stacker slot where it previously resided. Then the robot checks the slot(s) above and puts the next Exabyte volume found into the drive.

Running a suitable shell script with a series of `tape copy` and `mt` commands, no more manual intervention is necessary if the stacker is once filled. For more details see section C on page C-1 (How to use the Tape Stacker).

The Exabyte copy station facilitates the creation of (multiple) tape copies, e.g. for experiment data distribution to several locations, or the staging of files from many different tape volumes to a local disk.

However, be patient when handling Exabyte tapes under Unix. Tape initialization is rather slow, and an unload takes more time than you can imagine! Even more worse, in our experience the reliability of the 8mm Exabyte drives is rather poor. Therefore, though Exabyte tapes from historical reasons are still the best supported and still most heavily used tapes at GSI, we recommend the usage of 4mm DAT or DEC TZxx CompacTapes.

At the tape station also a 2.88Mbyte Diskette drive (device file `/dev/fd0`) is available.

Chapter 5: Editors

Text editing is one of the important things on any computer system. On UNIX systems several text editors are available: Emacs, vi, ed, LPEX, softedit, INed, edt+, uni-Xedit, Vuepad, and xedit are only a few UNIX editors.

A versatile Unix user knows at least vi or Emacs, which will probably be available on every platform.

The text in this section is designed to get you started using the vi, Emacs or softedit editor quickly. For vi and Emacs the elementary editing will be explained. softedit is described in more detail.

Before you invoke an editor, you should know one possible pitfall: An editor must know what kind of terminal type you are using. If your terminal type is not defined or incorrect, you may get problems with the editor. Quit the edit session and set the correct terminal type (see 1.7.1 on page 1-8).

5.1 vi Editor

The editor vi is a standard full-screen text editor that is available on almost every UNIX system. On AIX, HP-UX and ULTRIX vi is part of the operating system. On all POSIX.2 (IEEE 1003.2) conform systems vi is available. This allows users to move from one POSIX system to another using the same editor on each system.

5.1.1 Operating Modes

The vi editor has three operating modes:

- vi command mode.
- text input (or insert) mode.
- ex command (or line edit) mode.

In the vi command mode, each key initiates an instruction. In the text input mode the keyboard works like a typewriter. In the ex command mode you can use the ex line editor commands.

All commands in vi are case-sensitive.

5.1.2 Starting vi

To start vi type `vi` followed by the name of the file you want to edit.

```
vi myfile
```

When it is a new file, the buffer is empty and the screen will look like:

```
~  
~  
~  
"myfile" [New file]
```

The tilde (~) down the left-hand column of the screen indicates that there is no text in the file, not even blank lines. The prompt line (also called status line) at the bottom of the screen echoes the name and status of the file.

5.1.3 Exiting vi

The vi command to exit and save edits is **ZZ**. You can also use ex command **:wq** or **:x** to exit and save the file. **:wq/x** is equivalent to **ZZ**. Unlike vi commands, the ex commands introduced by **' : '**, require a **(Return)** after the command. Without saving you can exit vi with the ex command **:q!**.

5.1.4 vi Command Mode

When you start vi you are in vi command mode, and the editor is waiting for you to enter a command. Commands enable you to move anywhere in the file, to perform changes, or to enter insert mode to add new text. Commands can also be given to exit the editing in order to return to the UNIX prompt.

One of the most used vi commands is **' i '**. The **i** does not appear on the screen but after you typed it, all characters will be displayed and entered into the buffer. The cursor marks the current insertion point. To tell vi you want to stop insert mode, press **(Esc)**. **(Esc)** moves the cursor back one space and returns vi to the vi command mode. If there is no **(Esc)** on your Keyboard, try **(Ctrl) - (|)** instead.

When you opened a new file and want to insert the words **this is a new file** type **this is a new file**.

```
this is a new file
```

will be displayed. To start a new line press **(Return)**.

If you do not know whether you are in the vi command mode or the text input mode press **(Esc)** to enter the vi command mode. A beep will confirm the command mode.

5.1.5 ex Command Mode

Type **Q** to invoke the ex command mode. At the command line (bottom) the prompt **':'** will be displayed. The command **vi** returns you back to the vi command mode.

In the vi command mode you can call one ex command and immediately return to the vi mode by prefacing a ex command with a **':'**.

5.1.6 Basic vi Keystrokes

Moving around

l or →	Move forward one character (right).
h or ←	Move backward one character (left).
k or ↑	Move to previous line (up).
j or ↓	Move to next line (down).
w	Move word forward .
b	Move word backward .
0	Move to begin of line.
Return or +	Move to begin of next line.
-	Move to begin of last line.
\$	Move to end of line.
Ctrl - f	Move forward one screen.
Ctrl - b	Move backward one screen.
G	Move to end of buffer.
:1	Move to begin of buffer.
:n	Move to line number n .
Ctrl - g	Display current line number.
Ctrl - I	Redraw screen.
/pattern	Search forward for pattern.
?pattern	Search backwards for pattern.
n	Repeat last search in same direction.
N	Repeat last search in opposite direction.

Inserting Text

i	Inserting text before cursor.
a	Inserting text after cursor.
I	Inserting text at begin of line.
A	Inserting text at end of line.
o	Open new line for text below cursor.
O	Open new line for text above cursor.

Deleting Text

x	Delete previous character.
X	Delete character under cursor.
dw	Delete the word the cursor is on.

D	Delete from cursor to end of line.
dd	Delete current line.
p	Put deleted text after cursor.
P	Put deleted text before cursor.
Yank ¹	
yw	Yank (copy) word.
yy	Yank (copy) current line.
“aay	Yank (copy) current line into named buffer a .
p	Put yanked text after cursor.
P	Put yanked text before cursor.
“aP	Put text from buffer a before cursor.

Undoing and other vi Commands

.	Repeat last edit command.
u	Undo last edit.
U	Restore current line.
J	Join two lines.

Exiting Commands

ZZ	Save (write) and quit file.
:x	Save (write) and quit file.
:wq	Save (write) and quit file.
:w	Save (write) file.
:w filename	Write current buffer to <i>filename</i> .
:q	Quit file.
Q	Quit vi and invoke ex.
:e file	Edit <i>file</i> without leaving vi.

Some ex Commands

:set	Display options set by user.
:set all	Display list of all current options, both default and those set by the user.
:set number	Display line numbers.
:set showmode	Displays in insert mode a message on the prompt line indicating the type of insert you are making.
:set option	Activate <i>option</i> .
:set option=value	Assign <i>value</i> to <i>option</i> .
:set option?	Display <i>value</i> of <i>option</i> .
:set nooption	Deactivate <i>option</i> .

¹ Yanking means copying text into a buffer

:sh	Invoke shell.
(Ctrl) - (d)	Return to editor from shell.
!command	Execute UNIX <i>command</i> .
:r newfile	Read contents of <i>newfile</i> into current file.
:r !command	Read output of UNIX <i>command</i> into current file.
:map key sequence	Assign <i>sequence</i> to <i>key</i> .

The following example explains the map command:

```
:map q :q!
```

Press `q` in `vi` command mode to change to `ex` command mode. `q!` will be displayed. Press `(Enter)` to exit from `vi` without saving changes.

ex Block Commands

When you want to use the `ex` block operation commands in `vi`, it is convenient to issue `:set number` first to display line numbers. The block you want to operate on is specified by the numbers of its first and last line, separated by a `,` (comma).

:A,Bd	Delete all lines from line number <i>A</i> to <i>B</i> inclusively.
:A,BmC	Move all lines from line number <i>A</i> to <i>B</i> after line number <i>C</i> .
:A,BcoC	Copy lines <i>A</i> to <i>B</i> after <i>C</i> .
:A,Bw file	Write lines <i>A</i> to <i>B</i> into the specified file.

Example:

```
:140,155co34
```

will copy 16 lines (from 140 to 155 inclusively) between the 34th and 35th line.

5.1.7 The .exrc File

Your own `vi` environment is controlled by the `.exrc` file in your home directory. A sample `.exrc` file:

```
set number
set showmode
map q :q!
```

The file is read by `ex` before it enters the `vi` mode, commands in `.exrc` should not have a preceding colon.

5.1.8 The EXINIT Variable

Instead of having an `.exrc` file you can alternatively set the `EXINIT` variable. With the Korn Shell the following setting would be exactly equivalent to the previous `.exrc` file:

```
export EXINIT='set number show-
mode|map q :q!'
```

5.1.9 More about vi

More about `vi` can be found in the Hypertext reader (`info`) and in the man page (`man vi`).

Recommended books on `vi` are 'Learning the `vi` Editor' from Linda Lamb [12] and 'The Ultimate guide to the `vi` and `ex` text editors' from the Hewlett-Packard Company [5].

5.2 GNU Emacs

GNU Emacs is a powerful editor in the UNIX world. Emacs belongs to the GNU project of the Free Software Foundation (look appendix D) and is available on nearly all UNIX platforms, like AIX, HP-UX and ULTRIX. We have it running on AIX and HP-UX.

Unlike most other editors, Emacs is a complete working environment. You can start Emacs in the morning, work all day and night, and never leave it. You can use it to edit, rename, delete, and copy files; to compile programs; to interactive work with the UNIX shell; and so on. Before windowing systems like X became popular, Emacs often served as a complete windowing environment.

5.2.1 Emacs Commands

Emacs commands consist of a modifier, such as `(Ctrl)` (CONTROL), `(Esc)` (ESCAPE), or `<META>` (META), followed by one or two characters. In this text the following notation is used to describe the keystrokes.

`(Ctrl) - (g)` Hold down the `(Ctrl)` key and press `(g)`.
`(Esc) (X)` Press `(Esc)`, release it, and then press `(X)`.

Most Emacs manuals refer to the `<META>` key instead of the `(Esc)` key. But most keyboards don't have a `<META>` key, so we will refer to `(Esc)`. If you have a `<META>` key, you will probably refer to use it instead of `(Esc)`. The `<META>` key works like the `(Ctrl)` key described above. `(Esc) (X)` is then equivalent to:

`<META-x>` Hold down the `<META>` key and press `(X)`.

To complete a command you may need to press a carriage return:

Return Press the **RETURN** key. This key may be labeled **ENTER** on your keyboard.

All Emacs commands, even the simplest ones, have a 'full name': for example **forward-word** is equivalent to the keystrokes **Esc** **f** and **forward-char** is equivalent to **Ctrl** - **f**. Many commands only have 'full names'; there are no corresponding keystrokes.

5.2.2 Starting Emacs

To start Emacs, simply type `emacs` followed by the name of the file you want to edit.

```
emacs myfile
```

5.2.3 Exiting Emacs

To exit emacs, type

Ctrl - **X** **Ctrl** - **C**

5.2.4 Emacs Screen

When you enter Emacs, you are in a workspace. A cursor marks the position in the file. You don't have to do anything special before you start typing.

Just above the bottom of the screen, Emacs prints information about what it is doing. This line is called the 'mode line' and may look like this:

```
--*-Emacs: myfile (Text Fill)---5%----
```

At the left edge of the mode line, you may see two asterisks (**). This means that whatever you're editing has been modified since the last time you saved it. If you haven't made any changes, the asterisks won't be there. Next, Emacs prints 'Emacs:' followed by the name of the buffer or file you are editing (myfile). In parentheses following this Emacs shows the major (Text mode) and minor mode (Fill mode). Following this Emacs prints where you are in the buffer or file (5%). If the entire file is visible on the screen, Emacs prints the word ALL.

5.2.5 Emacs modes

Emacs has various editing modes in which it behaves slightly differently. When you often want features like word wrap so you don't have to press **Return** at the end of the line. When you are programming, your code must be formatted. For writing, there's the **text mode** and for programming in C is the **C mode**.

Text mode and **C mode** are major modes. A buffer can be in only one major mode at a time; to exit a major mode, you have to enter another one.

Whenever you edit a file, Emacs attempts to put you into the correct major mode for what you are going to edit. If you are editing a file with the ending .c, it puts you in the C mode. If the file has the ending .tex, it puts you in the T_EX mode. If Emacs can't determine a special mode, it puts you in the fundamental mode, the most general of all modes.

You can also change the mode manual with the command:

Esc **X** **startup-command** **Return**.

The important major modes and there **startup-commands** are in the following table:

Mode	Description	Startup-command
Fundamental	The default mode; no special behavior.	fundamental-mode
Text	For writing text.	text-mode
Directory	For editing directory contents	direc-mode
Indented text	Indents all the text you type.	indented-text-mode
Picture	For creating simple drawings.	picture-mode
C	For writing C programs.	c-mode
FORTRAN	For writing FORTRAN programs.	fortran-mode
Emacs LISP	For writing Emacs LISP functions.	emacs-lisp-mode
LISP	For writing LISP programs.	lisp-mode
LISP interaction	For writing and evaluating LISP expressions.	lisp-interaction-mode
nroff	For formatting file for nroff.	nroff-mode
T_EX	For formatting file for T _E X.	tex-mode
L^AT_EX	For formatting file for L ^A T _E X.	latex-mode
Scribe	For formatting file for Scribe.	scribe-mode
Outline	For writing outlines.	

	outline-mode
View	For viewing files but not editing. view-file
Mail	For sending mail. mail
Read Mail	For reading mail. rmail

In addition to the major modes there are also minor modes. These define a practical aspect of Emacs and can be turned on and off within a major mode.

Abbrev	Allows you to use word abbreviations. abbrev-mode
Fill	Enable word wrap. auto-fill-mode
Overwrite	Replaces characters as you type instead of inserting them. overwrite-mode
Auto-save	Saves your file automatically. auto-save-mode

In your `.emacs` file, the startup file of Emacs, you can set your favorite modes turned on automatically every time you start Emacs.

5.2.6 Basic Emacs Keystrokes

Moving around

Ctrl - f	Move forward one character (right).
Ctrl - b	Move backward one character (left).
Ctrl - p	Move to previous line (up).
Ctrl - n	Move to next line (down).
Esc - f	Move word forward .
Esc - b	Move word backward .
Ctrl - a	Move to begin of line.
Ctrl - e	Move to end of line.
Ctrl - v	Move forward one screen.
Esc - v	Move backward one screen.
Esc - >	Move to end of buffer.
Esc - <	Move to begin of buffer.
Ctrl - I	Redraw screen with current line in the center.

Deleting Text

Del	Delete previous character.
Ctrl - d	Delete character under cursor.
Esc - Del	Delete previous word.
Esc - d	Delete the word the cursor is on.

Ctrl - k	Delete from cursor to end of line.
Ctrl - @ or Ctrl - Space	Mark the beginning (or end) of a region.
Ctrl - w	Delete region (area between mark and cursor).
Esc - w	Copy region into kill ring.
Ctrl - y	Restore what you have deleted.
Ctrl - x - x	Exchange point (cursor) and mark.

Stopping and Undoing Commands

Ctrl - g	Aboard current command.
Ctrl - u	Undo last edit (can be done repeatedly).

Search and Replace

Ctrl - s	Search forward.
Ctrl - r	Search backward.
Esc - %	Interactively replace a text string. Valid responses are:
Space	Replace this one, go to next.
< , >	Replace this one, don't move.
Del	Skip to next without replacing.
< ! >	Replace all remaining matches.
< ^ >	Back up to previous match.
Esc	Exit

Buffers

Ctrl - x - b	Select another buffer.
Ctrl - x - b	List all buffers.
Ctrl - x - k	Kill a buffer.

File-handling and exiting

Ctrl - x - f	Read a file into Emacs.
Ctrl - x - I	Insert file at cursor position.
Ctrl - x - s	Save file (or may hang terminal; use Ctrl - q to restart).
Ctrl - x - w	Write buffer contents to file.
Ctrl - z	Suspend Emacs (use <code>exit</code> or <code>fg</code> to restart).

Ctrl - **X** **Ctrl** - **C**
Exit Emacs.

Multiple Windows

Ctrl - **X** 1
Delete all other windows.

Ctrl - **X** 2
Split window in 2 vertically.

Ctrl - **X** 5
Split window in 2 horizontally.

Ctrl - **X** 0
Delete this window.

Ctrl - **X** o
Switch cursor to another window.

Tutorial and Getting Help

Ctrl - **h** **Ctrl** - **h** **Ctrl** - **h**
Menu of help options.

Ctrl - **h** t Starts Emacs tutorial.

5.2.7 More information about Emacs

There is a man page available with:

man emacs

A Postscript file of the reference manual is in the directory `/usr/local/doc/gnu` on the central servers (nearly 300 pages).

A very good book about Emacs is 'Learning GNU Emacs' from D. Cameron and B. Rosenblatt [2]. It covers the first basics and the more advanced features of Emacs.

5.3 *edt+*

edt+ is a screen-oriented text editing utility. This editor emulates the functions and facilities of Digital Equipment Corporation's editor *edt*. Users who are familiar with Digital Equipment Corporation computers should find that this utility eases moving between DEC and Unix based systems by providing a common text editing tool. *edt+* incorporates a powerful word processor, disaster recovery system, an extensive help facility, full feature programmable text processing and the GOLD-KEY style of editing.

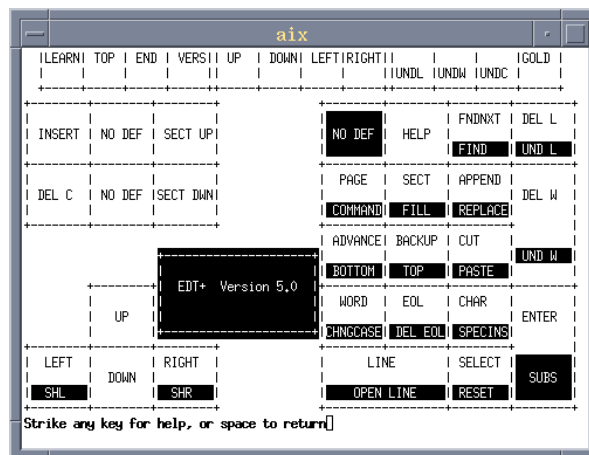
If *edt+* is installed on your system, you have to type `edt filename` to edit the file *filename*.

The *edt+* provides a full screen mode which is referenced as KEYPAD mode. In this mode functions are

attached to keys of the numeric keypad and function keys. Pressing the PF2 key gives you an on-screen help and the layout of the editing keypad. The PF1 key is the so-called GOLD key. Each key on the editing keypad is assigned two functions, one standard function and an alternate function. You reach the alternate function when you press the GOLD key first. Additional functions can be invoked as LINE commands. To type in LINE commands press the **PF1** **7** key sequence on the numeric keypad. Then an asterisk prompt appears in the lower left corner of your editing screen.

Numeric keypad functions

The layout for the numeric keypad functions looks like this:



PF3 **FNDNXT**
Search the next occurrence of the search string previously entered with the FIND key.

GOLD **PF3** **FIND**
Enter and search a string.

PF4 **DEL L**
Delete from cursor to the end of the line.

GOLD **PF4** **UND L**
Undelete with the last line deletion text.

7 **PAGE**
Move to the top of the page.

GOLD **7** **COMMAND**
Enter the LINE command mode

8 **SECT**
Move the cursor one section.

GOLD **8** **FILL**
Fill words in the selected range.

9	APPEND	Cut the selected range and append it to the paste buffer.
GOLD 9	REPLACE	Cut the selected range and place it in the paste buffer.
-	DEL W	Delete from cursor to the next word.
GOLD -	UND W	Undelete with the last word deletion text.
4	ADVANCE	Set the cursor direction to forward.
GOLD 4	BOTTOM	Move to the end of the buffer.
5	BACKUP	Set the cursor direction to backward.
GOLD 5	TOP	Move to the top of the buffer.
6	CUT	Move the selected text to the paste buffer.
GOLD 6	PASTE	Insert the contents of the paste buffer.
-	No Function	
1	WORD	Move the cursor one word.
GOLD 1	CHNGCASE	Change the case of characters.
2	EOL	Move the cursor to the end of line.
GOLD 7	DEL EOL	Delete up to the end of line.
3	CHAR	Move the cursor one character.
GOLD 3	SPECINS	Insert any ASCII character.
ENTER	ENTER	
GOLD ENTER	SUBS	Substitute the search string by the paste buffer contents.
0	LINE	Move the cursor one line.
GOLD 0	OPEN LINE	Insert a carriage return.
-	SELECT	Mark one end of a select range.
GOLD -	RESET	Reset and cancel internal buffers.

Note, the actions moving the cursor depend on the preselected direction.

You will find more information about `edt+` in the man pages (`man edt`).

5.4 `softedit` Editor

In the framework of `softbench` programming environment a powerful editor is available, which also can be used as general purpose editor (if you have a X-window display). Calling `softedit`, a Motif-style Window pops up. You can perform a variety of commands by clicking the command fields in the top-row of the window and their submenus. Most of the commands can also be executed by special key-strokes (accelerator keys).

If you place the mouse on any field and press the F1-key, a context-sensitive help menu will show up, giving detailed information on every item.

This section only covers the main aspects of the use of the `softedit` editor. For more details, please use the Online Help or the printed or online manual (`info`) **SoftBench Program Construction Tools, Users's Guide**, Chapter Using SoftBench Program Editor and SoftBench Edit Areas.

Starting a session

You can open a file either

- With the command


```
softedit filename .
```
- Click inside `softedit` or any other `softbench` tool the **File:Edit...** field of the command line (uppermost left field). A dialog box appears:
 - Type the filename in the input box or type * and select the **Show Match...** button for a list of files in your current directory. With the **Expand** button, filename completion can be done.

If you modify the contents of the file (if allowed, otherwise a **Read Only** button is shown) a **Save** button is activated in the top right corner, enabling you to save the modifications.

Multiple files can be edited easily using `softedit` by opening another file with one of the above methods. **Index** appears above the Edit Area when more than one buffer is in the Edit Area. When you select this button, you get an index of files being edited. Select a

file name to display that file in the Edit Area. An M next to the file names means the buffer has been modified. Cut and Paste is possible from and to different files in the Edit Area.

If you want to see two or more files in different windows simultaneously, select the **File:Transfer To New Stack** command from the command line or use the accelerator key `(Alt) - (↑) - (T)` to open a new editor window. You can transfer the file back to the working stack with **File:Transfer To Working Stack** or `(Alt) - (↑) - (W)`.

Moving around in the file

Fig. 5.1 shows a display of an editor window. You can scroll up and down by using the scroll bars with the mouse, `(PgUp)` and `(PgDn)` to scroll one page up or down or `(Ctrl) - (Home)` or `(Ctrl) - (End)` to jump at the beginning or end of the file.

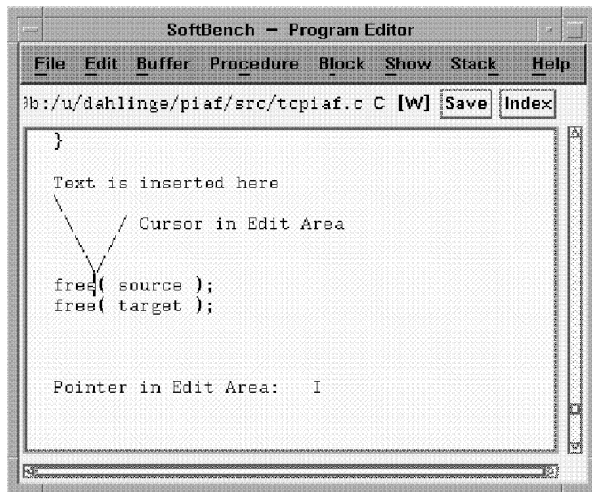


Figure 5.1: *Softedit* window with scroll bars, cursor and pointer

It is important to note the difference between **pointer** and **cursor** (see Fig. 5.1).

- The pointer moves as you move the mouse. It looks like a capital `I`.
- The cursor, a blinking vertical bar shows your current location for inserting text. It does not move, when you move the mouse. You can move the cursor with the commands or mouse actions from table 5.1.

Keystroke Mouse Action	Task Move Cursor
<code>(←)</code> , <code>(→)</code>	Forward, Backward one character
<code>(Ctrl) - (←)</code> ...	Forward one token (word)
<code>(Ctrl) - (T)</code> , <code>(Alt) - (T)</code>	Forward, backward one statement (sentence)
<code>(Ctrl) - (I)</code> , <code>(Alt) - (I)</code>	Beginning, End of block
<code>(Alt) - (P)</code> , <code>(Alt) - (N)</code>	Previous, Next procedure (paragraph)
<code>(Ctrl) - (A)</code> or <code>(Home)</code>	Beginning of line
<code>(Ctrl) - (E)</code> or <code>(End)</code>	End of line
<code>(↑)</code> , <code>(↓)</code>	previous, next line
<code>(Ctrl) - (PgUp)</code> ...	Top of page
<code>(PgUp)</code> , <code>(PgDn)</code>	Previous, next page
<code>(Ctrl) - (Home)</code>	Beginning of document
<code>(Ctrl) - (End)</code>	End of document
<code>(Ctrl) - (G)</code>	Goto line nr Brings up a dialog
<code>(Ctrl) - (Space bar)</code>	set a mark in the Edit area (place you wish to return to)
<code>(Ctrl) - (↑) - (Space bar)</code>	Return to where you set a mark
<code>(Ctrl) - (S)</code>	Locate a string. see below
press Mbl (left mouse button)	set cursor to pointer position

Table 5.1: Moving the cursor with *softedit*. Expressions in parentheses are valid for Text Mode.

Keystroke Mouse Action	Task Select Items
\uparrow - \rightarrow , . . .	Add character to the selection
\uparrow - \downarrow , . . .	Add line to the selection
drag with Mb1 pressed	select part of text
Double-click Mb1 or Ctrl - \uparrow - \downarrow	Select a token (word)
Triple-click Mb1 or Ctrl - \uparrow - \uparrow - \downarrow	Select a statement (sentence)
Quadruple-click Mb1 or Ctrl - \uparrow - \downarrow	Select block
Quintuple-click Mb1 or Ctrl - \uparrow - \uparrow - \downarrow	Select procedure (paragraph)
Ctrl - \uparrow	Select entire Edit Area
\uparrow click Mb1 or drag with Mb1 pressed	Extend the selection to include more characters

Table 5.2: Selecting items with `softedit`. Expressions in parentheses are valid for Text Mode.

Many commands depend on the edit mode. Currently available are edit mode **C** for `.c` files and **Text** for text and Fortran files. The current edit mode is displayed in the right upper corner of the edit area window.

For a full description, see the SoftBench Program Editor manual.

Selecting Items

Table 5.2 provides a list of keystrokes or mouse actions for selecting text. This selected text can then copied, deleted, etc. with the commands of the next section.

Cutting, Copying and Pasting Text

You can either cut or copy the selected area from an Edit Area (see previous paragraph) and paste it into another location of the same or a different Edit Area.

Table 5.3 shows the keystrokes for cutting, copying and pasting text.

There exist 3 buffers for storing selected text

- **Primary Selection** When you select text with Mb1 (highlight the text), it is automatically becomes the

Keystroke Mouse Action	Task Cut, Copy, Paste
\uparrow - Mb3 or \uparrow - Delete or Ctrl - W	Cut selected text and place it into the Clipboard and Cut_Buffer0.
Mb3 or Ctrl - Insert or Alt - W	Copy selected Text into the Clipboard and Cut_Buffer0
Mb2 or \uparrow - Insert or Alt - Y	Paste Contents of Clipboard
\uparrow - Mb2	Paste contents of Primary Selection
Ctrl - Mb2	Paste Contents of Cut_Buffer0

Table 5.3: Cut, copy or paste text with `softedit`.

Primary Selection. With this method you can paste text from another window, e.g. a terminal window.

- **Clipboard** Any text that is cut or copied is placed into the Clipboard. You can then paste the contents of the Clipboard into any other window.
- **Cut_Buffer0** Text that is cut or copied is also placed in the Cut_Buffer0.

Deleting and Killing Text

In addition to using the cut operation, you can also delete text and *not* place it in the Clipboard, or `killtext` and place it in a localized kill buffer (usable only within the current Edit Area).

Use the commands of table 5.4 to delete and kill text.

Finding and Replacing Text

After pressing Ctrl - S , a dialog windows opens with various options for searching and replacing text. Fig. 5.2 shows the dialog window.

Aligning Text

The system can automatically align C-source code and indent selected text portions by a fixed amount of spaces. Table 5.5 shows the keystrokes to perform alignment and indentation. You can customize the alignment and select various other options as the edit mode by choosing `Buffer:Settings...`

Keystroke Mouse Action	Task Delete, Kill
Delete or Ctrl - D	Delete next character
Backspace or Ctrl - H	Delete previous character
Alt - D	Delete from cursor to end of next token (word)
Alt - G	Delete from cursor to beginning of previous token (word)
Ctrl - K	Kill to end of line and place in local kill buffer
Ctrl - Y	Unkill (paste contents of kill buffer)
Select block of text, press Backspace	Delete block of text

Table 5.4: Delete and Kill text with softedit

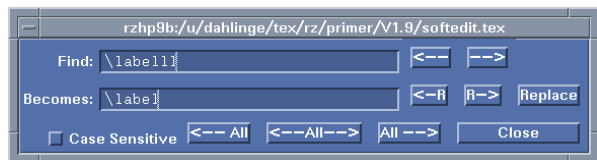


Figure 5.2: softedit search dialog window

Undoing Commands

As you edit, the system keeps track of the modifications (the depth of the undo stack is 50). To undo a modification press **Alt** - **Backspace** once or several times until the Edit Area is in the desired condition.

5.5 xedit

Xedit is a small editor that belongs to the X11 distribution. It was originally written as an example of using the Athena text widget. So all behaviors belong more to the text widget than to the xedit application (s. below)

The xedit editor could be started by typing:

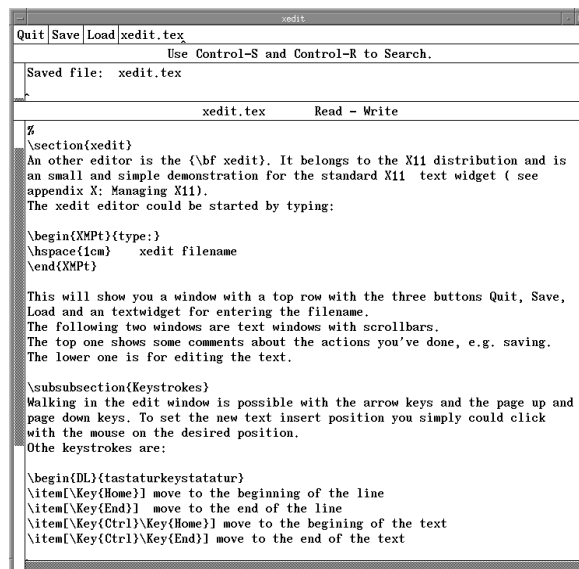
```

starting the xedit:

xedit filename
    
```

Keystroke Mouse Action	Task Align, Indent
Alt - A	Automatically align (format) selected part of text
Ctrl - R	Indent selected text by <i>n</i> spaces defined by Buffer:Settings...
Alt - R	Cancel Indentation of selected text

Table 5.5: Aligning text with softedit



This will show you a window with a top row with the three buttons Quit, Save, Load and an textwidget for entering the filename. The following two windows are text windows with scrollbars. The top one shows some comments about the actions you've done, e.g. saving. The lower one is for editing the text.

Keystrokes

Walking in the edit window is possible with the arrow keys and the page up and page down keys. To set the new text insert position you simply could click with the mouse on the desired position.

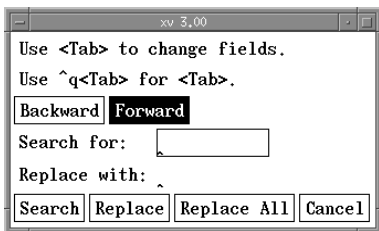
Other keystrokes are:

- Home** move to the beginning of the line
- End** move to the end of the line
- Ctrl** **Home** move to the beginning of the text
- Ctrl** **End** move to the end of the text
- Ctrl** **S** start search and replace with forward direction.

- Ctrl** **R** start search and replace with backward direction
- Ctrl** **k** delete from textposition to end of line
- Ctrl** **w** delete marked text
- Ctrl** **I** for inserting a file

Search & replace

After typing **Ctrl** **S** or **Ctrl** **R** a little window will pop up.



On top there is a toggle button who allows you to change the search direction. Then there are two textentries for specifying the search and the replace term. Then lower button row allows you to set the different types of searching: single **Search**, single search and **Replace**, and the global search and replace without asking by the **Replace All** button. With the **Cancel** the search could be stopped.

Cut & paste ...

Cut & paste uses the standard X11 cut & paste mechanism: a selection could be made with the left mouse button held down. By pressing the middle mouse button the selection will be pasted to the selected places.

Configuring the xedit

As standard X11 all application have an application default file. This could be found in the directory `/usr/lib/X11/app-defaults` or the directory specified by environment **XAPPLRESDIR** (in GSI Computing Environment: `/usr/local/lib/X11/app-defaults`).

Beneath the colors or the fonts, you can see, that the key bindings are as well overridable:

```
xedit*editWindow.Translations: #override\n\
<Key>Delete:          delete-next-
character()\n\
Ctrl<Key>End:          end-of-file()\n\
Ctrl<Key>Home:         beginning-of-file()\n\
Ctrl<Key>Right:        forward-word()\n\
Ctrl<Key>Left:         backward-word()\n\
<Key>Prior:           previous-page()\n\
...
```

On the left side the events are described. At the right side you can see several text widget buildin function calls. All buildin functions are described at the man page:

```
man TextActions
```

5.6 Other Editors

A lot of other Editors are available on UNIX systems.

5.6.1 ed Editor

The ed editor is a line editor that is available on all UNIX systems and accessible from any terminal. Like the Vi editor, the ed editor has command and input modes, but the ed editor allows you to move and copy only complete lines of data. The ed editor can be used to edit within a shell program or to edit very large files.

For more information about ed see the man page (`man ed`) or the Hypertext reader (`info`) on your UNIX system.

5.6.2 LPEX

LPEX, the Live Parsing Editor, is a programmable editor from the AIX SDE (Software Development Environment) WorkBench/6000 environment. It is based on the OS/2 LPEX editor. It can be called on AIX only by `softlpex`. On HP-UX see `softedit`, section 5.4

LPEX can be use to create and edit many kinds of data, including programs and documentation.

LPEX provides many capabilities for editing and manipulating documents. You can:

- Use multiple windows to display different documents or more than one view of the same document.
- Select a block of text and move or copy it within or between documents, or to a shell or to another application.
- Search for and change specific text or search for marks in a document.
- Control how often changes are automatically saved and recent changes can be undone.
- Change the appearance of a document by changing its fonts.
- Invoke a parser to use various techniques, such as color and indentation, to display the document.
- Perform many functions using either menus or keys on the keyboard.

LPEX has a menu bar at the top of the application window which contains pull-down menus to initiate commands. Mnemonic keys are also available for command invocation. In addition, LPEX commands can also be entered through the LPEX command dialog invoked from the Command menu pull-down.

LPEX is only available on AIX RS/6000 systems.

5.6.3 INed Editor

The INed editor is a full-screen AIX editor that allows you to do the following:

- Enter subcommands by command keys rather than on a command line.
- Edit multiple files in multiple windows on the screen.
- Scroll the screen horizontally (as well as vertically) through files.
- Move, copy, and delete blocks (as well as lines) of data.

The INed editor also includes:

- A file manager for the following:
 - Creating, changing, deleting, and recovering files and directories.
 - Moving among files and directories.
 - Moving and copying files and directories.
 - Restricting access to files and directories.
- Commands for working with structured files and their histories.
- Commands for limited text formatting.

The INed Editor is available only on AIX systems.

To edit the file *filename* start INed with:

e *filename*

More information can be obtained from the man page (man e) or the Hypertext reader (info).

5.6.4 uni-XEDIT editor

uni-XEDIT is a UNIX version of IBM's XEDIT editor, part of the VM/CMS mainframe operating system. It provides an easy to use full-screen editing function. It contains extensive features such as multiple file editing, prefix commands which visually perform line-oriented editing functions.

This editor is especially useful to users who are already familiar with IBM's mainframe text editor XEDIT or SPF-EDIT.

We have installed uni-Xedit on HP-UX.

To invoke uni-Xedit type :

xe *filename*

5.6.5 Vuepad

Vuepad is a text editor for use within the X Window System environment. Vuepad allows the use of the mouse for moving the edit cursor and for selecting portions of text for editing operations. Vuepad also supports the VUE drag and drop, allowing the user to drag file icons from the VUE file manager onto the vuepad window for editing.

The Vuepad editor is only available on HP-UX systems.

To start Vuepad type:

vuepad *filename*

For more information about Vuepad see the man page (man vuepad) or the Hypertext-reader (info) on the HP-UX system.

5.6.6 Notepad

The Notepad editor is a self explaining, very useful DEC window text editor. It allows you to use the mouse for moving and for selecting portions of the text.

Notepad is only available on ULTRIX.

To edit the file *filename* enter the command:

dxnotepad *filename*

For more information about editing with the Notepad editor double click on the help item inside the Notepad editor.

Chapter 6: Text processing

Text processing, word processing, text editing, and other phrases are often used to describe activities dealing with the input, editing and outputting of written forms of expressions. The important part of input of text into a file without further text formatting is described in chapter 5.

In this chapter, the text processing system $\text{T}_{\text{E}}\text{X}$ (and $\text{L}\text{A}\text{T}_{\text{E}}\text{X}$) will be discussed. Other text processing systems and the Unix-tools `awk` and `sed` to automate the editing, analyzing and processing of text will be described in a later release of this primer.

6.1 $\text{T}_{\text{E}}\text{X}$ and $\text{L}\text{A}\text{T}_{\text{E}}\text{X}$ text processing

$\text{T}_{\text{E}}\text{X}$ [8, 9] (pronounced *Tekh*) is a text processing system developed by Donald E. Knuth of Stanford University to compose by computer high quality documents, especially those containing many mathematical formulae. Its typographic quality is comparable to the finest works of the printing art. The $\text{T}_{\text{E}}\text{X}$ text formatting system offers several tens of custom-made fonts and over 300 basic and 600 “composed” `PLAIN` commands.

Unlike most desktop publishing systems, $\text{T}_{\text{E}}\text{X}$ does not attempt to show the appearance of the finished document on the screen as it is edited (although screen previewers are available). Instead, the document is “marked up” with codes, which indicate boldface, italics, special characters (e.g., “`\int`” for an integral sign), and the like. Large-scale aspects of design are automated; you just give the title of a chapter, and let the computer take care of numbering the chapter and putting the title in the right place on the page.

$\text{T}_{\text{E}}\text{X}$ is generally considered the most sophisticated computer typesetting system, as well as (for an experienced user) one of the easiest to use. $\text{L}\text{A}\text{T}_{\text{E}}\text{X}$ is a macro package build on top of $\text{T}_{\text{E}}\text{X}$.

Fundamental to $\text{L}\text{A}\text{T}_{\text{E}}\text{X}$ is the idea of a **document style** which determines exactly how a document will be formatted. $\text{L}\text{A}\text{T}_{\text{E}}\text{X}$ provides standard document styles that describe how standard logical elements should be printed. One may have to supplement these styles by specifying the formatting of logical structures peculiar to a given document, such as mathematical formulae. One can modify the standard document styles or create an entirely new one, though one should possess a basic understanding of typographical design before creating a radically new style.

6.1.1 Advantages of $\text{L}\text{A}\text{T}_{\text{E}}\text{X}$

- The layouts were developed by (American) design professionals, and they produce documents which indeed look as though “they were printed”.
- It is extremely simple to compose mathematical formulae.
- The user must only know a few easily memorized commands, which control the logical structure of the document, and (almost) does not have to know the technical details about how a document is formatted.
- More complex components such as footnotes, bibliography, table of contents etc. as well as simple pictures can be produced without great difficulty.
- $\text{L}\text{A}\text{T}_{\text{E}}\text{X}$ is the “de facto” standard for scientific publications in Europe and the U.S.A..

6.1.2 Disadvantages of $\text{L}\text{A}\text{T}_{\text{E}}\text{X}$

- The document can only be output on display units or printers with graphical (all point addressable) capabilities.
- One can vary a few parameters quite easily within the framework of a given $\text{L}\text{A}\text{T}_{\text{E}}\text{X}$ document layout style. However more basic deviations from the available layouts are only possible with a lot of (re)programming.

6.1.3 How does $\text{L}\text{A}\text{T}_{\text{E}}\text{X}$ work?

A $\text{T}_{\text{E}}\text{X}$ ($\text{L}\text{A}\text{T}_{\text{E}}\text{X}$) input file is a “plain” text file prepared with a text editor and in general has the extension `tex`.

The files describing the document styles or options (extension `sty`) are stored in a standard public directory (`/usr/local/lib/tex/inputs`). There exist many $\text{L}\text{A}\text{T}_{\text{E}}\text{X}$ styles, but only four standard ones, namely, `report`, `article`, `book` and `letter`. These styles can be further customized by specifying one or more options.

The output from $\text{L}\text{A}\text{T}_{\text{E}}\text{X}$ is a set of files. One of them (extension `dvi`) contains a device independent binary representation of the formatted text. This file can be converted into a printable form by a separate program.

A logfile of the (La) $\text{T}_{\text{E}}\text{X}$ run is generated (extension `log`); it contains information which also appeared on the screen (e.g. the names of the files read, the numbers of the pages processed, error messages etc).

The other files contain information about cross-referencing (extension `aux`), table of contents (extension `toc`), list of figures (extension `lof`) and list of tables (extension `lot`). They are used in a subsequent \LaTeX run to produce particular elements of the document.

A `idx`-file contains all indexed items. They can be sorted and included as an `ind`-file. To produce this sorted index, the program `makeindex` is provided.

A `bb1`-file contains the bibliographic references extracted by $\BIB\TeX$ from the bibliography database.

Figure 6.1 summarizes the dependencies of the different files used by \LaTeX .

6.1.4 \TeX glossary

This section is partly based on a glossary posted to \TeX -HAX by Jacques Goldberg of the Department of Physics, Technion–Israel Institute of Technology, Haifa (Israel).

glyph	A graphical representation of a character, for example a , <i>a</i> and \mathfrak{a} are three different glyphs for the same character.
font	A set of glyphs corresponding to a character set.
primitives	About three hundred basic commands wired into the \TeX program.
macros	More sophisticated or simpler to use commands, built upon the primitives.
Plain \TeX	Donald Knuth's \TeX augmented by his own set of macros.
\LaTeX	Donald Knuth's \TeX augmented by Leslie Lamport's alternative set of macros.
$\mathcal{A}\mathcal{M}\mathcal{S}\text{-}\TeX$	Donald Knuth's \TeX plus American Mathematical Society macros. The functionality of this package is now available to \LaTeX users with the <code>amstex</code> style option.
format	A given set of macros, compiled and written into a file of filetype <code>fmt</code> once to save repeated compilation time. \TeX will restore all definitions by reading such a format file.
TFM	Font Metric files contain <i>device independent global</i> information about the space on the output requested for each glyph in the font, as well as other boundary conditions such as relative spacing with neighboring glyphs (e.g. ligatures). \TeX does <i>not</i> need to know the detailed shape of a glyph. Filetype is <code>tfm</code> .
DVI	A DeVice Independent file (extension <code>dvi</code>) is output by \TeX . DVI files use the ASCII

255-character set and can be copied between different computers (PC's and mainframes) as *binary (image)* files. They contain the description of the formatted document.

DVI driver A program which turns the human-unreadable DVI file into ink on paper or light on a display. The DVI driver *must* know exactly how to produce the image of a given glyph. Therefore it needs a *raster* image for each glyph, and this raster is device (printer, display) dependent, at least, but not only, because devices have different resolutions.

previewer DVI driver to output the DVI file onto a screen.

font file A set of raster representations for each glyph in a font, clearly printer and magnification-dependent. These files have various filetypes such as `pxl`, `pk`, `gf`. The differences in names are historical and technical. `pxl` is obsolete but still used by some old drivers, `pk` is by far the most efficient especially when it comes to saving disk space, `gf` is produced on the way to making fonts by the METAFONT program.

DVI drivers are *not* supposed to be able to magnify fonts, but they should not generate a fatal error when a font is not available. DVI drivers expect to find the magnified rasters in a specific directory structure, and the user is responsible for not invoking magnifications for which no raster files exist; again \TeX does *not* check that the driver will or will not be able to print out a font at the requested magnification. This is *not* \TeX 's business.

An exception is the `dvips` driver. If it finds a missing font, it tries to calculate the font in the required magnification from the METAFONT input files (extension `mf`). They are then stored in the `/usr/local/Localfonts` directory.

magnification A scale factor applied to the original design size for a font. The following magnification factors are accepted:

$$0.5, 0.6, \dots, 1.0, \sqrt{1.2}, 1.2, (1.2)^2, \text{etc}$$

Not all fonts at each magnification will be present on all machines.

METAFONT A companion program to \TeX used to generate new fonts, or existing fonts at new magnifications. The program METAFONT reads

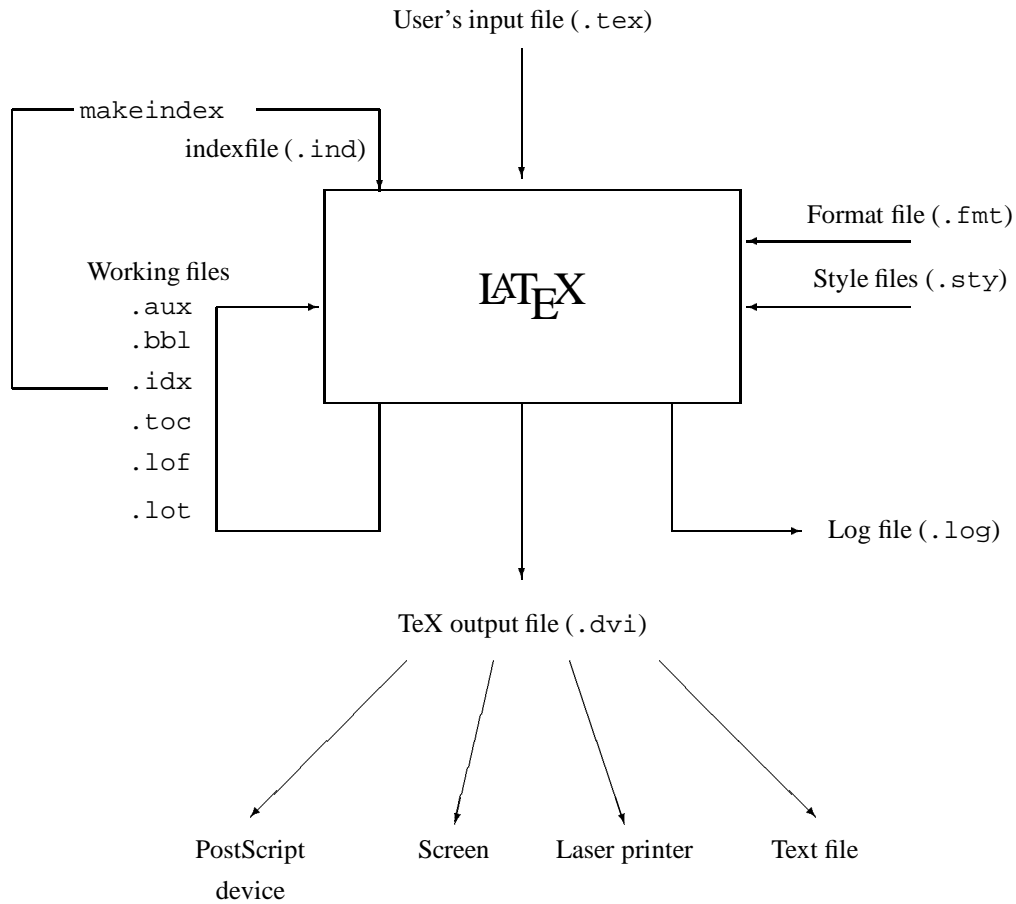


Figure 6.1: Data flow for the files used by L^AT_EX

a font definition file, actual device parameters, magnification etc., to generate *two* output files per input font definition. The input is an `mf` file and the two outputs are the corresponding `tfm` and `gf` files. If you wish to make a font at 5 different magnifications you have to run METAFONT 5 times, on almost the same `mf` file. If you wish to create the font for 3 different devices (a dot matrix printer, a laser printer, a display) you again must run METAFONT 3 times, with the correct printer definition.

All these runs will always produce the same `tfm` file but a different `gf` file.

L^AT_EX style A file which contains the description of or changes to a document layout style (extension `sty`). These are the files which are specified in the L^AT_EX command

```
\documentstyle[minor_style]
{major_style}
```

where `minor_style` is an optional part, which specifies small changes to be made to the mandatory `major_style`.

6.1.5 Documentation in PostScript form

All documentation related to T_EX, L^AT_EX, `stys`, drivers etc. are located in the `/usr/local/doc/tex` directory. You can `cd` to there and have a look on the various documents and print them with the usual `lp`, `lpr` commands. Often you will find documentation on `sty` files also in the `/usr/local/lib/tex/inputs` directory.

6.1.6 Suggested Reading

In order to exploit the power of T_EX and L^AT_EX you should consult one of the many books available. Below is given a short selection of the literature, as well as some articles in T_EX or PostScript form:

- *The T_EXbook* by D. Knuth, Addison Wesley [8]
- *L^AT_EX, a Document Preparation System* by L. Lamport, Addison Wesley [11]
- *L^AT_EX, Eine Einführung* by H. Kopka
- *Kompaktführer L^AT_EX* by R. Wonneberger
- *L^AT_EX Kurzbeschreibung* by H. Partl, `/usr/local/doc/tex/lkurz.ps`

6.1.7 Running L^AT_EX

The latest versions of T_EX, METAFONT and the L^AT_EX macro package have been installed on all centrally supported HP and IBM UNIX workstations at GSI. Below we show the basic commands to format a L^AT_EX document, preview the document on a graphics screen (X11), transform the `dvi` into PostScript and print the output. Then each major system will be reviewed in turn.

Running a Sample File

Before preparing your own documents, you may want to get acquainted with L^AT_EX by running it on a sample input file. You can copy the file `sample.tex`, which is present in the `/usr/local/lib/tex/inputs` directory.

You can now modify your copy of the file `sample.tex` if you want and then run L^AT_EX on your file by typing:

```
latex sample
```

When L^AT_EX has finished, it will have produced the file `sample.dvi` in your current (working) directory. You can view this file on your X-screen by the command

```
xdvi sample.dvi
```

In general the extension `.dvi` can be omitted.

You can print this file by first transforming the information into (file extension `ps`) PostScript using the command `dvips`.

```
dvips sample.dvi
```

In general the extension `.dvi` can be omitted.

At GSI, `dvips` is configured so that it produces a file `sample.ps`. You can either view this PostScript file on your X-screen with the command

```
ghostview sample.ps
```

or print it on a PostScript laserprinter:

```
lp sample.ps           -- on HP
lpr sample.ps         -- on AIX,ULTRIX
```

To specify a printer or use other options for that specific printer, see chapter 4.4 on page 4-6.

After your output has been printed, you can delete `sample.dvi` and possibly `sample.ps`.

Going into more details

The `tex`, `latex` and other relevant commands reside in the `/usr/local/bin` directory. When you get a `latex: not found` error from the system, first check whether the above directory is in your command search path by typing `echo $PATH`. If it is, please type `ls /usr/local/bin/latex` to check whether the `latex` file is indeed installed. If it is not, contact your system administrator.

Once you get \LaTeX started, you should get the message:

```
This is TeX, C Version 3.14t3
```

If the command `latex` is found, but you do not get this far, then you have probably the wrong version of the \LaTeX format. In this case you should contact your system administrator. Please remind that the \LaTeX version installed on the Unix workstations is newer than on the IBM MVS and VAX VMS systems and uses the New Font Selection Scheme (NFSS) developed by Frank Mittelbach and Rainer Schöpf [13]

Interrupting \TeX and respond to ? messages

When \TeX is running you can interrupt it by typing `(Ctrl)-(C)`. This will stop \TeX as if it had encountered an ordinary error, and you can then return to Unix shell command level by typing `_`, as described in the \TeX book.

If \TeX stops with an error message and a question mark (?), you have several possibilities to proceed:

<code>_</code>	stop the execution of \TeX
<code>\h</code>	ask \TeX for a more detailed information about the error
<code>-</code>	(just press <code>(Enter)</code>): continue and hope that \TeX can go ahead in spite of the error encountered
<code>\e</code>	call the editor of your <code>.tex</code> file and jump to the line in error
<code>\r</code>	run without stopping
<code>\s</code>	scroll future error messages
<code>\q</code>	run quietly
<code>\i</code>	to insert something, e.g. to correct a misspelled control sequence

A “deadlock” may occur if \LaTeX requires a file which is not accessible. The user will be prompted to input the correct file name, if any. If you want to exit from \TeX at that point, specify `null` as filename. This is an empty file, which is provided in the standard input file directories.

You should be aware that \TeX only searches its own input directories. By defaults, these are the system inputs directory (`/usr/local/lib/tex/inputs`) and the current working directory. You can specify other directories by relative or absolute pathnames. The `~` character for the home directory is not recognized.

dvips - DVI converter to PostScript

The DVI driver `dvips` translates a DVI file into a PostScript file. The latter can be printed on any PostScript printer, usually the HP Laserjet III Si or a DEC PostScript printer.

The syntax for the `dvips` command with a selection of important options is:

```
This is dvips 5.482 Copyright 1986-92
      Radical Eye Software
Usage: dvips [options] filename[.dvi]
l # Last page
n # Maximum number of pages
o f Output file
p # First page
q* Run quietly
r* Reverse order of pages
c # Uncollated copies
C # Collated copies
t s Papersize (s=landscape,portrait,a3,a4)
E* Try to create EPSF

# = number   f = file   s = string
* = suffix, '0' to turn off
c = comma-separated dimension pair
    (e.g., 3.2in,-32.1cm)
```

Many other options can be found in the man pages, in the file `/usr/local/lib/tex/doc/dvips.ps` or by `dvips -?`.

The DVI file may in general be specified without the `.dvi` extension. Fonts used may either be resident in the printer or defined as bitmaps in `pk` files, or a “virtual” combination of both. `dvips` will automatically invoke METAFONT to generate fonts that don’t already exist.

If you want to print your file, use the usual printer commands:

```
lp [-d myprinter] sample.ps      -- on HP
lpr [-P myprinter] sample.ps    -- on AIX,
                                ULTRIX
```

For a description of the other options of the `lp`, `lpr` command and possible printers see chapter 4.4 on page 4-6.

You can direct the output from `dvips` directly to the printer without writing to the intermediate PostScript file:

```
dvips -o!lp sample.dvi          -- on HP
dvips -o!lpr sample.dvi        -- on AIX,ULTRIX
```


xdvi – DVI previewer for X-windows

The DVI previewer `xdvi` is a program which runs under the X window system. It has the capability of showing the file shrunken by various (integer) factors, and also has a “magnifying glass” which allows one to see a small part of the unshrunk image momentarily.

In addition to using keystrokes to move within the file, `xdvi` provides buttons on the right side of the window, which are synonymous with various sequences of keystrokes.

```
xdvi [+ [<page>]] [-s <shrink>]
[-paper <papertype>] [-mgs[n] <size>]
[-fg <color>] [-bg <color>] [-hl <color>]
[-bd <color>] [-cr <color>] [-bw <width>]
[-geometry <geometry>]
[-iconic] [-display <host:display>]
dvi_file
```

The many other options of `xdvi` are described in the man page or by typing `xdvi`.

gs and ghostview – a PostScript previewer

The public domain Ghostscript previewer `gs` is available on Unix workstations. It is part of the GNU project. Ghostscript is a programming language similar to Adobe Systems’ PostScript (tm) language. `gs` reads a file and displays it as a Ghostscript file. It then interprets commands from standard input until an end-of-file character (`(Ctrl) - (D)`) is typed. The ‘quit’ character (`(Ctrl) - (C)`) also terminates Ghostscript execution.

To invoke the interpreter, give the command

```
gs <filename1> ... <filenameN>
```

The interpreter will read in the files in sequence and execute them. After doing this, it reads further input from the primary input stream (normally the keyboard). Each line (i.e. characters up to a `<return>`) is interpreted separately. To exit from the interpreter, type `quit<return>`. The interpreter also exits gracefully if it encounters end-of-file.

The program `ghostview` provides a menu/mouse interface for the `ghostscript` previewer. To invoke this interface, give the command

```
ghostview <filename>
```

You can then easily browse through your document, select specific pages, print the whole document or only selected pages, see the document with all PostScript figures included and zoom it.

6.1.8 Using PostScript fonts

Thanks to the NFSS (new font selection scheme) by Schöpf and Mittelbach [13] it is easy to replace the computer modern fonts provided with \TeX by the arbitrarily scalable original Adobe PostScript fonts. This task has been simplified by the use of some document-style sub-options to be included in the `\documentstyle` command. The following style options and new commands are available:

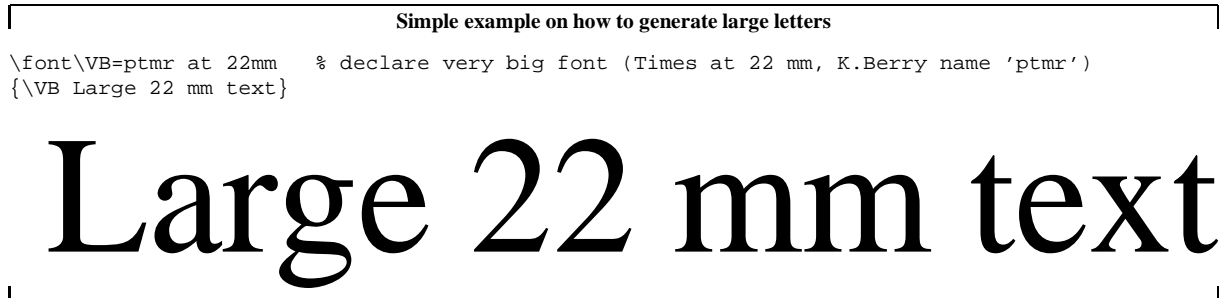
<code>avant</code>	makes AvantGarde default text font, and Times default sans font.
<code>bookman</code>	makes Bookman default text font, and Helvetica default sans font.
<code>dingbat</code>	sets up various commands which use the Zapf Dingbats font, each of which take as a parameter the character number of a symbol in the font. Possibilities are: <ul style="list-style-type: none"> <code>\ding</code> Just prints the character <code>\dingfill</code> wherever you use a filler, fills the space with the selected character <code>\dingline</code> a freestanding line of symbols <code>\dinglist</code> a list environment which tags items with the selected symbol
<code>epsfig</code>	a style file for including Encapsulated PostScript figures. See section 6.1.9 below for a description.
<code>helv</code>	makes Helvetica default text font, and Times default sans font.
<code>ncs</code>	makes NewCenturySchoolbook default text font, and AvantGarde default sans font.
<code>palatino</code>	makes Palatino default text font, and Gill-Sans default sans font.
<code>times</code>	makes Times default text font, and Helvetica default sans font. This document has been produced with this style option.

For a more complete description of the NFSS together with PostScript fonts, refer to the article `/usr/local/doc/tex/psnfss.dvi`.

Please remind that the dvi-driver has to be able to handle PostScript fonts. At the moment, `xdvi` is *not* able to deal with these fonts. Instead you have to use `dvips` and `ghostview`!

Playing with PostScript

\LaTeX offers only a certain number of font sizes (`\Large`, `\LARGE`, `\Huge`). When for a particular

Figure 6.2: Large Text with L^AT_EX and PostScript

purpose a non-available type size or a non-preloaded font is needed, then the basic font commands of T_EX can be invoked and the `dvips` driver will then include the font at the desired size. In this case K.Berry's short file name as found in the NFSS font description [13] should be specified. For more details the reader should consult the `dvips` manual, available in the `/usr/local/doc/tex` directory, which describes not only the (short) names under which the various PostScript fonts are known to the T_EX programs but it also shows other tricks, which can be used to get special effects with PostScript.

Figure 6.2 shows an example of writing a very large text.

6.1.9 Merging Graphics and Text

Some T_EX macro packages allow the creation of graphics e.g. L^AT_EX's `picture` environment and its extensions, `epic` and `eepic` or `PiCTeX`. These will not be discussed here. We will describe, however, ways to merge text and graphics, if the latter are prepared as Encapsulated PostScript form.

T_EX has a primitive command called `\special`, which allows arbitrary text to be included in the `dvi` at the current position. The text given to the `\special` command is **ignored** by T_EX itself. This text can however be interpreted by the `dvi-driver` when it prepares printable or viewable output from the information in the `dvi` file, in particular it can be used to insert a graphics file. So, the problem of producing proper graphics output is solved if the `dvi-driver` and the printer can handle a given format.

The EPS format is a set of rules for writing programs in PostScript¹. An EPS file starts with two characters `%!` followed by any text, and has some metacomments, starting with `%%`, for instance:

¹ See "Appendix G: Document Structuring Conventions — Version 3.0" and "Appendix H: Encapsulated PostScript File Format — Version 3.0" in *The PostScript Language Reference Manual* [1]

```
%!PS-Adobe-2.0 EPSF-1.2
%%BoundingBox: 40 -505 507 -87
%%Creator:Adobe Illustrator(TM) 1.1
%%Title:elephant.ps
%%CreationDate:25/4/88 7:57 pm
%%DocumentFonts:Courier
%%EndComments
...
```

For EPS conformance only the first line and the one containing the `BoundingBox` comment are mandatory. The program `dvips` which converts a DVI file into PostScript, uses the latter metacomment, which specifies the coordinates of the "bounding box" of the picture (see next section).

What is a bounding box?

In order to be able to properly translate and scale a figure, T_EX must know its "natural" position on the page; this information is present in what is called the *bounding box* of a PostScript program. The bounding box is an outer limit to the marks created by a program, and is specified as four coordinates of a rectangle: the lower-left x coordinate (`bbllx`), the lower-left y coordinate (`bbilly`), the upper-right x coordinate (`bburx`), and the upper-right y coordinate (`bbury`). Adobe uses the convention that the bounding box of a PostScript program must be contained in a "bounding box comment" if the file is to be used as an Encapsulated PostScript figure. It is a line of the form:

```
%%BoundingBox:  bbllx  bbilly  bburx  bbury
```

Note once more that the only mandatory PostScript convention is that the first line of the file should begin with the characters `"%!"` (a `"%"` begins a comment in PostScript). A good place for the bounding box comment is as the second line of the file.

All coordinate values must be given in so-called Big Points (72 big points equal one inch, or $1\text{cm} = 28.35$ big

points), e.g. the bounding box corresponding to an A4 (210 mm by 297 mm) page is:

```
%%BoundingBox: 0 0 595 842
```

If a bounding box comment is present in the figure file, the `epsfig` T_EX interface (see section 6.1.9) will extract its values. The bounding box values may instead be specified directly in the `epsfig` argument, using clauses of the form `bbllx=bbllx`, `bblyy=bblyy`, ... in which case the figure file is not searched for the bounding box.

Producing Encapsulated PostScript pictures

Many programs dealing with graphics, produce Encapsulated PostScript files as graphics output files.

With PAW, workstation type 113 will produce PostScript output with the BoundingBox command parameters included. Below the beginning of a PostScript file generated by this procedure with PAW is shown.

```
!PS-Adobe-2.0 EPSF-2.0
%%BoundingBox: 0 0 567 567
%%Title: /PAW PS A1
%%Creator: HIGZ Version 1.13/00
%%CreationDate: 23/01/92 16.19
%%EndComments
/s stroke def /l lineto def
/m moveto def /t translate def
.....
```

Other picture formats should first be transformed to PostScript before they can be included easily into L^AT_EX. Several public domain programs exist to transform popular picture formats into PostScript (see section 8.3 on page 8-4).

Moreover the utility `xpick` dumps an image of an X window as Encapsulated PostScript (see section 8.3.6 on page 8-9). The image can be color or grayscale.

Usage of the `epsfig` command

The style option `epsfig`, which is based on the `psfig` macros of T.J. Darrell as extended by S. Rahtz, facilitates the inclusion of PostScript figures into T_EX documents. With the help of a compatible DVI postprocessor, PostScript figures are automatically scaled and positioned on the page, and the proper amount of space is reserved. Custom characters such as “∅” and “⊙” may be created and used freely throughout a document, or figures can be presented as traditional broken-out displays (see figs. 6.3, 6.4).



StarLines

Figure 6.3: Encapsulated PostScript example 1

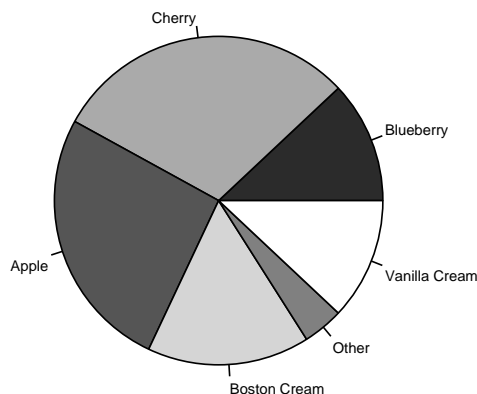


Figure 6.4: Encapsulated PostScript example 2

To include Encapsulated PostScript pictures with `dvips` the style option `epsfig` must be specified as follows:

```
\documentstyle[epsfig,...]{...}
```

The command `\epsfig` is then defined with the following parameters (given in one line !):

```
\epsfig{file=fn,height=ht,width=wd,
bbllx=blx,bblyy=bly,bburx=brx,bbury=bry}
```

- `file` The file name of the Encapsulated PostScript file.
- `figure` Alias for `file=`, i.e. the file name of the Encapsulated PostScript figure.
- `height` The desired height of the picture (in any of the accepted T_EX units). If this parameter is not specified, then the picture will be printed with its “natural” height, i.e. the one specified on the BoundingBox line inside the PostScript file. When a width is specified (see below) and no height, the latter is scaled in the same proportion as the width.
- `width` The desired width of the picture (in any of the accepted T_EX units). If this parameter is

not specified, the picture will be printed with its “natural” width, i.e. the one specified on the `BoundingBox` line inside the `PostScript` file. When a height is specified (see above) and no width, the latter is scaled in the same proportion as the height.

The following *optional* parameters (see discussion of bounding box in paragraph 6.1.9 *What is a bounding box* on page 6-7 and below) should be given in `pt`:

<code>bbllx</code>	The x -coordinate of the lower left hand corner of the <code>BoundingBox</code> of the <code>PostScript</code> picture.
<code>bblyy</code>	The y -coordinate of the lower left hand corner of the <code>BoundingBox</code> of the <code>PostScript</code> picture.
<code>bburx</code>	The x -coordinate of the upper right hand corner of the <code>BoundingBox</code> of the <code>PostScript</code> picture.
<code>bbury</code>	The y -coordinate of the upper right hand corner of the <code>BoundingBox</code> of the <code>PostScript</code> picture.

When the `BoundingBox` parameters are specified on the `epsfig` command, then the ones inside the `PostScript` file are ignored. This facility is particularly useful if the `BoundingBox` parameters are absent from the `PostScript` file or are wrong.

By specifying `BoundingBox` parameters you can cut out a part of a given `postscript` picture. An easy way to specify the appropriate coordinates is to view the `.eps` file with `ghostview` and read the coordinates of the cursor in the upper left corner of the `ghostview` window.

The `\epsfig` macro is (unfortunately) sensitive to whitespace, and will be confused by any extra spaces or newlines in its argument!

Simple figures

The code below shows the simplest way of how one can include an `Encapsulated PostScript` file with `\epsfig`.

```
\documentstyle[epsfig]{article}
\begin{document}
  .... some text ....
  \epsfig{file=input-file}
  .... some more text ....
\end{document}
```

Here `input-file` is the name of a `PostScript` file. `epsfig` will automatically position the figure at the

current point on the page, and reserve the proper amount of space in \TeX so as to avoid blocking any other objects on the page.

As a more realistic example let us include a `EPS` picture (e.g. generated with `PAW`) and specify the desired width on the output page (if we do not specify any dimensions, the “natural” dimensions of the picture are taken, as read on the `BoundingBox` line in the file, corresponding to the size shown when the picture is printed separately on a `PostScript` printer). The upper edge of the picture will be located at the point where the command `epsfig` is issued. The graphic will be scaled to the desired width (or height), but it will be scaled by the same factor horizontally and vertically (if only one of the parameters `height` or `width` are specified). The actual commands typed are given below (the `epsfig` macro must be written in one line!).

```
\begin{figure}
\begin{center}
\mbox{\epsfig{file=../eps/tac2dim.eps,
             width=\linewidth}}
\end{center}
\caption{A single centered figure}
\label{fig:simple}
\end{figure}
```

Figure 6.5 shows the resulting picture, with width of the picture equal to the current `linewidth`. The width (or height) can be given also in `cm` or any other valid \TeX length unit. It is centered by putting it in an `mbox`, which is itself in a `center` environment.

Draft figures and Silent mode

Normally, `epsfig` will print advisory messages to remind you that it is including figures as \TeX processes a document. This behavior can be disabled with `\psilent` and re-enabled with `\psnoisy`.

Some `PostScript` figures can take quite a long time to transmit to the printer and print; for these figures a “draft” mode is available to speed printing of draft versions of the document. A figure printed in draft mode will appear as a box with the name of the figure file (Figure 6.6). The macro `\psdraft` will switch into draft mode, and all subsequent `epsfig` macros will produce draft figures until reaching the macro `\psfull`, which switches out of draft mode. No `\special` commands are used in draft mode, so a draft document can be previewed using any `DVI` driver.

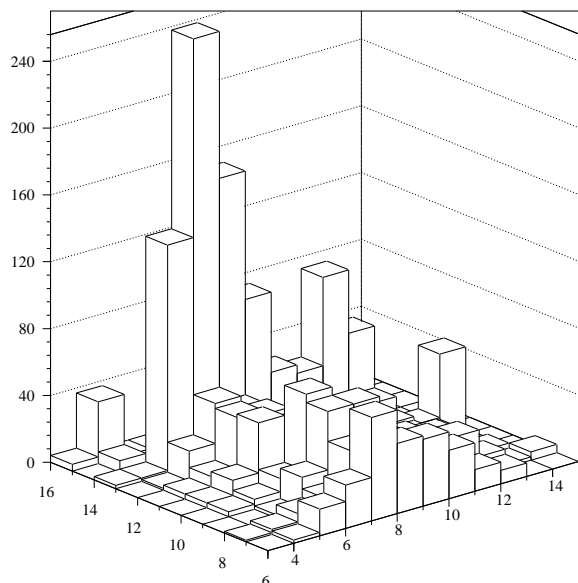


Figure 6.5: A single centered figure

Distorting a figure Users who want to obtain special effects by distorting a figure can specify *both* parameters to the `\epsfig` macro, where both dimensions height and width are taken literally, thus making disproportionate scaling possible. (see figure 6.7).

Colourfull \TeX

With the help of the PostScript driver `dvips`, it is easy to produce colorful slides and include colored pictures into your text. To use the color option, you have to include the document substyle option `dpscolor` into your \LaTeX document or input `dpscolor.tex` into your \TeX document.

After having processed your document in the usual way (`latex`, `dvips`) you can view it with the `ghostview` previewer or print it on the **psteka** color printer in the I/O room (see section 4.4 on page 4-6 for details and a list of actual printers).

A set of macros is available to use colors for the background and foreground text:

The first macro lets the user specify the background color for the document. It sets the background color for the current page and all succeeding pages, unless changed by another command of this type. To change the background color back to the default, issue

```
\background{white}
```



Figure 6.6: The same figure as 6.5, but in draft mode

There are two types of text color commands. The first is in the form `\ColorName` (note the uppercase for the color name). It is called a local color command since it takes one argument enclosed in brackets. It writes the contents of its argument in the selected color. This should be used for local or nested color changes, since it restores the original color state when it completes. The second type of color command is in the form `\text-ColorName`. This uses the same naming convention as before. It is called a global color command since it takes no arguments and simply sets the color at this point.

As an example, to **write this peace of text in JungleGreen**, use the following command:

```
\JungleGreen{write this peace of  
text in JungleGreen}
```

In table 6.1 the list of predefined colors is shown. If you want to define your own colors, look into the style file `/usr/local/lib/tex/inputs/dpscolor.sty`.

6.1.10 Other Stylefiles

This section describes how to use some selected \TeX stylefiles. For more information about this and other styles see also *\TeX on Cern* by M. Goossens and A. Samarin [4] or see WWW-page (URL: <http://www.gsi.de/computing/TeX/TeXHome.html>).

<code>\GreenYellow</code>	<code>\Yellow</code>
<code>\Goldenrod</code>	<code>\Dandelion</code>
<code>\Apricot</code>	<code>\Peach</code>
<code>\Melon</code>	<code>\YellowOrange</code>
<code>\Orange</code>	<code>\BurntOrange</code>
<code>\Bittersweet</code>	<code>\RedOrange</code>
<code>\Mahogany</code>	<code>\Maroon</code>
<code>\BrickRed</code>	<code>\Red</code>
<code>\OrangeRed</code>	<code>\RubineRed</code>
<code>\WildStrawberry</code>	<code>\Salmon</code>
<code>\CarnationPink</code>	<code>\Magenta</code>
<code>\VioletRed</code>	<code>\Rhodamine</code>
<code>\Mulberry</code>	<code>\RedViolet</code>
<code>\Fuchsia</code>	<code>\Lavender</code>
<code>\Thistle</code>	<code>\Orchid</code>
<code>\DarkOrchid</code>	<code>\Purple</code>
<code>\Plum</code>	<code>\Violet</code>
<code>\RoyalPurple</code>	<code>\BlueViolet</code>
<code>\Periwinkle</code>	<code>\CadetBlue</code>
<code>\CornflowerBlue</code>	<code>\MidnightBlue</code>
<code>\NavyBlue</code>	<code>\RoyalBlue</code>
<code>\Blue</code>	<code>\Cerulean</code>
<code>\Cyan</code>	<code>\ProcessBlue</code>
<code>\SkyBlue</code>	<code>\Turquoise</code>
<code>\TealBlue</code>	<code>\Aquamarine</code>
<code>\BlueGreen</code>	<code>\Emerald</code>
<code>\JungleGreen</code>	<code>\SeaGreen</code>
<code>\Green</code>	<code>\ForestGreen</code>
<code>\PineGreen</code>	<code>\LimeGreen</code>
<code>\YellowGreen</code>	<code>\SpringGreen</code>
<code>\OliveGreen</code>	<code>\RawSienna</code>
<code>\Sepia</code>	<code>\Brown</code>
<code>\Tan</code>	<code>\Gray</code>
<code>\Black</code>	<code>\White</code>

Table 6.1: List of predefined colors for use with `dps-color`

```

\begin{center}
\mbox{%
\epsfig{file=rosette.eps,
,height=20mm,width=6mm}
\epsfig{file=rosette.eps,
,height=20mm}
\epsfig{file=rosette.eps,
,height=20mm,width=35mm}
}% end of \mbox
\end{center}

```

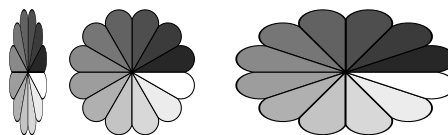


Figure 6.7: Distorting a picture with `epsfig`

gsifoil.sty

If you want to create foils with the GSI-Logo then you can use the minor style `gsifoil`.

There are 5 special commands available:

```

\Trtitle{ } title of the lecture
\Tran{ } title of the foil
\Stitle{ } sectiontitle
\Tr newpage
\Example{ } Examples

```

In the introduction you have also to define:

```

\author{ } name of the author.

```

An example of this stylefile you can find in

`/usr/local/lib/tex/samples/samplefoil.tex`.

gsibrief.sty

The documentstyle `gsibrief` is helpful to create an official GSI-letter with or without GSI-Logo.

You find an example of this stylefile in

`/usr/local/lib/tex/samples/gsibrief.tex`.

gsirep.sty

The documentstyle option `gsirep` creates the layout for the scientific report of GSI.

This means: DIN A 4, twocolumn, titel, author, helvetica postscript-font.



Figure 6.8: Example 1 - gsifoil.sty

To use it write

```
\documentstyle[gsirep]{article}.
```

The first command after `\begin{document}` should be `\maketitle\small`.

fancyheadings.sty

The `documentstyle` option `fancyheadings` allows you to customize your page headers and footers in an easy way. To use this option in your document you must include

```
\documentstyle[fancyheadings, ...]{...},
```

and issue the `\pagestyle{fancy}` command.

You can define:

- three-part headers and footers,
- rules in header and footer,
- headers and footers wider than `\textwidth`,
- multiline headers and footers,



Ges. für Schwerionenforschung - Postfach 13 05 52 - 64220 Darmstadt

Prof. Dumm bach
Auf dem Holzweg 80a
3090 Erbarmsstadt

Planckstr. 1
Postfach 11 05 52
64220 Darmstadt
Telefon (06131) 339 - 1
Durchwahl 339 - 339-351
Telefax (06151) 339-990
Telex 0419 393
Email R221 8DDA GSI3
1. Januar 2000
MF/zi

Lieber Ben,

wie ich höre, gibt es Probleme mit Alka-Seltzer Tabletten. Da jedes Alka-Seltzer Röllchen exakt 25 Tabletten faßt und die Gebrauchsanweisung ein "plop, plop, fizz, fizz" empfiehlt, habe ich erfahren, daß sich bei Dir eine beachtliche Menge an Röllchen angesammelt hat, in denen eine Tablette übrig ist.

Ich beschäftige mich im Augenblick mit einem Forschungsprojekt zur Untersuchung der Anwendungsmöglichkeiten isolierter Analgesi ca. Falls du so freundlich wärest, deine Alka-Seltzer Sammlung unserem Projekt zu spenden, wäre ich mehr als glücklich, Dir Preprints eventueller Progress Reports, die wir über dieses kritische Problem veröffentlichen sollten, zuzusenden.

Mit freundlichen Grüßen
(Matthias Feyerabend)

Kopie: B. Trinker
Sch. Luck

Anlage: Alka Seltzer
Div.

P.S. Falls gewünscht, könnte ich überprüfen, ob Deine Spende und die notwendigen Essen sich von der Steuer abziehen lassen aufgrund unserer Forschungsarbeit.

Figure 6.9: Example 2 - gsibrief.sty

- separate headers and footers for separate even and odd pages,
- separate headers and footers for chapter pages.

An example of this stylefile you can find in

`/usr/local/lib/tex/samples/samplefancy.tex`.

cernmins.sty

You have to use the `documentstyle` `cernmins` for making minutes of a meeting e.g.. There are 3 new commands defined:

```
\Title{ ...} Title for putting at top of front page
\Stitle{ ...} Secondary title for putting as running head
\Those{ ...}{ ...} First parameter text in front, then contents
```

An example of this stylefile you can find in

`/usr/local/lib/tex/samples/samplemins.tex`.

6.1.11 L^AT_EX_{2_ε}—the new L^AT_EX release

L^AT_EX_{2_ε} has better support for fonts, graphics and colour.

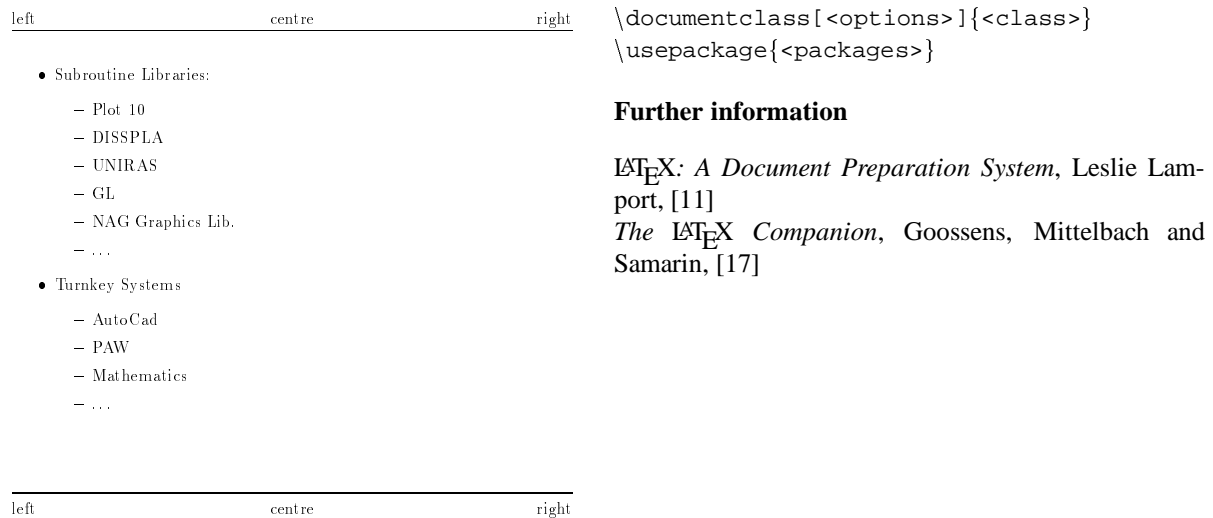


Figure 6.10: Example 3 - fancyheadings.sty

UNIX Meeting

Present:: ...

Excused:: ...

Invited for specific points:: ...

1. Minutes of the last meeting
 - ...
 - ...
2. Matters arising from the minutes
 - ...
 - ...

Next meeting: Monday 20.10.93 at 13:00 Uhr

Recorder: ...

Figure 6.11: Example 4 - cernmins.sty

Processing documents with $\LaTeX 2_{\epsilon}$

Documents written for $\LaTeX 2.09$ will still be read by $\LaTeX 2_{\epsilon}$. Any such document is run in $\LaTeX 2.09$ compatibility mode. Unfortunately, compatibility mode comes with a price: it can run up to 50% slower than $\LaTeX 2.09$ did. If you want to run your document in the faster *native mode*, you should try replacing the line: Unfortunately, this will not always work, because some $\LaTeX 2.09$ packages will only work in $\LaTeX 2_{\epsilon}$ compatibility mode. You should find out if there is a $\LaTeX 2_{\epsilon}$ version of the package available.

```
\documentstyle[<options>, <packages>]{<class>}
with:
```


Chapter 7: Program Development

7.1 Overview

This chapter introduces program development tools under UNIX with simple examples. Its focus is on commands and command options which work on most UNIX systems at GSI, rather than on the differences.

7.2 Compiling and linking a program

To create an executable program, you compile a source file containing a main program. For example, to compile a C program named `hello.c` use:

```
$ cc hello.c
```

If no errors occur, the compiler creates an executable file named `a.out` in the current working directory. This process is essentially the same for most UNIX compilers. For instance, to compile and run a FORTRAN program use:

```
$ f77 hello.f  
$ a.out
```

If your source is divided among separate files, simply specify all files in the compile command:

```
$ cc main.c func1.c ... funcn.c
```

The `-o name` option causes the compiler to name the output file `name` instead of `a.out`. For example, to compile a C program `hello.c` and name the resulting executable `hello` use:

```
$ cc -o hello hello.c
```

The `-c` option suppresses the link-edit phase. The compiler generates an object file with the extension (`.o`) for each input file and not the `a.out` file. This is useful when compiling source files that contain only subprograms, which can be linked later with other object files. The resulting object files can then be specified on the compiler command line:

```
$ cc -c func.c  
$ cc main.c func.o
```

Notice that you need not call the linker. This is done by the compiler. If you have a lot of different object files you can create an **archive** library to store them. The command `ar` is used to create and manage archive libraries. Its syntax is:

```
ar keys archive [obj_files]
```

The most important *keys* are `r` to replace or add modules to the archive and `t` to display a table of contents. The *archive* is a name composed of `libname.a`. For example the command sequence:

```
$ cc -c func1.c  
$ cc -c func2.c  
$ cc -c func3.c  
$ ar r libmy.a func1.o func2.o func3.o
```

compiles `func1.c, func2.c, func3.c` and adds the object files into the archive `libmy.a`. Libraries are specified on the compile commands with the `-lname` option, where *name* is the part of the library name following `lib`. By default the compiler searches the `/lib` and the `/usr/lib` directory for libraries. It is possible to specify additional directories to the search path with the `-L libpath` option. For example:

```
$ cc -lmy -L/u/peter/lib main.c
```

looks in the directories `/u/peter/lib, /lib` and `/usr/local/lib` in that order for `libmy.a`. There are three more important general available compile options:

<code>-I includepath</code>	path for user includes
<code>-O</code>	optimize flag
<code>-g</code>	debug flag

All options described until now work at least with the following compilers:

- **C**
 - `cc` all flavors
- **FORTTRAN**
 - `f77` DEC, IBM RS/6000
 - `fort77` HP
 - `xlf` IBM RS/6000

PL/I compilers are currently available with AIX and Ultrix. Some details will be described in a future release of this document.

C++ compilers are also available. Details will be described in a future release of this document.

7.3 Correcting errors in a program

If your program does not execute properly, you have to use a debugger to locate and correct problems. The old fashioned UNIX debugger `adb` is available on most systems, but it is highly recommended to use the symbolic debugger of your UNIX flavor.

DEC	<code>dxdb</code>
HP	<code>xdb</code>
IBM	<code>dbx</code>
IBM	<code>xde</code> (X11 interface to <code>dbx</code>)

Before invoking a symbolic debugger you should recompile your program with the `-g` option and without any optimization `-O` flags. This ensures that necessary debugging information is incorporated into the object code. The debuggers have many commands for viewing and manipulating programs. You can:

- Control execution with single step execution or the use of breakpoints.
- Look at data values.
- Look at the contents of your source files.
- Look at the execution stack.

A sample of simple commands for the HP `xdb` debugger are:

Command	Description
<code>r</code>	Run the program
<code>b 82</code>	Set a breakpoint at line 82
<code>c</code>	Continue running until the next breakpoint
<code>s</code>	Single step through the next source line
<code>S</code>	Step over a function or subroutine
<code>t</code>	Print a trace of the current execution stack
<code>v</code>	View a window of lines
<code>/string</code>	Search forward in the source for occurrence of <i>string</i>
<code>?string</code>	Search backward in the source for occurrence of <i>string</i>
<code>p abc</code>	Print the value of variable <code>abc</code>
<code>p abc = 2.2</code>	Assign a new value to <code>abc</code>
<code>p buffer\ 10d</code>	Print the first 10 elements of array <code>buffer</code> in decimal format

<code>D "source dir"</code>	Add directory <i>source dir</i> to the list of directories to search for sources (default: current directory)
<code>q</code>	Quit the debugger

On DEC and IBM workstations you see the available options via the X11 interface.

7.4 Building and Maintaining a Program

Under UNIX facilities are provided to help to control changes and build a program from many source modules:

- The `make` command builds a program from source modules. Since the `make` command compiles only those modules that were changed since the last build, its use can reduce compilation time when many source modules must be processed (see below and reference [15]).
- The Source Code Control System (SCCS) is a set of UNIX commands that enable you to maintain separate versions of a program without storing complete copies of each version. Beside the reduction of storage requirements the use of SCCS can help in tracking the development of a project that requires keeping many versions of large programs. For more information consult the manuals.

There are two GNU tools which can be used instead of the SCCS: the Revision Control System (RCS) and the Concurrent Version System (CVS). If you develop programs in CERN software environment you should consider `cmz` (see section 9.2.3 on page 9-6) as code management system.

7.4.1 make

The basic operation of `make` is to update a target file. By ensuring that all of the files on which it depends exist and are up to date, `make` creates the target if it has not been modified since its dependents were. It uses information from a description file named `makefile` or `Makefile`, last-modified times from the file system and some built-in rules.

To illustrate, consider a simple example: A program named `prog` is made by compiling and linking three files `x.c`, `y.c` and `z.c` with the library `libS.a`. The files `x.c` and `y.c` share some declarations `xy.h` (that is they have the line: `#include "xy.h"`). The following `makefile` describes these dependencies:

```

prog : x.o y.o z.o
      cc x.o y.o z.o -ls -o prog

x.o  : x.c xy.h
      cc -c x.c

y.o  : y.c xy.h
      cc -c y.c

z.o  : z.c
      cc -c z.c

```

The first line says that `prog` depends on three object files. Once these object files are current, the second line describes how to link them to create `prog`. It is important that such a command line starts with a tab (`Tab`) sign. If `x.o` is not up to date the third line says that it depends on `x.c` and `xy.h` and so on. If none of the source or object files had changed since the last time `prog` was made, all of the files would be current, and the command `make` would just announce this fact and stop. If, however, the `xy.h` file had been edited, `x.c` and `y.c` would be recompiled, and then the `prog` would be created.

If no target name is given, the first target mentioned in the `makefile` is created. Otherwise the specified targets are made:

```
$ make x.o
```

would recompile `x.o` if `x.c` or `xy.h` had changed. `make` has a simple macro mechanism. Macros are defined by lines with embedded equal signs. A macro is invoked by preceding the name by a dollar sign. Macro names longer than one character must be parenthesized.

```

CC = cc
prog : x.o y.o z.o
      $(CC) x.o y.o z.o -ls -o prog
x.o  : x.c xy.h
      $(CC) -c x.c
y.o  : y.c xy.h
      $(CC) -c y.c
z.o  : z.c
      $(CC) -c z.c

```

In this example the first line defines the macro `$(CC)` to be `cc`, what is used in the compile steps. If you want to use another compiler you have two possibilities: You can edit that line, or macro definitions on the command line override definitions in the `makefile`. The command `make CC=gcc` uses the GNU C compiler instead of `cc`.

To conclude the structure of a description file:

- `make` ignores blank lines,
- characters from a number (#) sign to the end of a line are comments,
- lines containing an equal (=) sign define macros,
- lines containing a colon (:) are dependency lines,
- lines beginning with a tab (`Tab`) sign are command lines.

The most important flag of the `make` command is: `-n` no execute mode (print commands, but do not execute them).

`make` is most useful for medium-sized programming projects, so the typical cycle of program development operations becomes:

think → edit → make → test ...

Chapter 8: Applications and Utilities

8.1 Mathematical packages

There are two interactive systems to meet your needs in scientific computations from symbolics to numerics and graphics:

- **Mathematica** three licenses to be used on rzhp9a/b and clri6a,
- **AXIOM** one license on rzri6f.

You can use both systems as:

- A numerical and symbolic calculator.
- A visualization system for functions and data.
- A programming language.
- A modeling and data analysis environment.

Mathematica is widely used. There are user contributed packages from all fields of science. AXIOM is the newest of all symbolic packages. The advantages of AXIOM are:

- The HyperDoc system, offering on-line help, examples, tutorials and reference material.
- A variety of data structures not available in other systems.

A reference manual for each system [16] and [7] is available at the help desk (Benutzerberatung).

8.1.1 Mathematica

To start Mathematica simply type `math`, to browse the Mathematica hypertext documentation type `mathbook` and click to the desired topics.

```
$ math
Mathematica 2.1 for HP 9000 RISC
Copyright 1988-92 Wolfram Research, Inc.
-- Motif graphics initialized --
In[1]:= Plot3D[Sin[x y],{x,0,Pi},{y,0,Pi}]
Out[1]= -SurfaceGraphics-
In[2]:= PPrint[%]
Out[2]= -SurfaceGraphics-
In[3]:= Display["!psfix > test.ps",%]
Out[3]= -SurfaceGraphics-
In[3]:= Quit
$
```

Figure 8.1 is the result of the `Plot3D` command. In this example you see two possibilities to plot out of Mathematica. The `PPrint` command sends a plot to the default printer (psaa). The `Display` command creates a PostScript file.

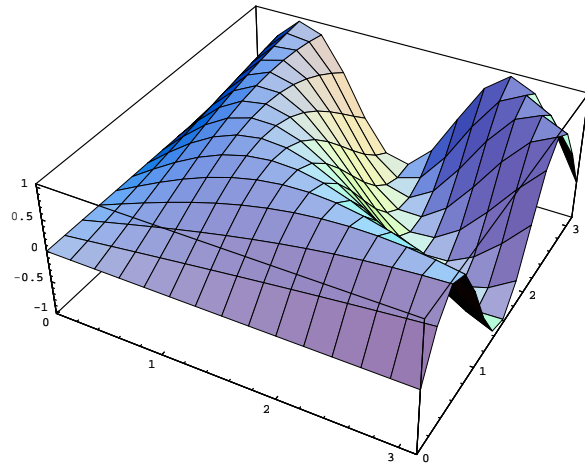


Figure 8.1: Mathematica display

8.1.2 AXIOM

To start AXIOM on rzri6f you have to add `$AXIOM/bin` to your `PATH` and to type `axiom`. In figure 8.2 you see the top level of the AXIOM hyperdocumentation. It is easy to explore the features of AXIOM by clicking on the items in bold font.

8.2 Mathematical Libraries

8.2.1 NAG FORTRAN Library

The NAG FORTRAN Library is available on the RS/6000 and HP cluster. It is a collection of highly efficient numerical and statistical algorithms. Important areas are:

- complex arithmetic
- zeros of polynomials

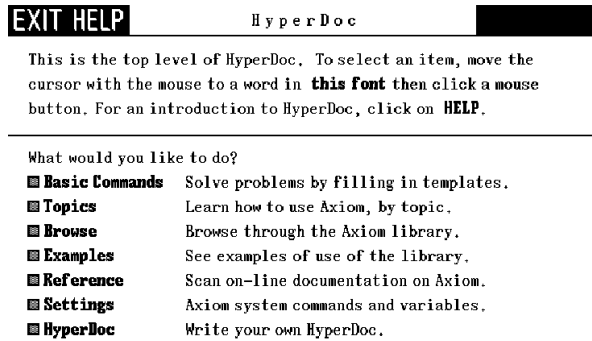


Figure 8.2: AXIOM hyperdoc entry panel

- roots of transcendental equations
- summation of series
- quadrature
- ordinary differential equations
- partial differential equations
- numerical differentiation
- integral equations
- interpolation
- curve and surface fitting
- minimizing or maximizing a function
- matrix operations
- eigenvalues and eigenvectors
- determinants
- simultaneous linear equations
- orthogonalisation
- linear algebra support routines (BLAS)
- linear equations (LAPACK)
- simple calculations on statistical data
- correlation and regression analysis
- multivariate methods
- analysis of variance
- random number generators
- univariate estimation
- nonparametric statistics
- contingency table analysis
- survival analysis
- time series analysis
- operations research
- sorting

- approximations of special functions

A user may link to the NAG Fortran Library in the following manner: `f77 driver.f -lnag` where `driver.f` is the user's application program. If a user wishes to link to the AIX supplied Basic Linear Algebra Subprograms (BLAS) library and/or the IBM Engineering and Scientific Library (ESSL), instead of the NAG BLAS, this can be done in the following manner: `f77 driver.f -lessl -lblas -lnag`

The AIX supplied BLAS Library and/or IBM ESSL should be specified before the NAG Fortran Library.

The following machine-readable information files provided by NAG are in `/usr/local/doc/nag`:

`unUsers' Note.`

`essint Essential Introduction to the NAG Fortran Library.`

`summary a brief summary of the routines.`

Additional help:

- The complete set of printed manuals may be browsed in the user support room 2.244.
- The On-line Information, a key-word-driven guidance via the command `'naghelp'`.
- The command `'nagtest nagprog'` copies an example driver for `'nagprog'` to the current directory, compiles, links and runs it.
- The NAG entry of the GSI WWW page.

Example of naghelp

```
naghelp
HELP Version 2.0100
HELP Portability Interface Version 1.0100

Scope: NAG

Please type one of the following:
SPEC      - for subroutine specifications
           (and list of routines)
STATS     - for advice on statistical
           subroutines
GRAPHICS  - for advice on the NAG Graphics
           Library
NUMERIC   - for advice on all other numerical
           subroutines
CONTENTS  - for the contents of the NAG library
NEWS      - for new routines in this Mark, etc
PRODUCTS  - for other NAG products
LOCAL     - for information about using
           NAG at your site
GUIDE     - for a guide to the NAG database
NOTES     - for notes on this scope

... Type a response or .Q to quit, .H for help
... Response? numeric
Scope: NAG Numerical Advice
```

Type one of the following menu items or Chapter names:

C - non-linear equations, fast Fourier transforms, etc (Chapters C02, C05, C06)

D - integration, differential equations, differentiation, integral equations (Chapters D01, D02, D03, D04, D05)

E - interpolation, curve and surface fitting, optimization (Chapters E01, E02, E04)

F - linear algebra (Chapters F01, F02, F03, F04, F05, F06, F07) and its terminology (type TERM)

S - special functions (Chapter S)

MISC - complex arithmetic, operations research, sorting, error trapping, constants, inner products, utilities (Chapters A02, H, M01, P01, X01, X02, X03, X04, X05)

GUIDE - for a guide to the NAG database

NOTES - for notes on this scope

... Type a response or .Q to quit, .H for help
... Response? d

Type one of the following:

D01 - for quadrature (numerical integration)

D02 - for ordinary differential equations

D03 - for partial differential equations

D04 - for numerical differentiation

D05 - for integral equations

... Type a response or .Q to quit, .H for help
... Response? d03

PARTIAL DIFFERENTIAL EQUATIONS

Type:

ALL - for all information

GEN - for general information

ELL - for elliptic equations

PAR - for parabolic equations

HYP - for hyperbolic equations

... Type a response or .Q to quit, .H for help
... Response? ell

The routine D03EAF solves Laplace's equation by an integral equation method in an arbitrary domain, perhaps containing holes. The boundary conditions may be of Dirichlet, Neumann or mixed type.

The routine D03FAF solves the Helmholtz equation over a 3-dimensional cuboid region by a fast Fourier transform method, and is efficient on vector machines.

There are six general purpose routines for elliptic

partial differential equations:

D03EDF - solves linear equations generated by a seven-point finite-difference scheme for an elliptic P.D.E. and its boundary conditions defined on a rectangle, by a multigrid method due to Wesseling

D03EEF - discretises an elliptic P.D.E. and its boundary conditions defined on a rectangle using a seven-point finite-difference scheme. D03EDF may then be called to solve the linear equations generated

D03EBF - solves linear equations generated by a 2-dimensional five-point finite-difference scheme by Stone's strongly implicit procedure (SIP)

... Pause: press RETURN for more or type .Q to quit,
... ? .q

Example of nagtest

```
$ nagtest d03eef
xlf -lnag -L/usr/local/lib d03eefe.f
** _main    === End of Compilation 1 ===
** pdef     === End of Compilation 2 ===
** bndy     === End of Compilation 3 ===
** fexact   === End of Compilation 4 ===
1501-510  Compilation successful for file d03eefe.f.
time a.out
D03EEF Example Program Results

** The linear equations were not diagonally
    dominated
** ABNORMAL EXIT from NAG Library routine D03EEF:
    IFAIL =      6
** NAG soft failure - control returned

Exact solution above computed solution

    I/J      1      2      3      4      5      6
    9      .000  .105  .208  .308  .403  .492
    9      .000  .105  .208  .308  .403  .492
.....

    1      .000  .000  .000  .000  .000  .000
    1      -.004 -.004 -.003 -.003 -.002 -
.001

Number of Iterations = 10
RMS Error = 2.89D-03
```

```
Exact solution above computed solution

  I/J      1      2      3      4      5      6
  9      .000   .105   .208   .308   .403   .492
  9      .000   .105   .208   .308   .403   .492
.....
  1      .000   .000   .000   .000   .000   .000
  1     -.018  -.018  -.017  -.015  -.013  -
.009

Number of Iterations = 4
RMS Error = 1.21D-02
STOP

real 0m0.32s
user 0m0.02s
sys 0m0.01s
$
```

8.2.2 The ESSL Library

The Engineering and Scientific Subroutine Library (ESSL) is a set of high performance mathematical subroutines, which can be used with FORTRAN, PL/1, C and assembler. ESSL is specially tuned for the IBM RS 6000. It includes subroutines for:

- Linear algebra subprograms
- Matrix operations
- Eigensystem analysis
- Fourier transforms, convolutions/correlations, and related computations
- Sorting and searching
- Interpolation
- Numerical quadrature
- Random number generation

To access ESSL, you can use for example: `f77 -O -lessl xyz.f` where `xyz.f` is the name of your FORTRAN program. You should use the ESSL if you want highest performance on a RS/6000, but remember that it is an IBM only library. Online documentation for the ESSL is available via BookManager.

8.2.3 LAPACK

LAPACK is a public domain package for linear algebra of dense systems. It is included in the NAG FORTRAN library. To use it simply link to that library. If you need the source code of LAPACK, contact P. Malzacher (551).

8.3 Graphical Tools

Graphics is used as an essential link between human and computer. Visualization of data allows to extract information much better rather than interpreting raw numbers. Consequently there is a growing demand for high performance graphical tools.

The easiest way to create graphics images is to use complete graphics packages. There are two types of packages available: graphics interpreters with commands and procedures, or menu driven systems. GSI offers both types of systems: IDL as one of the best examples for graphics interpreters and the IBM Data Explorer as an AVS like system.

To allow for a free interchange of graphical information between different UNIX-platforms, we support standardized graphical tools and a set of conversion routines to transform from and to different formats of graphical output, which are also introduced in the following.

8.3.1 Interactive Graphics with IDL

IDL, Interactive Data analysis Language, is a complete package for the interactive reduction, analysis, and visualization of scientific data and images. Optimized for the workstation environment, IDL integrates a responsive array oriented language with numerous data analysis methods and an extensive variety of two and three dimensional displays into a powerful tool for researchers.

IDL supports an extensive data import capability, publication quality hard copy output, and user-defined Motif graphical user interfaces.

Users can create complex visualizations in hours instead of weeks with the aid of IDL's high level capabilities and interactive environment.

IDL is useful in physics, astronomy, image and signal processing, mapping, medical imaging, statistics, and other technical disciplines requiring visualization of large amounts of data.

This version supports the following systems (besides others):

- IBM 6000: AIX 3.2.
- HP 9000 Series 700: HP-UX 9.01.
- Risc Ultrix: Ultrix 4.2.
- DEC ALPHA: OSF1 1.2.
- VMS: [ALPHA] OpenVMS AXP 1.5.
- VMS: [VAX] VMS 5.1 and up.
- Apple Macintosh computers running System 7
- DOS based personal computers running Microsoft Windows 3.1 or later

At GSI the newest version 3.5 (Nov. 23, 1993) is currently available on the IBM workstations (30 days trial version). In 1994 we will offer IDL on all GSI workstations and on VAX computers, too. The PC trial version is installed at the 'Benutzerberatung'.

After starting IDL (currently with `IDL` in uppercase letters) you get online help with the `?` command. You also may start different demos: with `.run demo` you will see a demo showing commands and the associated image output. With `xdemo` a window oriented demo with an IDL tour and some special applications are available. IDL is a command language with a large amount of builtin or user written procedures and functions; in addition the user may develop his or her own procedures and functions. You have to get used with the comma just after the command name, e.g.

```
PRINT, 3*5
```

But besides this it is very simple to create nice plots with only a few commands, e.g.

```
example
a = FINDGEN(100)
PLOT, sin(a/5) / exp(a/50)
```

Output will be generated for the X terminal and for files in the PostScript format.

A more complex image 8.3 was created with the following commands:

```
more complex example
a = FINDGEN(100) ; define an array 0 .. 99
b = sin(a/5) / exp(a/50) ; damped sine wave
LOADCT, 0 ; load B&W color table
; image, wire frame + contour
SHOW3, b#FINDGEN(50)
```

There is an interface to IBM's Data Explorer under development to tap the potential of both powerful scientific computing packages.

8.3.2 Handling of Image Files

In order to support modification of pixel data and conversion from and to different image file standards the following tools are made available.

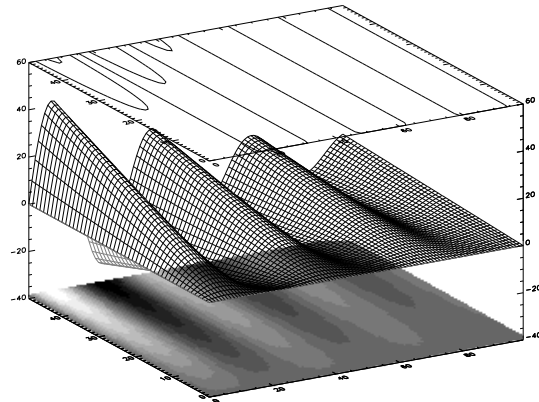


Figure 8.3: IDL display

University of Utah Raster Toolkit, URT

The Utah Raster Toolkit is a collection of programs and C routines for dealing with raster images commonly encountered in computer graphics. A device and system independent image format stores images and information about them. Called the RLE format, it uses run length encoding to reduce storage space for most images.

The programs (tools) currently included in the toolkit are listed below, together with a short description of each one. Most of the tools read one or more input RLE files and produce an output RLE file. Some generate RLE files from other information, and some read RLE files and produce output of a different form. For general information on available tools use

`man urt`

For specific information on a chosen command and parameter list type e.g.

`man applymap`

urt - overview of the Utah Raster Toolkit

applymap Apply color map to image data.

avg4 Simple 2x2 downsizing filter.

crop Crop image.

fant Image scale/rotate with anti-aliasing.

getx11 Display using X11.

giftorle Convert gif files to RLE.

graytorle Convert separate rgb files to RLE.

mcut Median cut color quantization.

mergechan Merge colors from multiple images.

pyrmask Generate “pyramid” filter mask.

rawtorle Convert various raw formats to RLE.

repos Reposition an image.

rleClock Draws a clock face.

rleaddcom Add comments to an RLE file.

rleaddeof Add an EOF code to an RLE file.

rlebg Generate a “background”.

rlebox Find bounding box of an image.

rlecomp Image composition.

rledither dither image to a given colormap.

rleflip Flip an image or rotate it 90.

rlehdr Print info about an RLE file.

rlehisto Make a histogram of an image.

rleldmap Load a new colormap into a file.

rlemandl Make a Mandelbrot image.

rlenoise Add noise to an image.

rlepatch Patch smaller images on a big one.

rleprint Print all pixel values in image.

rlequant Variance based color quantization.

rlescale Generate a “gray scale”.

rleselect Select images from an RLE file.

rlesetbg Set the background color of an image file.

rlespiff Simple contrast enhancement.

rlesplice Splice two images horizontally or vertically.

rlesplit Split concatenated images into files.

rleswap Swap or select color channels.

rletoascii Make a line-printer version of an RLE image.

rletogif Convert RLE images to gif format.

rletogray Convert RLE to separate rrr ggg bbb files.

rletops Convert RLE to (BW) PostScript.

rletoraw Convert RLE to rgbrgb raw format.

rletotiff Convert RLE to tiff 24 bit format.

rlezoom Scale image by sub- or super-sampling.

smush Generic filtering.

to8 24 to 8 bit ordered dither color conversion.

tobw Color -> BW conversion.

unexp Convert “exp” format to normal colors.

unslice Paste together “slices” into a full image.

An input file is almost always specified by mentioning its name on the command line. Some commands, usually those which take an indefinite number of non-file arguments (e.g., **rleaddcom**) require a **-i** flag to introduce the input file name. If the input file name is absent the tool will usually read from the standard input. An input file name of “-” also signals that the input should be taken from the standard input.

On Unix systems, there are two other specially treated file name forms. A file name starting with the character “|” will be passed to sh to run as a command. The output from the command will be read by the tool. A file whose name ends in “.Z” (and which does not begin with a “|”) will be decompressed by the **compress** program. Both of these options supply input to the tool through a pipe. Consequently, certain programs (those that must read their input twice) cannot take advantage of these features. This is noted in the manual pages for the affected commands.

An output file is almost always specified using the option **-o outfile**. If the option is missing, or if **outfile** is “-”, then the output will be written to the standard output.

On Unix systems, the special file name forms above may also be used for output files. File names starting with ‘:’ are taken as a command to which the tool output will be sent. If the file name ends in “.Z”, then **compress** will be used to produce a compressed output file.

Several images may be concatenated together into a single file, and most of the tools will properly process all the images. Those that will not are noted in their respective man pages.

Picture comments

Images stored in RLE form may have attached comments. There are some comments that are interpreted, created or manipulated by certain of the tools. In the

list below, a word enclosed in <> is a place-holder for a value. The <> do not appear in the actual comment.

image_gamma=<float number>

Images are sometimes computed with a particular “gamma” value – that is, the pixel values in the image are related to the actual intensity by a power law,

$$pixel_value = intensity^{image_gamma}$$

Some of the display programs, and the buildmap function will look for this comment and automatically build a “compensation table” to transform the pixel values back to true intensity values.

display_gamma=<float number>

The display_gamma is just image_gamma. That is, it is the “gamma” of the display for which the image was computed. If an image_gamma comment is not present, but a display_gamma is, the displayed image will be gamma corrected as above. The to8 program produces a display_gamma comment.

colormap_length=<integer>

The length of the colormap stored in the RLE header must be a power of two. However, the number of useful entries in the colormap may be smaller than this. This comment can be used to tell some of the display programs (getx11, in particular) how many of the colormap entries are used. The assumption is that entries 0 colormap_length1 are used. This comment is produced by mcut, rlequant, and rledither.

image_title=<string>

This comment is used by getx11 to set the window title. If present, the comment is used instead of the file name. (No other programs currently pay attention to this comment.) The comments IMAGE_TITLE, title, and TITLE are also recognized, in that order. No programs produce this comment.

HISTORY=<string>

All toolkit programs (with the exception of rleaddcom) create or add to a HISTORY comment. Each tool appends a line to this comment that contains its command line arguments and the time it was run. Thus, the image contains a history of all the things that were done to it. No programs interpret this comment.

exponential_data

This comment should be present in a file stored in “exponential” form. See unexp and float_to_exp for more information. The unexp program expects to see this comment.

8.3.3 Image Format Conversion

To convert different image formats, besides URT the command imconv (AIX only) supports the following

formats:

gif, hdf, icon, iff, mpnt, pbm, pcx, pgm, pic, pict, pix, pnm, ppm, ps, ras, rgb, rle, rpbm, rpgm, rpnm, rppm, synu, tif, x, xbm, xwd.

image conversion is performed due to suffixes, for example,

imconv infile.gif outfile.ps

converts the gif-format file infile.gif to PostScript-format file outfile.ps or alternatively

imconv -xwd a -ps b

transforms the inputfile named a from xwd format to the output file named b in PostScript-format.

Under X-Windows, window images may be stored in a dump file. The target window is selected by mouse click. The keyboard bell is ringing once at the beginning of the dump and twice at the end, e.g, the commands:

xwd -out mywindow.xwd

imconv mywindow.xwd mywindow.rle

getx11 mywindow.rle

generate a X-window dump file named mywindow.xwd, convert it to the RLE-format file mywindow.rle and viewed once again.

A truncated subset of the functionality of the commands xwd and imconv is available under HP-Unix via XtoPS. e.g. the command

XtoPS +border mywindow.ps

generates a postscript-file named mywindow.ps excluding the image-borders of the window chosen.

8.3.4 PHIGS

PHIGS (Programmer’s Hierarchical Interactive Graphics System) is a programming interface (subroutine package) used in the development of graphics applications. It is based on the American National Standards Institute (ANSI) and International Standard Organization (ISO) standard:

PHIGS is a graphics system that supports the definition, modification, and display of hierarchically organized graphics data. It provides graphics application developers with a significant amount of additional function beyond the CORE and GKS-2D and GKS-3D systems. The system adds new concepts to provide a interactive, three-dimensional system that enhances the design and visualization process. The ability to organize graphic primitives into hierarchical structures makes it possible to edit, modify, and transform graphic entities.

Using over 300 high-level graphic functions, programmers can develop applications in various programming

languages. ISO-standard is defined for C and FORTRAN77 binding. On the other hand, a subset of about 50 functions suffices in most application problems.

PHIGS offers a set of device-independent programming tools, it decides whether to use local device processing or to have your Central Processing Unit do the processing for your less intelligent workstations. Identical PHIGS functions are available on both mainframe and standalone environments.

Following is a brief summary of common terms and their definitions:

Primitives Graphic objects are defined by a sequence of elements, including output primitives, attributes, and transformations. Basic output primitive elements include lines, markers, polygons, and text definitions.

Attributes Attributes define the characteristics of an output primitive. An attribute, for example, may define the color of a polyline primitive or size of a polymarker primitive.

Structures The graphical primitives, attributes and model transformations are grouped together to form structures. A structure may be used to represent the geometry of an object as well as information regarding the appearance of that object. Elements may be inserted into, or deleted from, structures at any time, in an operation called structure editing. This editing capability minimizes the need to redefine data in order to modify it. Structures may be related in a number of ways including geometrically, hierarchically, or characteristically, according to your application needs.

Input PHIGS provides essential tools for application interaction. Input devices operate synchronously or asynchronously, relay information to the application, which in turn responds by defining, editing, or displaying the graphical data. PHIGS supports six classes of input devices. These classes represent generic physical devices that differ from one another by the type of data they return to the application. Input device classes include the following:

Locator, Stroke, Valuator, Choice, Pick, String.

Three modes of interaction that allow you to request and obtain data from a logical input device. In REQUEST mode, your application prompts for input

and then waits until the operator either enters the requested input or performs a break action which terminates interaction. In SAMPLE mode, your application obtains the current values of the input device by explicitly sampling it. In the EVENT mode, an asynchronous environment is established between your application and a chosen device. In this mode, both your application and any corresponding device operate independently of each other with the help of a centralized input queue.

Workstations The term “workstation” refers to an abstraction of a physical graphics device. It provides the logical interface through which your application program controls physical devices.

PHIGS provides an environment that supports multiple workstations. How your application interacts with a particular workstation depends on the interactive capabilities of that workstation and the design of your application.

PHIGS supports three categories of workstations: INPUT, OUTPUT, and OUTIN. The capabilities of a workstation determine its category. For example, a workstation such as a plotter may only be capable of generating output. Still another, such as an interactive design station, may be capable of providing both input and output.

Inquiry Functions Inquiry functions allow the application programmer to access the program data contained in state lists, description tables, or structures. They are useful for determining both error conditions and device characteristics.

States The system state defines whether the graPHIGS API has been activated or deactivated, using the Open graPHIGS or Close graPHIGS functions respectively. No other functions can be accessed until the system is “open”.

The workstation state defines whether a workstation has been activated or deactivated, using the Open Workstation or Close Workstation functions respectively. The PHIGS structure display functions can only be used if workstation is “open”.

The structure state defines whether PHIGS display structure is “open” and able to be modified or closed and unavailable for modification. A structure is

opened and closed with the Open Structure and Close Structure subroutines. Graphics primitives and attributes can only be created if the structure state is “open”.

Implementations of PHIGS

I. AIX

ISO-PHIGS subroutines are available as a subset of the graPHIGS-IBM library version 2.2 which is available under AIX on RISC6000 workstations and may be used within C-, FORTRAN- and PLI-applications.

Example programs for the use of graPHIGS subroutines are installed in individual sub-directories in the directory:

```
/usr/lpp/graPHIGS/
samples/gettingstarted
```

An interactive tutorial is also available and may be run by just typing: gPtutor

```
cc -o square -lgP square.c
```

compiles the file square.c and links with the graPHIGS library yielding the executable square.

The GDF file standards are different for AIX and MVS. The command

```
cvtgdf
```

transforms AIX standard to MVS standard. This converted file may be transferred via FTP to your MVS userid and may be interpreted using TSO GDFIP.

For further information you can use the info-explorer accessing the RAM-DISCS or use the graPHIGS IBM subroutine reference version 2.2 directly.

Literature

IBM-graPHIGS

The graPHIGS Programming Interface,

- Subroutine Reference, version 2.2, SC33-8194-1
- Understanding Concepts, version 2.2, SC33-8191-1
- Writing Applications, version 2.2, SC33-8192-1
- Messages and Codes, version 2.2, SC33-8196-1
- ISO PHIGS Implementation Reference, version 2.2, SC33-8118-00

Graphics in General

- *A Practical Introduction to PHIGS and PHIGS PLUS* by T.L.J. Howard, W.T. Hewitt, R.J. Hubbard, K.M. Wyrwas [6]
- *Computer Graphics: Principles and Practice* by J.D. Foley, A. van Dam, S.K. Feiner, J.F. Hughes [3]

8.3.5 xv: Interactive Image Display for the X Window System

xv is a X11 program that displays images in the gif, jpeg (jpg), tiff, pbm, pgm, ppm, X11 bitmap (xbm), Utah Raster Toolkit rle, PDS/VICAR, Sun Rasterfile, bmp, pcx, IRIS RGB, possibly PostScript, and pm formats on workstations and terminals running the X Window System, Version 11.

The documentation for xv is now distributed only as a PostScript file and can be found in /usr/local/doc/graphics/xvdocs.ps.

How xv works

You can either invoke xv with the name of an image file or without any parameter. In the latter case, a default startup picture is displayed.

To invoke the command menu, place the mouse in the graphics window of xv and press the right mouse button (MB3).

The most important features are to load or save files in a variety of formats (saves also Encapsulated PostScript for printout or inclusion in other documents), to change the size of the image, perform various operations on the image including dithering, smoothing, rotating or color editing. You can also grab a window or an arbitrary rectangle of the screen and store it later, but consider to use xpick (see section 8.3.6), because of the compression algorithms used for the latter program. As an example, a 462 × 427 pixel image uses 1.2 MB as uncompressed Encapsulated PostScriptfile with xv, 74 kB as compressed Encapsulated PostScriptfile with xv and 20 kB as compressed Encapsulated PostScriptfile with xpick.

8.3.6 xpick: Pick images from an X11-screen

The program xpick lets you pick an image from a window or arbitrary rectangle area of an X11-server and write it to a file in a variety of formats:

```
xpick [-] [ file]
```

The format is defined by the extension in the file name. Possible extensions are:

- .ps** A PostScript file with a compressed image. The image is centered, rotated and scaled to fill the maximum space on a page. It is displayed in color on viewers and printers that support color Postscript, otherwise it is displayed as grayscale.
- .eps** An Encapsulated PostScript file with a compressed image, keeping the original dimensions and containing a BoundingBox starting at (0,0). It is intended for insertion into a document.
- .epsi** Similar to .eps file, but a black and white preview image is added to the file.
- .gif** Graphics Interchange Format (gif). Use this format when you want to keep files or transfer them to other computers. It is also convenient for visualizations, for example, with xv by John Bradley.
- .pcx** pcx format for IBM PC.
- .pict** pict format for Macintosh.
- .ppm** Portable PixMap format (ppm) from the PBM+ library by Jef Poskanzer.

The main difference between `xpick` and similar tools for capturing screen images of an X11 servers, like `xwd`, `xv` or `XtOPS` are:

- easy operation without the need of specifying many parameters allows natural selection of the rectangular area to be picked by either moving the mouse over a window (only window contents, window + frame, pop-up windows) or dragging with Mb1 pressed.
- the compression scheme used for image encoding in Postscript files: `xpick` uses the Lepmpel-Ziv Welch (LZW) compression algorithms, thus producing very compact files (4-5 times less than files produced with Run-Length encoding and 10-20 times less than files produced without compression)

Almost all screen dumps in this Unix primer have been produced with `xpick` in eps format.

How `xpick` works

The easiest way to invoke `xpick` is just to type

```
xpick
```

You will then be prompted to supply an output file name.

Alternatively, you can also specify the file name on the command line, e.g.

```
xpick exampl.eps
```

When `xpick` is invoked, the user sees a blinking rectangle surrounding the contents of the window in which the mouse pointer is currently placed. If the mouse pointer is placed on the Window Manager border of a window, then the blinking rectangle will surround the window together with the Window Manager border. To select the image inside the blinking rectangle it is sufficient to click the left mouse button.

If an arbitrary rectangle is required, then hold down the left mouse button (Mb1) to choose the first corner of the perimeter and then drag the mouse to define the opposite corner. The blinking rectangle will expand with the movement of the mouse.

After the rectangle has been selected the user should press the `[Space]` bar to pick the image.

Options

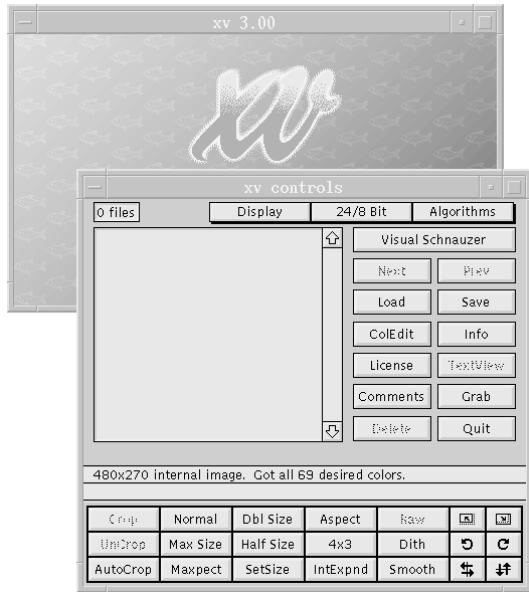
- When `-` is in the parameter list, `xpick` picks an image from the window under the mouse pointer. This option is intended to pick images from pop-up menus, which are on the screen only when a mouse button is pressed and disappear immediately after the button is released.

8.4 Screendump on XScreens

For making a screendump there are several possibilities. In this chapter we will describe the three major possibilities:

8.4.1 The interactive way: `xv`

The easiest way is, to use the interactive program `xv`: After startup the `xv` program by typing `xv` from the commandline `xv` will show the startup image. Pressing the third mousbutton the `xv` Control pannel will start in a seperated window.



After pressing the Grab button you can do the screendump in two different ways:

- pressing the first mousebutton in the desired window will dump the whole window. Attention: overlapping areas with the dumped windows will not be restored!
- pressing the middle mousebutton enables you to drag a rectangle area on the screen

After generating the screendump you could save this to several file format by pressing the Save button in the xv Control panel.

8.4.2 The commandline way: xwpick

The xwpick let's make you the window dump from the commandline. Siplly typing `xwpick` will tell you what to do:

```
[kay@clri6a]/u/kay> xwpick
*****
*
*      XWPICK 2.20 - Pick X Window Image
* Possible file names: *.ps (PostScript)
*                    *.eps (Encapsulated PS)
*                    *.epsi (EPS with preview)
*                    *.gif (Graphics Interchange Format)
*                    *.pcx (IBM PC)
*                    *.pict (Macintosh)
*                    *.ppm (PBM Plus)
*
* ***** Evgeni Chernyaev (chernyaev@mx.ihep.su) *****
File name:
```

After typing in the filename ending with the desired extension let appear a small cross cursor and will flicker the rectangle around the window below this. Pressing the first mousebutton will cause the screendump to written to file. The xwpick has the advantage, that it's eps output is the smallest of these programms. Manpages are available.

8.4.3 The classic way: xwd

As xwd is contributed with the X11 software, it should be available on nearly every platform that has X11. It's the oldest way of making a dump and uses a special file format, the xwd format that could be viewed with the xwd program. the handling is somewhat similar to the xwpick command, as it's starting from the commandline and asks by a small cross cursor for the input window.

The main advantage is that it has an special parameter that enables you to switch of the cross cursor window request:

```
sleep 20; xwd -root > XScreen.xwd
```

After typing this command, you have 20 seconds to arrange your windows, especially walking through pull-down menus. The end of the screendump process will be signaled by a double beep.

Chapter 9: Experiment Data Processing

9.1 Experiment Data Handling

Currently most experiment data acquired and evaluated at GSI are stored under control of the **Hierarchical Storage Manager** (HSM) in the **Automatic Tape Library** (ATL), both running under the MVS operating system.

On a Unix workstation there are several possibilities to access experiment data from the central fileserver:

- Files can be copied with a GSI utility (command **copymvs**) between server and Unix workstation,
- they can be copied via **ftp**, or
- MVS files (called datasets) can be mounted via **NFS**.

Using **ftp** some knowledge on data storage in MVS is required. Therefore a client-server program package has been developed enabling **full control on the client side** (AIX) without any knowledge on MVS. It is specially designed for large file transfers in the GSI environment and described in section 9.1.4 on page 9-2.

Normally the root privilege is required to mount a remote file system. However, on HP workstations a tool is available enabling **normally privileged users** to mount files from the central server in special directories (see 9.1.6 on page 9-4).

Experiment data analysis on the Unix platforms at GSI is done in general with the Cern software (see chapter 9.2 on page 9-5). However, most experiment data acquired at GSI are available in GOOSY format. Currently two program packages are available that help to analyze GOOSY events with Cern software. Both provide an interface to the HBOOK package and allow the generation of histograms and N-tuples for further study with PAW.

1. A program package called **APE** has been developed in the Epos group. An extensive documentation is available as postscript file in `/cern/doc/ape.ps`. In a data analysis with APE the GOOSY events are read from a file.
2. At GSI client-server systems are available providing **events over the network**. They may be sent on-line from a data acquisition on a Single Branch System (SBS) or from a GOOSY session on a Vax. The clients are presented below in section 9.3 on page 9-11.

Most informations and documentations on experiment data handling and analysis are accessible in the WWW. The *Computing* home page contains a section on **Experiment Data Processing and Electronics** with many entries covering the whole area.

9.1.1 Experiment Data in Unix

On Unix platforms, experiment data should be kept in the `/d` file system. Each user group owns a directory named `/d/groupname` inside which any group member may create his/her own subdirectory (with the **mkdir** command) for the local files. The name of the subdirectory should be identical with the account name of the user:

`/d/groupname/username`

Currently no special procedure exists which automatically deletes files after some time without access. Instead it is assumed that each user erases own files no longer needed.

Reading the data you must be aware of the blocksize, which is normally different from the Unix default of 512 bytes. For example, raw data available in GOOSY or SATAN format typically consist of blocks of 4kB or multiples.

To read experiment data with a Fortran analysis program, treat them as unformatted direct access files. For example, 8kB buffers can be read with Fortran statements of the following type:

Buffer Read Sample

```
INTEGER*4 L_BUFLEN
PARAMETER( L_BUFLEN=8192 )
INTEGER*2 I_DATA(L_BUFLEN/2)
OPEN( UNIT = unit,
1      FORM = 'UNFORMATTED',
1      ACCESS = 'DIRECT',
1      RECL = L_BUFLEN,
1      FILE = file,
1      STATUS = 'OLD',
1      ERR = errorlabel,
1      IOSTAT = ios)
```

9.1.2 The Central Experiment Data Server

The experiment data on the central data server are stored in the Automatic Tape Library (ATL). The ATL has eight 3490E cartridge drives and a capacity of 5152 cartridge volumes. More than 3000 Gbytes of experiment data are stored in the ATL - **available 24 hours each day without operator intervention**. The maximum capacity amounts to about 4200 GBytes.

Additionally about 1000 Gbytes of experiment data not used for some time are stored on cartridges kept **outside of the ATL**. These data can be mounted only during operator times (monday to friday from 6 am to 11 pm). Therefore, before accessing older data, you should check their location with the MVS command

```
fwhere filename
```

If the files are neither on (staging) disk nor in the ATL, you should ask the operators to put them into the ATL.

All data in the ATL are accessible via file name and **visible as if they were on disk**. File lists with wildcard character support ('*') may be obtained

- over the network with the `dir` or `ls` subcommands of `ftp`, or
- locally in MVS with the ISPF 3.4 menu.

Data in the ATL cannot be accessed directly, but are automatically **recalled** by the Hierarchical Storage Manager HSM to a staging disk pool with a capacity of about 62 Gbytes.

If no longer needed, the HSM should be invoked to **migrate** the data back to the ATL. Otherwise - though very large - the staging disk pool might be overloaded! To avoid that, the centrally offered tools for access to the experiment data server from AIX (see section 9.1.4 on page 9-2) and from VMS do this automatically!

9.1.3 Experiment Data Naming Conventions in MVS

When you put experiment data files to the ATL, you should follow the MVS naming conventions. Only then proper treatment of the files in MVS is assured!

The recommended MVS names for experiment data files are summarized in table 9.1.3 on page 9-3. The following rules apply:

- MVS dataset names consist of several parts separated by dots and are **not case-sensitive**.

- They always start with the MVS user identification (*mvsuid* in table 9.1.3)
- The second name part must be LMDV, LMDI, DSTV, DSTI, or DSTX - this assures that the experiment data are properly handled by the HSM and not treated like source code or text files.
- The remainder *dsn* may consist - with some constraints - of any combination of letters, digits, and the special characters '\$', '#', and '@'. The restrictions:
 1. Immediately after a dot, only letters, \$, @, and # are allowed.
 2. The number of characters between two dots must not exceed 8.
 3. The maximum total length of an MVS dataset name is 44.

9.1.4 A Client-Server Package for Access to the Experiment Data Server

For efficient transfer of experiment data files between the central data server in MVS and a Unix workstation, a client-server program package has been developed in the 4π collaboration at GSI. It is centrally offered and maintained by the DV&EE department.

The package supports binary data with fixed record length, which matches all common experiment data formats at GSI, and is specially designed for the transfer of large amounts of data.

It provides an appropriate tool for Unix clients to control in an efficient way both, the HSM and the MVS file allocation, without the need of having MVS knowledge. It decreases not only the execution time, as files are recalled in advance from the ATL to staging disks, but also saves staging disk space, as files no longer needed are migrated back to the ATL immediately. Currently this package is available only under AIX.

Three shell commands are offered:

1. `copymvs` to copy a list of files between **MVS disk** and **Unix tape or disk**,
2. `copybin` to copy a list of files between a local **Unix disk** and a local **Unix tape**,
3. `tscan` to scan tapes for files with ANSI Label (AL) file headers.

MVS dataset name	description	byte ordering
mvsuid.LMDV.dsn	raw data	high byte at high address end (DEC)
mvsuid.LMDI.dsn	raw data	high byte at low address end (IBM, HP)
mvsuid.DSTV.dsn	DST data, CERN FZ native	high byte at high address end (DEC)
mvsuid.DSTI.dsn	DST data, CERN FZ native	high byte at low address end (IBM, HP)
mvsuid.DSTX.dsn	DST data, CERN FZ exchange	

Table 9.1: Naming Conventions for Experiment data in MVS

Tape files are written in the well-proven **ANSI label** format. Fully compatible to tape volumes written on VMS each data file on tape is preceded by a **header file** and followed by a **trailer file**, both containing control information about the data file. Correspondingly tape volumes to be read are assumed to be in AL format.

Tape files written by this package on AIX can also be read on VMS and - in case of fixed record length datasets not spanning more than one volume - also vice versa. Especially the whole contents of a tape written by a data acquisition in VMS can be transferred from the Unix tape copy station with one single command call to the central experiment data server in MVS! (For the available tape drives see section 4.9.2 on page 4-16 and table 4.9 on page 4-16).

Some useful tools for tape handling in general are described in section 4.9 on page 4-15. For more information see also the WWW page **Tape Handling in Unix at GSI**, which can be entered via the *Unix* page from the *Computing* home page.

9.1.5 Moving Experiment Data with FTP

The knowledge provided in this section is not needed to transfer binary data with fixed record length between AIX and the central data server in MVS. Such data should be copied with the `copymvs` command described above. However, to transfer data between other Unix platforms and the central server, or to copy data with e.g. **variable** record length, this section might be useful.

Files can be copied over the network with the `ftp` command (see chapter G.1 on page G-1). For experiment data the **binary** mode must be used. An ftp connection to MVS is initiated with the command

```
ftp mvs
```

Get Data from MVS

Such a session might look as follows:

Sample FTP Session

```
Name (mvs:goeri): RZ26
331 Send password please. ...
230 RZ26 is logged on.
Remote system type is OS/MVS.
ftp> binary
200 Representation type is IMAGE.
ftp> get lmdv.raw raw.lmd
...
ftp> quit
221 Quit command received. Goodbye.
```

As in this example, the MVS userid in the file name may be omitted, if you are connected with the corresponding account. If you require files from another MVS account than you are connected to, you have to specify the MVS userid. In this case the whole file name must be enclosed in quotes (''). Access is only granted if you have the proper read privileges in MVS. Then the `get` command in the ftp session might look:

Sample FTP Session

```
ftp> get 'xy01.lmdv.raw' raw.lmd
```

Data not accessed for some time are **migrated** by the HSM to the ATL. As the ATL is not directly accessible by users, in this case the data set is recalled to a staging disk in MVS before the transfer to your local node can take place, and in the example above a message appears:

Recall Message in FTP

```
150-Waiting for recall of dataset
RZ26.LMDV.RAW
```

Put Data to MVS

To transfer data to MVS, you should specify some suitable allocation parameters for the remote site with the ftp subcommand

```
quote site
```

GOOSY raw data, for example, are available in buffers of length 8kB in most cases. Then, for allocation in MVS, the following parameters should be specified:

- a *logical record length* of 8kB (parameter `lrecl`),
- a *block size* of 24kB (parameter `bl`),
- the *record format* fixed blocked (`fb`) (parameter `recfm`).

The data buffer size should be specified with the parameter `lrecl`. For best utilization of the staging disks in MVS, buffers should be added to blocks of size ~24kB or ~16kB.

Several parameters can be specified within one `quote site` command. Then an ftp session might look as follows:

Sample FTP Session

```
Name (mvs:goeri): RZ26
331 Send password please. ...
230 RZ26 is logged on.
Remote system type is OS/MVS.
ftp> binary
200 Representation type is IMAGE.
ftp> quote site lrecl=8192 bl=24576
200 Site command was accepted
ftp> quote site recfm=fb
200 Site command was accepted
ftp> put rawdata.lmd lmdv.rawdata
...
ftp> quit
221 Quit command received. Goodbye.
```

If the specified file already exists in MVS, it will be overwritten without further notification!

Experiment data files in MVS are recognized by the HSM due to their name structure (see section 9.1.3 on page 9-2). New experiment data in the staging disk pool will be migrated to the tape library ATL normally in the next night. However, in case of large transfer actions, this might be too late! Therefore, to avoid overloading of the staging disk pool, you should immediately migrate your files manually with the MVS command

```
hmigrate filename
```

From an AIX workstation you can submit the `hmigrate` command to MVS with

```
tsosubtcp mvs "hmigrate filename"
```

The GSI tools offered for data transfer to the central experiment data server do that **automatically!**

Background processing

Larger file transfer actions should be made in background with suitable shell scripts. To avoid the prompting for userid and password on the remote host, create a file named `.netrc` in your home directory and fill it in the following format:

Format .netrc File

```
machine mvs login userid password passwd
```

Please take care that only you can access the `.netrc` file by proper setting of the file permission bits (see command `chmod`).

If you write your ftp commands into a file, you can run an ftp session without further intervention by redirection of the input from standard input to this file:

```
ftp mvs < ftp-commandfile
```

9.1.6 Mounting the Server with NFS

Experiment data residing in MVS can be analyzed directly from a Unix workstation, if the files are **mounted**. However, the root privilege is required to mount a remote file system with the `mount` command.

In order to enable also normally privileged users the mounting of their experiment data, the command

```
mountmvs -u mvsuid -f filetype
```

has been developed on HP workstations. It requires no root privilege.

The parameter `mvsuid` specifies the MVS account (user identification). The corresponding MVS password is prompted for authorization.

The parameter `filetype` selects which experiment data will be mounted. Currently supported types are **lmdv** and **lmdi** (see table 9.1.3 on page 9-3).

If the local user `user` enters the command

```
mountmvs -u mvsuid -f lmdv
```

all MVS files with names

```
mvsuid.lmdv.dsn
```

are locally visible as files

```
/d/group/user/lmdv/dsn
```

(see section 9.1.1 on page 9-1. The directory **/d/group/user/lmdv** is called **mount point**. Files may be accessed (and displayed) with names *dsn*, if the current directory is the mount point, or as */d/group/user/lmdv/dsn*.

Some characters, such as the '\$' sign, have a special meaning in Unix shells. If a '\$' is contained in an MVS file name, the '\$' must be prefixed by the escape character '\\$' to switch off this special meaning.

The mount is set up with the attributes 'read only', 'fast-size', and 'noretrieve'. The latter means that data migrated to the Automatic Tape Library (ATL) cannot be recalled via this mount - else a simple `ls` command would initiate the recall of all files with the selected name structure! Instead the message '... *not found*' appears, if the required file(s) are migrated. If you need currently migrated files, you have to recall them in MVS with the command

```
hrecall lmdv.dsn
```

e.g. in a telnet session.

However, there are two restrictions you should keep in mind.

1. Due to limitations in HP-UX, the mount point must NOT reside on a disk connected to a HP cluster **client** (e.g. node rzhp9b).
2. Though the /d file system is common for the central HP and AIX workstations, the data mounted with `mountmvs` are visible only on the HP cluster, but not in AIX.

Unmount the Server

If no longer needed, you should remove your mount connection again with the command

```
umountmvs -f filetype
```

Again the parameter *filetype* selects the experiment data with qualifiers **lmdv** or **lmdi**.

Each user may obtain information about mounted file systems when invoking the `mount` command without parameters. Sometimes it may happen, e.g. if a mount connection is not used for some time, that you cannot access MVS files, even if the mount seems to be okay. In this case you most probably must authorize yourself again to the MVS system with the command

```
mvslogin mvs mvsuid
```

9.2 CERN Software

The CERN program library is a large collection of general-purpose programs maintained and offered in both source and object code from the CERN computer center. Most of the software was developed at CERN and is therefore oriented towards the needs of a physics research laboratory. Nearly all, however, are of a general mathematical or data-handling nature, applicable to a wide range of problems.

The library contains about 3000 subroutines and complete programs. 80% of the programs are written in Fortran77 and the remainder in C or assembly code.

At GSI computing center, the library is available on all supported Unix platforms, i.e. IBM AIX and HP HP-UX, as well as on VMS and MVS.

Documentation

Documentation is available in printed form or on-line via `xwww` on page *CERN Software* under section *Computing* of the GSI home page (<http://www.gsi.de/computing/cern.html>).

From this page you have also access to the most recent version of the manuals, release notes, product and usage descriptions and other actual information.

In the following, some major programs and packages (`paw`, `cmz`, `GEANT`) are shortly described. Furthermore, an overview of the organization of the CERN-library as well as instructions on how to link own programs with the library are given.

9.2.1 paw

`paw` is an interactive utility for visualizing experimental data on a computer graphics display. It may be run in batch mode, if desired, for very large data analyses or with a Motif-style user interface `paw++`. `paw` combines a handful of CERN Program Library packages that may

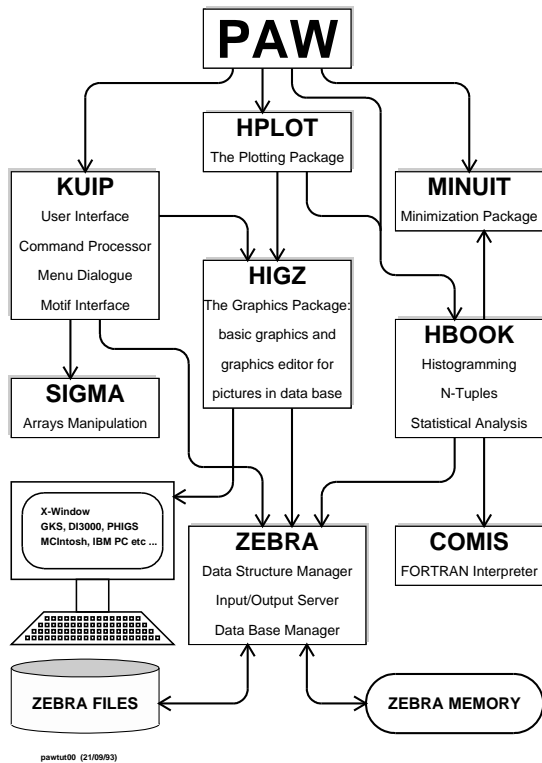


Figure 9.1: PAW and its components

also be used individually in users applications dealing with experimental data. Figure 9.1 shows the various components of paw.

The following list points to some typical paw applications (for more details, the PAW-manual should be consulted):

- Plot a vector of data fields for a list of events (Ntuples)
- Histogram of a vector of variables for a list of events
- Fit a function to a histogram
- Annotate and print graphics

9.2.2 PIAF, Parallel Interactive Analysis Facility

PIAF (**P**arallel **I**nteractive **A**nalysis **F**acility) provides a service for processing large Ntuples *in parallel* on the GSI cluster of AIX or HP Unix machines. Currently 11

AIX, 8 SP/2 AIX and 2 HP machines are available for PIAF.

Users store their data files in the /d filesystem (see section 3.1.1 on page 3-1) and then use them from within a PAW session running on their local machine or on one of the central machines. After connection to the PIAF main server, slave servers are started on the other machines of the same cluster and scan the Ntuple in parallel. For more details, issue HELP PIAF inside PAW or see <http://www.gsi.de/computing/piaf.html>

9.2.3 cmz

cmz is an advanced, interactive, fast, self-documenting, customizable, machine-independent and patchy compatible source-code management system with emphasis on FORTRAN and C source code. cmz is based on the same user interface package as PAW (KUIP). It allows to develop and transfer machine independent code for application programs.

The source-code management for almost all packages of the CERN program library is done with cmz, so it is a useful tool to develop user applications for different hardware platforms in the CERN environment. cmz runs on all supported UNIX flavors as well as on IBM/MVS, VM, VMS and other operating systems.

9.2.4 GEANT

GEANT is a system of detector description and simulation tools which should help the physicist in such studies. GEANT is most useful in the

- design and optimization of the detector
- development and test of the reconstruction and analysis programs, and
- the interpretation of experimental data.

In order to run GEANT, the user has to provide his own set of standardized subroutines describing the geometrical setup, the kinematics of the event and storing the results into the desired histograms.

GEANT can be run in interactive mode with graphics capabilities for program development and detector setup and, if settled, in batch mode for production of statistical data.

9.2.5 Organization of the CERN program library

The CERN program library is available in machine independent source (`car` file) and machine dependent source and binary form. Usually, CERN offers updates of the library 2-3 times a year. Three different versions of the CERN-library are stored on every machine:

- `pro`: production version
- `old`: old version, you can fall back to this version in case of incompatibilities
- `new`: newest version
- `95a`: sometimes the most recent version is also available, may be changed without notice. It is accessible by its real version number.

The CERN library is stored in a standardized place: You will find all files in the directory `/cern`. The organization of the subdirectories is shown in table 9.2. The libraries listed in table 9.2 have been installed in the `lib` subdirectory.

9.2.6 Usage of CERN-library programs

Initialization of CERN environment

Before you can use any program of the CERN library, you have to initialize the CERN environment. This can be done by the command

```
. cernlogin [new | pro | old ]
```

If none of the optional parameters is given, the `pro` version is selected.

If you like to know which release level of the CERN library is hidden behind the version `new` or `pro` or `old`, issue the command `cerninfo`:

cerninfo

The actual CERN Software Versions on `rzhp9b` are:

```
Version new is 93d
Version pro is 93d
Version old is 93c
```

```
Your current version set is: pro
```

You can initialize the CERN environment automatically during login. To do so, put the following line (as single line) into your `$HOME/.profile` file (it should be already in the file, just remove the comment sign `#` in front of the line):

cernlogin in .profile

```
[ -x /usr/local/bin/cernlogin ] &&
. /usr/local/bin/cernlogin
```

If you login on a HP machine via VUE-login, remember that the `.profile` file *is not* executed, but instead the `$HOME/.vueprofile` is executed. In this case, put the following line (as single line) into your `$HOME/.vueprofile` file (it should be already in the file, just remove the comment sign `#` in front of the line):

cernlogin in .vueprofile

```
[ -x /usr/local/bin/cernlogin ] &&
. /usr/local/bin/cernlogin >/dev/null 2>&1
```

Starting paw:

You can start `paw` after having initialized the CERN environment (see above) by typing

```
paw
```

```
Calling new version of paw-X11
```

```
*****
*
*      W E L C O M E      t o      P A W      *
*
*  Version 2.03/22      22 October 1993      *
*
*****
Workstation type (?=HELP) <CR>=1 : ?
List of valid workstation types:
  0:  Alphanumeric terminal
 1-10: Describe in file higz_windows.dat
 n.host: Open the display on host (1<n<10)
 7878:  FALCO terminal
 7879:  xterm

Workstation type (?=HELP) <CR>=1 : 5
Version 1.19/16 of HIGZ started
=> Start of system login Version pro
-----
.
.  (Various actual messages)
.
-----
=> End of system login. User login com-
mands now starting
*** Using default PAWLOGON file
"/dk2/rzhp9a/u/rz/dahlinge/.pawlogon.kumac"

PAW>
```

After the start-up, `paw` asks for the workstation type. A question mark (?) gives you a list of valid workstations. Usually you enter either 0 for no graphics or a

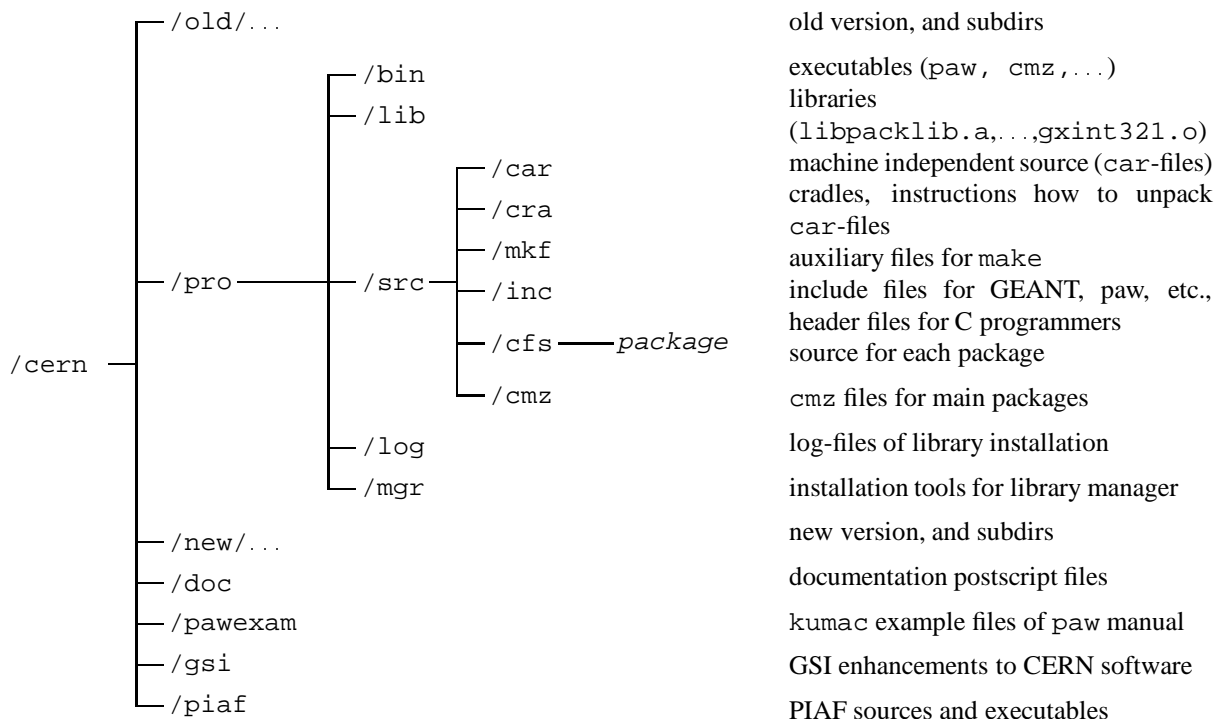


Figure 9.2: File system structure of the CERN program library

name	library	
KERNLIB	libkernlib.a	basic mathematical and general purpose routines
PACKLIB	libpacklib.a	main packages (CSPACK, EPIO, FFREAD, HBOOK, KAPACK, KUIP, MINUIT, ZBOOK, ZEBRA) and duplication of KERNLIB
GRAFLIB	libgraflib.a libgrafX11.a libgrafGKS.a	grafic packages (HPLOT, HIGZ), kernel X11-part of GRAFLIB GKS-part of GRAFLIB
PAWLIB	libpawlib.a	PAW-libraries (PAW, COMIS, SIGMA)
GEANT	libgeant315.a libgeant321.a geant315.o geant321.o	GEANT library version 3.15 GEANT library version 3.21 interactive main program GEANT 3.15 interactive main program GEANT 3.21
MATHLIB	libmathlib.a	general purpose subroutines + mathematical functions

Table 9.2: CERN libraries installed on /cern/version/lib subdirectories

number between 1 and 10. You can customize the initial appearance of your graphics window by editing the file `$HOME/higz.windows.dat`. Each of the 10 lines in this file corresponds to one workstation number and describes the position and size of the graphics window.

The following additional parameters can be given to the `paw-command`:

option	
<code>-v [new old pro]</code>	version, default is selected version of <code>cernlogin</code>
<code>-n</code>	disable automatic execution of <code>pawlogon.kumac</code>
<code>-b macroname</code>	batch mode, execute <code>macroname</code> , implies workstation type = 0

Starting `cmz`:

`cmz` is started after initialization of the CERN environment (see above). The following additional parameters can be given to the `cmz-command`:

option	
<code>-v [new old pro]</code>	version, default is selected version of <code>cernlogin</code>
<code>-n</code>	disable automatic execution of <code>cmzlogon.kumac</code>
<code>-l logon</code>	read logon commands from file <code>logon.kumac</code>
<code>-r [restore]</code>	restore environment of previous CMZ session. By default, <code>cmzsave.dat</code> will be read, if specified file <code>restore</code>
<code>-b macroname</code>	batch mode, execute <code>macroname</code> , implies also <code>-n</code>

Creating your own applications

To compile and link your own Fortran or C programs together with routines from the CERN-library, you can use standard Unix compile or link statements (see chapter 7 on page 7-1). The use of a makefile is strongly encouraged (see section 7.4.1 on page 7-2).

ATTENTION: You *must* specify the following compile options, depending on the operating system of your machine, otherwise no CERN library subroutine can be found by the linker:

option	OS
<code>-q extname</code>	AIX
<code>+ppu</code>	HP-UX
	other UNIX

The following examples show two Makefiles for creating

1. a GEANT program
2. a PAW application with own extensions

A GEANT example:

The following Makefile shows an example of how to create the interactive GEANT application `shower` from the user supplied subroutines `gudigi.f`, `gufldi.f`, `gukine.f`, These routines use `INCLUDE` statements to include the standard GEANT common blocks from the directory `/cern/pro/src/inc`.

The following Makefile can be used on an HP machine. You can copy this Makefile to create your own from `/cern/doc/Makefile.geant`.

```

GEANT Makefile HP-UX example:

# GEANT Makefile example
# M.D. 15/10/92, 17/1/94
#
# definition of symbols
#
LIB=/cern/pro/lib
INC=/cern/pro/src/inc
#
# definition of own modules
#
GEANTOBJ=gudigi.o gufldi.o guffgo.o gufld.o \
         gugeom.o guhbook.o gukine.o \
         gumate.o gustep.o
#
# Default action
#
.DEFAULT: shower
#
# Compiler options
#
FOPTS=+ppu
# disable FCOPTS when using softbench!
# for debug:
# FCOPTS=-g
# for production:
FCOPTS=-O
FFLAGS=$(FCOPTS) $(FOPTS) $(COMPFLAG)
#

```

```

# general rule how to compile .f
#
.f.o:
fort77 $(FFLAGS) -c -I$(INC) $<
#
# rule how to create program shower
#
shower: $(LIB)/gxint321.o $(GEANTOBJ)
fort77 $(LIB)/gxint321.o $(GEANTOBJ) \
`cernlib geant321,pawlib,graflib/X11, \
packlib,mathlib` \
$(FFLAGS) -o shower;

```

The following Makefile can be used on an HP machine. You can copy this Makefile to create your own from /cern/doc/Makefile.paw.

If you are running on an IBM RS/6000, the following lines have to be changed:

GEANT Makefile AIX example:

```

:
:
# FCOPTS=-g -q extname
# for production:
FOPTS=-O -q extname
:
:
.f.o:
xlf $(FFLAGS) -c -I$(INC) $<
#
# rule how to create program shower
#
shower: $(LIB)/gxint321.o $(GEANTOBJ)
xlf $(LIB)/gxint321.o $(GEANTOBJ) \
`cernlib \
:
:

```

For an explanation of the -I, -c, -l, -L, -o flags on the compile command, see the man pages and section 7.2 on page 7-1 of this primer.

A PAW example:

The following Makefile shows an example of how to create the interactive PAW application pawgr1 with user supplied commands. The new commands are defined in the command definition file (cdf file) user.cdf. They call action routines from condset.f, condshow.f,

Additionally, the user written subroutines are archived in a library libpawgr.a. Object files are archived in this library by the command ar.

The last entry of the Makefile shows how to create automatically the appropriate fortran routine for command definition – which has to be called in the PAW main program – from the cdf-file by calling the KUIP-compiler kuipc.

PAW Makefile HP-UX example:

```

# PAW Makefile example
# M.D. 15/10/92, 17/1/94
#
# definition of symbols
#
LIB=/cern/pro/lib
#
# definition of own modules
#
PAWOBJ= condset.o \
        condshow.o\
        conddraw.o\
        user.o

# If you use standard paw mainfile, use this:
PAWMAIN=$(CFS)/pawm/0pamain.f
# else your own:
# PAWMAIN=mypaw.f
#
# Default action
#
.DEFAULT: pawgr
#
# Compiler options
#
FOPTS=+ppu
# disable FCOPTS when using softbench!
# for debug:
# FCOPTS=-g
# for production:
FCOPTS=-O
FFLAGS=$(FCOPTS) $(FOPTS) $(COMPFLAG)
ARFLAGS=-crv
#
# general rule how to compile .f
#
.f.o:
fort77 $(FFLAGS) -c $<
#
# rules how to create program pawgr1
#
pawgr1: $(PAWMAIN) libpawgr.a
fort77 $(PAWMAIN) \
-L. \
-lpawgr `cernlib pawlib,graflib/X11, \
packlib,mathlib` \
$(FFLAGS) -o pawgr1

libpawgr.a: $(PAWOBJ)
echo "Loading libpawgr.a ..."
ar $(ARFLAGS) libpawgr.a $(PAWOBJ)
#
# create user.f from command
# definition file user.cdf
#
user.f: user.cdf
kuipc user.cdf user.f

```

If you are running on an IBM RS/6000, the same changes as above in the GEANT example have to be done.

For an explanation of the `-I`, `-c`, `-l`, `-L`, `-o` flags on the compile command, see the man pages and section 7.2 on page 7-1 of this primer.

The cernlib command

The `cernlib` command has been designed to ease the access to the CERN library. It is platform independent and can be used in linker commands and Makefiles. If used in linker commands e.g., it should be surrounded by backquotes (!)

cernlib command:

```
f77 -o mypaw +ppu -O mypaw.o
`cernlib pawlib,graflib/X11,packlib,mathlib`
```

9.3 Unix Clients for the Event Servers at GSI

At GSI client-server systems are available providing events over the network for physical analysis with PAW. Servers are available with the Single Branch System (SBS) running on CVCs (GSI-developed CAMAC-VME-Computer) with Lynx OS, or on Vax nodes (GOOSY event server). Therefore the events provided may be on-line from a data acquisition on a SBS or from a GOOSY session on a Vax.

The clients - currently offered for AIX and OpenVMS platforms - may request a certain number of events or a continuous stream of events from the server. By specifying filter conditions a strong selection on the events may be applied.

The clients have a user interface to create and fill HBOOK data structures (histograms and Ntuples) for further analysis and visualization with PAW. There are three different clients available providing the data structures in **shared memory** or in the **user's address space**:

1. a stand-alone program writing to shared memory,
2. a PAW module writing to shared memory (AIX only), and
3. a PAW module providing the data in the user's address space.

Obviously in the last two cases the client is integrated into PAW. However, a PAW executable can have only one of the two clients integrated.

For further analysis the HBOOK histograms and Ntuples are accessible in the third case only within the client PAW session. In case of shared memory usage, however, the data can be accessed in parallel also by any number of additional PAW sessions running on the same node as the client. This is described in more detail in section 9.3.3 on page 9-13.

Event analysis on the client side is performed in three Fortran interface programs defined similar to those of the APE package:

- **uastart**: create HBOOK histograms and Ntuples
- **uanal4**: fill HBOOK histograms and Ntuples from the incoming events
- **uastop**: user actions before end of analysis

The template files provided for these interfaces have to be filled by the user with analysis code. A further interface program named **uclinfo** is provided enabling the user to create his own individual event transfer statistics. Optionally a default statistics is provided.

More information on the clients is available in the WWW, beginning with the page **Event Server Clients** in the *Computing* home page.

9.3.1 Making Event Server Clients

The make files needed to create the AIX clients and template files of the user interfaces are available in the directory **/cern/goopaw**. All make files check and handle the version of the CERN software as specified in advance with the command

```
. cernlogin version
```

All three AIX clients work with the versions 94A and later of the CERN software.

The Shared Memory Client

The stand-alone client providing the HBOOK histograms and Ntuples in shared memory can be created with

- **gooshr.make**: makefile for client
- **uastarts.f**, **uanal4.f**, **uastop.f**:
template files for user interface

The executable is created in the current directory and named **gooshr**.

The Shared Memory PAW Client

The PAW module with the shared memory client integrated can be created with

- **goopawshr.make**: makefile for client
- **uastartc.f**, **uanal4.f**, **uastop.f**:
template files for user interface

The PAW executable is created in the current directory and named **goopawshr**.

The PAW Client without Shared Memory

This PAW client collects the data only in the user's address space:

- **goopaw.make**: makefile for client
- **uastartc.f**, **uanal4.f**, **uastop.f**:
template files for user interface

The PAW module created in the current directory is named **goopaw**.

Note that there are two different template files containing the initialization subroutine, namely **uastarts.f** for the stand-alone client, and **uastartc.f** for the two PAW clients. These file names are also used in the corresponding make files. However, the name of the corresponding Fortran subroutine must be **uastart** in all cases.

The templates for the other user interfaces are the same for all clients. To create your client proceed as follows:

1. Select the client which is best suited for your purposes.
2. Then copy the corresponding templates to your file system and fill them with your analysis code.
3. In case of a shared memory client replace the name of the shared memory area (SHARE) in **uastart** by your own individual name.
4. If not already done, select the version of the Cern software you want to work with:

```
. cernlogin version
```

5. Finally create your executable by invoking the make file:

```
make -f /cern/goopaw/makefile
```

9.3.2 Starting the Clients

Starting a PAW Client

After successful execution of the corresponding makefile start your PAW session with the command **goopawshr** or **goopaw**, respectively. Then in both cases the client is started with:

```
PAW > /gsi/goocli/input parameterlist
```

```
From /GSI/GOOCLI/...
```

```

1: * INPUT           The command to connect to server and to request events.
2:  GOOCLI           Event server client - an overview
3:  FILTER_DESC      Some general features of the event filter.
4:  FILTER_SPEC      Detailed description of filter specification
5:  FILTER_EXAM      Filter examples for interactive or file input
6:  UASTART          User interface to initialize the analysis.
7:  UANAL4           User interface to perform the analysis.
8:  UASTOP           User interface to terminate the analysis.
9:  UCLINFO          Provide information on event transfer.
10: SHR_MEMORY       The stand-alone client utilizing shared memory.
11: IMPLEMENT        Implementation details on the event server clients.
```

Figure 9.3: Help Menues in PAW for Clients of Event Server

For the parameter list see the AIX man page `goocli` or the on-line help information in PAW:

```
PAW > help /gsi/goocli
```

Several help menus are offered on the command interface of the client, on the event filter (including extensive examples), on the user interfaces, and on some general informations, including also some hints on the stand-alone client writing to shared memory (see figure 9.3.2 on page 9-13).

By default the HBOOK data structures of the PAW clients are initialized only once, so that all events received from all client invocations are accumulated. However, optionally the data structures may be initialized with each call of the client (parameter *imode*) thus loosing their previous contents.

Starting the stand-alone Client

Start the stand-alone client with

```
goopaw parameterlist
```

Except for the parameter *imode*, the parameter list is identical to that of the PAW clients (see the AIX man page `gooshr`). Contrary to the case of the PAW clients the HBOOK data structures are initialized with each invocation of a stand-alone client thus loosing any previous contents.

9.3.3 Handling Shared Memory

The HBOOK histograms and Ntuples filled by the shared memory clients can be accessed for visualization and further analysis in parallel by any number of additional PAW sessions running on the same node. Data access in such an additional PAW session might look as follows. At first map the shared memory segment:

```
PAW > global shared-memory-segment
```

The name of the shared memory segment must be identical with that used in `uastart`! To see what happens in your client session you may prepare some display in a kind of refresh mode:

```

┌───────────────────────────────────────────────────────────────────────────────────┐
│                                     Display with Refresh I                                     │
└───────────────────────────────────────────────────────────────────────────────────┘
PAW > zone 1 3
PAW > idle 1 'h/pl 1 ; h/pl 2 ; h/pl 3'
```

In this example, the display window is prepared for three displays. The last PAW command `idle` means that if nothing is typed at the keyboard for 1 second, you will see new plots of IDs=1,2,3. You can type at any time at the keyboard to interrupt this sequence. To stop this sequence, type

```
PAW > idle 0
```

If a big number of events is provided, you can see your histograms growing in time. You can do this also with overlays:

```
IPC status from /dev/mem as of Mon Jan 30 12:58:01 CET 1995
T   ID   KEY           MODE           OWNER   GROUP   CREATOR   CGROUP   SEGSZ
Shared Memory:
m   4096 0x0d050222  --rw-----   root   system   root     system   64480
m     1 0x444d4131  --rw-----   root   system   root     system 2138112
m     2 0x0d074192  --rw-rw-rw-   root   system   root     system   1440
m  16387 0x53484152  --rw-r-----  goeri    rz      goeri    rz8000000
```

Figure 9.4: List of Shared Memory Segments on an AIX node

Display with Refresh II

```
PAW > h/pl 1 k ; h/pl 2 k ; h/pl 3 k
PAW > idle 1 'h/pl 1 u; h/pl 2 u; h/pl
```

CAUTION: You should replace the name of the shared memory area (*share* in the templates) by your own individual name, because in the current CERN software implementations any user on the same node can read from this section, if he knows its name and maps it!

Deleting Shared Memory

The shared memory segments are **permanent** and keep existing even if you logout! If no longer needed, look for the segment-id of your shared memory segment and remove the area:

Remove a Shared Memory Segment

```
ipcs -mbc
ipcrm -m segment-id
```

The first command provides a list of all shared memory segments existing at that time on the current node. See figure 9.3.3 on page 9-14 as an example.

In this example, the user *goeri* owns a shared memory segment of size 8 Mbyte and with segment-id 16387. This segment may be removed by the owner with the command

```
ipcrm -m 16387
```

To **change the size** of your shared memory segment, it **MUST** be deleted first (and then be recreated)! If you want to create a new, like-named segment of the same size, you need not delete the old one, but of course the old contents is lost!

9.4 SATANGD on Unix

With the end of MVS life-time at GSI some powerful and heavily used program packages developed at GSI will no longer be available. Examples are the SATAN and SATANGD systems for analysis and plotting of experiment data.

In order to preserve the functionality of SATANGD also for the future, a user has written a program package which is data- and command-compatible to SATANGD under MVS. This new version of SATANGD runs on various Unix systems and is available as public domain software on an 'as-is' basis also outside of GSI.

SATANGD is a data plotting program package specially suited for the needs of the scientists at GSI. It provides all tools necessary for the creation of high quality graphical pictures for scientific publications and presentations.

The new package not only saves these powerful tools, but also allows the continued usage of all SATANGD data files produced on MVS in the rapidly expanding Unix world. Additionally, as the new SATANGD is based on a SATAN-style command decoder, all old command lists for the creation of plots can be reused on Unix.

There are only few functions of the MVS version that do not yet work in Unix. Besides that, some 'MVS-style' features have been converted to 'Unix-style'. In addition, new useful features have been added that are not available in the original MVS version. As main enhancement, the Unix version of SATANGD supports part of the powerful SATAN **fit package**.

More information including the complete SATANGD documentation can be found in the WWW, beginning on the page **SATAN**, which may be entered from the *Computing* home page. Here details are available on

- general usage,
- implementation,

- things not available in the Unix version, and
- enhancements of the Unix version.

On the central Unix workstations of GSI, SATANGD can be started with the command

```
gd [version]
```

Supported versions are *new*, *pro* (default) and *old*. For a short demonstration of some SATANGD capabilities, enter the SATANGD command

```
gt
```

which starts a tutorial.

Appendix A: Telnet support for MVS (3270) terminals

The Telnet support for terminals directly attached to the MVS mainframe has been realized by the UCLA **Xtelnet** client as part of a C/Sockets packet, which has mainly been developed by the University of California in Los Angeles.

Xtelnet is a TSO/ISPF based Telnet client, which allows to access remote telnet hosts using vt100 terminal emulation. It provides 3270 terminal emulation and features full data logging/capture facilities for data from telnet sessions.

Xtelnet must be invoked from within ISPF, either from ISPF panel C (GSI COMMUNICATION FACILITIES), option X or as command from ISPF's TSO/CMD entry panel 6 or from ISPF Command line via the TSO command.

The command syntax is:

```
xtnetnet <options> <host> <port>
```

Valid options are: -l - set telnet EOL to <lf> instead of the default <cr><lf> -v - don't negotiate '3270' terminal type, use 'vt100' only -2 - in tn3270 negotiate a model-2 (24x80) term-type only -b - disable receipt of TSO broadcast/alert messages (11)

If invoked without parameters or from ISPF panel C.X, an ISPF Dialog panel is offered to drive execution of Xtelnet. An extended help text can be accessed by pressing the <PF1> button.

The following tables show the PF key mappings for the two supported emulation types.

VT100 Function Key Mappings pa1 Command Key pa2
Reshow 1 Help 2 Hide 3 Ctl/"c 4 Esc 5 Kill("u) 6 Eof("d)
7 Up 8 Down 9 Input 10 Left 11 Right 12 Asis/' ' 13 F1
14 F2 15 F3 16 F4 17 Home 18 End 19 PgUp 20 PgDn
21 Ispf 22 Log 23 Snap 24 Review

3270 Function Key Mappings pa1 Command Key pa2
Reshow pa1 PA1 1 Help 2 PA2 3 Quit 4 IP 5 AO 6 AYT
7 BRK 21 Ispf 22 Log 23 Snap 24 Review

In VT100 emulation mode, the <PF2> key can be used to hide subsequent input, e.g. the login password. The <PA1> command key offers some VT100 control commands:

```
..... vt100 Command Keys .....  
..... | 1 Help 3 Quit 4 IP 5 AO 6 AYT 7 BRK  
| | Press ENTER to continue Xtelnet, or END to quit  
Xtelnet | '.....  
.....'
```

A.1 Appendix

Printer	Model	Style	Print Queue	Location
pshpa	HP LaserJet IIIsi	B/W ASCII PostScript single-sided natural paper double-sided natural paper single-sided plain white paper double-sided plain white paper	pshpa pshpad pshpaab pshpadab	Output Room RZ 2.223
pshpb	HP LaserJet IVsi	B/W ASCII PostScript single-sided natural paper double-sided natural paper single-sided plain white paper double-sided plain white paper	pshpb pshpbd pshpbab pshpbdab	Graphics Room
psaa	DEC LPS20 Printserver	B/W ASCII PostScript single-sided plain white paper double-sided plain white paper A3 single-sided plain white paper	psaa psaad psaaa3	Output Room RZ 2.223
psag	DEC LPS17 Printserver	B/W ASCII PostScript single-sided natural white paper double-sided natural white paper	psag psagd	Output Room RZ 2.223
p41	QMS Magicolor	Color PostScript Paper Transparencies	p41wcs p41fcs	Output Room RZ 2.223
psteka	Tektronix Phaser 3 PXi	Color PostScript A3 with separate tray	psteka	Output Room RZ 2.223
pslexa	IBM Jetprinter PS 4079	Color PostScript Transparencies	pslexa	Output Room RZ 2.223

Table A.1: List of available Printers

pshpc	HP LaserJet IVsi	B/W ASCII PostScript single-sided double-sided single-sided double-sided	pshpc pshpcd pshpcab pshpcdab	Detektorlabor
pshpd	HP LaserJet IVsi	B/W ASCII PostScript single-sided	pshpd	Lepton-Container C01 204
psab	DEC LPS17 Printserver	B/W ASCII PostScript single-sided double-sided	psab psabd	1.222a
psac	DEC LPS17 Printserver	B/W ASCII PostScript single-sided double-sided	psac psacd	Geschäftsführung
psad	DEC LPS17 Printserver	B/W ASCII PostScript single-sided double-sided	psad psadd	Atomphysik vis-a-vis 2.294
psae	DEC LPS17 Printserver	B/W ASCII PostScript single-sided double-sided	psae psaed	KPIII 4.174
psaf	DEC LPS17 Printserver	B/W ASCII PostScript single-sided double-sided	psaf psafd	KPII vis-a-vis 4.141
ln03r_a	DEC LN03R	B/W ASCII PostScript	ln03r_a	Elex Lab 2.252
ln03r_b	DEC LN03R	B/W ASCII PostScript	ln03r_b	KC1 Lab LN03 2.124
ln03r_d	DEC LN03R	B/W ASCII PostScript	ln03r_d	Cave-B Meßhütte
ln03r_e	DEC LN03R	B/W ASCII PostScript	ln03r_e	Cave-A Meßhütte
ln03r_f	DEC LN03R	B/W ASCII PostScript	ln03r_f	HLI
ln03r_g	DEC LN03R	B/W ASCII PostScript	ln03r_g	Container Z2
ln03r_h	DEC LN03R	B/W ASCII PostScript	ln03r_h	Prof. Metag 4.102
ln03r_i	DEC LN03R	B/W ASCII PostScript	ln03r_i	2250 HKR 1.121
ln03r_j	DEC LN03R	B/W ASCII PostScript	ln03r_j	Cont.01 204
ln03r_k	DEC LN03R	B/W ASCII PostScript	ln03r_k	Labor 2.103
ln03r_l	DEC LN03R	B/W ASCII PostScript	ln03r_l	Labor 3.001
ln03_a	DEC LN03	B/W ASCII	ln03_a	Elex Lab 2.252
ln03_c	DEC LN03	B/W ASCII	ln03_c	KP3 3rd Fl. 4.174
ln03_d	DEC LN03	B/W ASCII	ln03_d	AP1 1st Fl. 2.294
ln03_e	DEC LN03	B/W ASCII	ln03_e	KP2 3rd Fl. vis-a-vis 4.141
ln03_g	DEC LN03	B/W ASCII	ln03_g	SHIP
ln03_j	DEC LN03	B/W ASCII	ln03_j	Cave-B Meßhütte
ln03_k	DEC LN03	B/W ASCII	ln03_k	FRS Meßhütte
ln03_l	DEC LN03	B/W ASCII	ln03_l	ESR Meßhütte
ln03_m	DEC LN03	B/W ASCII	ln03_m	Cave-C KaoS Meßhütte
ln03_n	DEC LN03	B/W ASCII	ln03_n	Cave-A Meßhütte
ln03_p	DEC LN03	B/W ASCII	ln03_p	Sicherheit & Strahlenschutz
ln03_q	DEC LN03	B/W ASCII	ln03_q	KP2 2nd Fl. behind ESR

Appendix B: Logging in from MVS via Telnet

Access to a Unix workstation via telnet is also possible from a 3270 terminal under MVS. However, the login procedure is a little bit complicated, and you can only work in line mode. As the invocation of full screen applications (e.g. the editors vi or emacs) may hang up your session, this access is not recommended in general. If your session hangs up, hit one of the buttons <F4> to <F12>. Then the prompt

Telnet command:

appears, and you can finish the current telnet session with the `close` command.

For example, to access from your 3270 terminal the RS/6000 workstation rzri6f, enter the TSO-command

```
telnet rzri6f
```

Now TSO confirms the action:

```
TCPTELO001I MVS TCP/IP Telnet V2R1
TCPUTM110I Connecting to RZRI6F
                140.181.64.35, port TELNET (23)
***
```

Press , and the following information appears:

```
TCPUTM118I
TCPUTM119I Using Line Mode...

TCPUTM120I
TCPUTM121I Notes on using Telnet
            when in Line Mode:
TCPUTM122I - To hide Password,
            Hit PF3 or PF15
TCPUTM123I - To enter Telnet Command,
            Hit PF4-12, or PF16-24
***
```

After pressing , a new screen appears. The top line is

```
AIX telnet (rzri6f)
```

The bottom line is

Telnet command:

From now, the bottom line is the command line for the duration of your Telnet session. At this time, however, you must **not** enter a Telnet command, but just press . The top line remains unchanged, and the login prompt appears:

```
login:
```

Enter your account name at the current cursor position, and the password prompt appears.

Password:

To hide your password, enter at first <F3>, as indicated in the information screen above, and then type your password. Be careful when entering your account name and password, because Unix is **case sensitive!** Upper and lower case letters have a different meaning!

If you have a valid account on rzri6f, the system greets you with a welcome message and provides some information, such as the the existence of unread news or new mail files. When prompted for your terminal type, enter 3270. Because your terminal has no display address, just press when asked for your display address. Then the system responds with the corresponding default prompt as in the example above and is ready for command input.

After some command inputs, the bottom line of your screen shows the string **Holding** on the right, and further inputs are not possible until you press the **PA1** key.

There may be applications that can not work correctly in this environment, because they require hardware features not available. However, many Unix commands can be used, e.g. to compile and run user programs, or to look into the file system.

Appendix C: How to use the Tape Stacker

A stacker containing ten slots for 8mm Exabyte tape volumes and an EXB-10e Exabyte tape drive is available at GSI. Together with other tape drives the stacker is located in the graphics room of the computing center in the ground floor of the southern building. The tape stacker consists mainly of

1. a data cartridge holder for up to ten 8mm Exabyte tape volumes,
2. a small robot called **Cartridge Handling Subsystem** (CHM), and
3. an 8mm Exabyte tape drive called **Cartridge Tape Subsystem** (CTS).

The ten slots for tape volumes in the stacker can be accessed **in sequential order** from bottom to top via the special device file of the tape drive. With the command

```
mt -t devicefile offl
```

the current Exabyte volume is unloaded and put back to the stacker slot where it previously resided. Then the robot checks the slot(s) above and puts the next Exabyte volume found into the drive.

You should keep in mind that **only the robot** serves the Exabyte drive - if you try to help him you will have a problem! You have to fill the slots in the holder with tape volumes, and you have to remove the volumes again when finished. But only the robot puts the volumes into the tape drive and removes them again, and only the robot opens and closes the door of the drive!

If unused, the tape stacker is normally in the following state:

- all slots are empty
- the door of the drive is open
- the status indicated on the LCD screen is either **Wait for restart**, or **Tenpack detected empty during seq.**

To resolve abnormal states, see below. If in normal state, proceed as follows:

1. Open the door of the stacker by pushing the door button.
2. Put your Exabyte volume(s) into the slots - beginning from the bottom. **Don't put them into the drive.** The robot insists in doing it for you!

3. Close the door of the stacker again (but not the door of the drive!).
4. Press the RESET button at the operator panel.
5. Confirm the reset by pushing the ENTER button.

Now the robot checks the slots from the bottom up for the first Exabyte volume, puts it into the drive, and closes the door of the drive. Afterwards, the LED screen indicates the status **Wait for CTS**, and the system is ready to receive your tape command(s).

After the last Exabyte volume is handled, unload it with the command `mt` (see above). The robot, of course, searches then again for the next Exabyte volume. As there is no one, the robot arm waits at the top, and the LED screen indicates the status **Wait for restart**. Now remove your Exabyte volume(s) from the holder and close the door of the stacker (but not of the drive!).

Warning: Be patient when handling Exabyte tapes under Unix. Exabyte initialization is rather slow, and an unload takes more time than you can imagine!

If there is no status indicated on the LED screen, press the MENU button at the operator panel. Move the arrow with the arrow buttons to MAIN MENU and press ENTER. Then the status will be indicated. If pressing the MENU button has no success, try the ESCAPE button or the RESET button (followed by ENTER for confirmation) before.

If the status message indicates an error state, press the RESET button and the ENTER button for confirmation. If the error remains, please contact the Benutzerberatung (tel 2555).

If the stacker is available (indicated by the command `tape`), but if there are still some Exabyte tape volumes within the holder, remove them. If a volume is (also) within the drive, unload it with the command `mt`. Deposit the Exabyte volume(s) near the stacker - your predecessor who forgot them will appreciate it.

Appendix D: GNU Software

GNU stands for **Gnu's Not Unix** and is the name for the complete Unix-compatible software system developed by the Free Software Foundation. Some large parts of this system are already working, and are distributed now.

The Software is distributed as a 'Free Software'. The word 'free' stands here for freedom and not to price. To get the GNU Software you may pay or may not. But in contrast to commercial Software, you have the freedom to copy the program and give it to your friends and co-workers and you have the freedom to change the program as you wish, by having the full access to the source code. Furthermore, you can study the source and learn how such programs are written.

The main work of the Free Software Foundation is concentrated on the development of new free software, working towards a complete GNU system.

Beside developing GNU, FSF distributes copies of GNU software and manuals, and accepts tax-deductible gifts to support GNU development. Most of the FSF's funds come from its distribution service.

Several parts of the GNU Software (about 120 programs) are available for UNIX, DOS, and a lot of other operating systems.

Interesting GNU programs for a UNIX system are:

emacs	full-screen editor (Chapter 5.2)
gcc	the GNU C/C++/Objektiv-C Compiler
gdb	GNU C,C++ Debugger
GNU C Lib	POSIX.1 C library
libg++	C++ class library
bison	GNU advanced yacc
make	GNU advanced make
indent	C reformatting program
RCS	Revision Control System
CVS	Concurrent Version System
patch	apply diffs (patches) to files
gnuchess	a chess playing program (X11)
Ghostscript (gs)	a Postscript interpreter
ghostview	X11 user interface for Ghostscript
gunzip	uncompress utility for .gz files
gzip	compress utility using Lempel-Ziv coding
fontutil	fonts for Ghostscript or T _E X
gnuplot	interactive plotting program
Texinfo	structured documentation system, produces on-line help and printed documents

ispell	advanced spell checker (also for T _E X)
less	a better pager
bash	GNU's Bourne Again Shell
GNU tar	GNU (Tape) archive
patch	applies diff files

We have installed some of the GNU programs in the `/usr/local` filesystem on the central AIX and HP-UX file servers, so that the programs are available via NFS.

Nearly all of the above listed programs are installed on the central AIX server.

For the most programs exist a man page or a Postscript file of the reference manual in `/usr/local/doc/gnu`.

All the software and publications from the Free Software Foundation are distributed with permission to copy and redistribute. If you are interested in a copy you can get the latest software via anonymous FTP (program: 'ftp', user: 'anonymous', password YOUR NAME, mode 'binary') from prep.ai.mit.edu (18.71.0.38) (Directory: `/pub/gnu`). This ftp server is in Cambridge. To reduce the transfer time and costs you should take a ftp server at your side. Two good candidates are ftp.th-darmstadt.de (130.83.55.75) or rusmv1.rus.uni-stuttgart.de (129.69.1.12).

If you find a GNU program of common interest, and you think it should be installed on a central file server, let use know about it.

From our own experience: Please restrict your self in collecting programs from ftp servers. The disk space is restricted and it makes no sense that every body has the same programs on his user disk.

Appendix E: Motif Windows

With a mouse, a lot of functions can be performed with windows:

1. You can accomplish **geometrical** operations, such as moving or resizing.
2. The contents of windows can be **scrolled**.
3. Windows can be **iconized**, that means, they can be converted to a small picture called **icon**, which is a representation of an inactive window, and they can be restored again into the state of an active window.

Table E.1 summarizes the operations possible with Motif windows. The terms used there are explained in the following.

The remainder of this paragraph describes in detail, how these operations can be performed.

Standard GSI Terminal Windows

A standard GSI terminal window can be obtained with the command `xt`. It is an Motif window and has an outer **frame** and a horizontal **title bar** on top and a vertical **scroll bar** at the right, but both within this frame. They consist of several parts with each of them enabling specific functions. The mouse cursor, which has the shape of an 'I' when positioned within the text field, changes its appearance when moved to one of these elements. The new shape depends on the specific location.

Motif Window Frame

The Motif window frame - as a rectangle - consists of two horizontal and two vertical bars, and of four corners. When moved to one of the bars, the mouse cursor appears as an arrow pointing outwards to a line. This shall indicate that the corresponding border of the window can be moved. To do it, press the left mouse button and move the mouse cursor while keeping the button pressed: The window is resized by **dragging** this border, whereas the other borders keep their location.

When moved to one of the corners, the mouse cursor appears as an arrow pointing outwards to the corresponding window corner. In the same way as with the bars, the window can be resized by dragging this corner, whereas the diagonal opposite corner keeps its location.

Title Bar

The title bar is the horizontal bar on top of the screen, just inside the window frame. It consists (from left to right) of the window menu button, the title area, the minimize button, and the maximize button.

The Title Area If you move the mouse cursor to the long horizontal field of the title bar, the cursor changes its shape to a fat arrow. Now, with the same mouse procedure as described above, you can move the window as a whole without changing its size.

The Minimize Button The minimize button can be used to convert a window to an icon. The button is identified by a very small square. To **iconify** a window might be useful in case of programs that can run unattended in background, or in case of applications used only occasionally. It preserves screen space by reducing the window stack.

The Maximize Button The maximize button can be used to convert an icon or a window to a new window covering the whole screen. It is identified by a big square in its center.

The Window Menu The window menu button is located leftmost within the title bar and identified by a narrow rectangle in its center. If activated with the left mouse button, either by clicking or pressing and holding down, the window menu appears as popup menu. It can be used for the handling of windows and also of icons - the window menu also pops up if you click an icon. The functions not available are printed in a lighter typeface (minimize and size in the case of an icon). All functions available with the mwm window frame and the other parts of the title bar are also available with the window menu.

If a window menu function is activated by click with the left mouse button, it can be performed just by shifting the mouse - no more button needs to be pressed during mouse movement. The following functions are available in the window menu:

- **Restore:** Icons, or windows covering the full screen ('maximized'), are converted back to a window.
- **Move:** Move window or icon with the mouse.
- **Size:** Resize window. If you move the mouse to one of the four corners of the window frame, the corresponding corner points are shifted. If you move the mouse to one of the four vertical or horizontal borders, the corresponding borders are shifted.
- **Minimize:** Convert window to icon.

Operation	window part	action
resize window	window frame	drag vertical or horizontal bar (top or bottom)
	window frame	drag corner (one of four)
	window menu	click size button, shift frame bar
	window menu	click size button, shift frame corner
move window	title bar	drag title area
	window menu	click move button, shift window
iconize window	title bar	click minimize button
	window menu	click minimize button
maximize window	title bar	click maximize button
	window menu	click maximize button
restore window from icon	window menu (icon)	click restore button
lower window in stack	window menu	click lower button
close window	window menu	click close button
move icon	window menu (icon)	click move button, shift icon
maximize icon	window menu (icon)	click maximize button
lower icon in stack	window menu (icon)	click lower button
close icon	window menu (icon)	click close button

Table E.1: Summary of Operations with Motif Windows and Icons

- **Maximize:** Convert window or icon to window covering the complete physical screen.
- **Lower:** Put window or icon to the bottom of the window or icon stack.
- **Close:** The window will be closed (and disappears).

The window menu functions can also be invoked via accelerator keys, e.g. **Alt** - **F9** for Minimize. This works with or without an activated window menu. If the window menu is already activated, you can also enter a unique abbreviation (the underlined letter), e.g. 'n' for Minimize. The required key combinations and abbreviations are indicated in the window menu.

The Scroll Bar

In your terminal windows you are not limited to the displayed lines of text. When created with `xt`, up to 200 lines of text are saved, and you can use the scroll bar at the right to scroll through the saved text.

Figure F.1: Present status of the Hardware installed at GSI.

F.1 Managing your X-Terminal by XDM

An easy way to manage the sessions on your X-terminal is by means of the standard XDMCP, the X Display Manager Control Protocol. The X Display Manager manages a collection of X displays on remote servers. It provides services like prompting for login name and password, authenticating the user, and running a "session". A session is defined by the lifetime of the session manager in the X11 window environment. When the session manager is terminated, xdm resets the X server and restarts the whole process.

The X terminals offer a menu of possible hosts that run the xdm display manager. The GSI hosts supporting XDMCP are :

- AIX operating systems : rzri6f, clri6a
- HP-UX operating systems : rzhp9a, rzhp9b
- ULTRIX operating system : ds5a

When the user logs in, the XDM reads the user profile in the user's HOME directory and tries to start the session manager on the predefined display. The user profile therefore must define the DISPLAY name because at this point there's no possibility to respond to any questions.

When the display is initialized, xdm reads in .xsession file in the user's HOME directory to open the window environment. In case there is no .xsession file in the user's HOME directory xdm looks for .xinitrc in the user's HOME filesystem and executes it. If there is no .xinitrc file as well then the system's default file will be used.

Sample .xsession file

```
#!/bin/ksh
#
# FUNCTIONS: .xsession
#
# aixterm window to open

aixterm &

# xclock with chime and update every second
xclock -0+0 -chime -update 1 &

# to execute the local mwm on the Xterminal

xpsh -display $DISPLAY -xpenv /
  ''SETUP_DISPLAY=$DISPLAY'' launcher
xpsh mwm

# to execute remote mwm

# exec mwm
```

Appendix G: Introduction to Internet Services

G.1 About Internet

The Internet is a global network of networks that provides access to hundreds of thousands of computers around the world. As the reach of the network has grown, so has the number of services accessible. The main tools that allow the user to navigate through the Internet, are

`telnet` to access remote hosts,
`ftp` to retrieve data files.

Anonymous FTP will be briefly described next. For `telnet` see chapter 4.6 on page 4-11.

G.1.1 FTP

FTP stands for "file transfer protocol" and is the method used to transfer files over the Internet. "Anonymous" ftp means that one can login to the remote system using the userid of "anonymous" and password of either "guest" or usually your own userid and internet address. Ftp is like telnet in that the "open" command and access to the remote host is similar.

A typical session might go as follows:

```
$ftp any.host.i.know
login:anonymous
guest login ok...send user id as password
ftp>ls -al (list all files)
ftp>cd pub (change to the " pub" directory)
ftp>get my.file
transfer complete
ftp>quit
$
```

The standard transfer protocol is ASCII. This is suitable for text. Use command `binary` if transferring program or image files. (Note: on VAX-VMS computers use `IMAGE`).

Large files are usually "tared" and compressed. You have to use binary FTP to get such files. The file extension shows how to uncompress it:

```
.tar tar -xvf myfile.tar
.Z uncompress myfile.Z
.tar.Z uncompress myfile.tar.Z
tar -xvf myfile.tar
```

G.1.2 Internet addresses

There are two forms that express an Internet address, an alphabetic name, or a series of numbers. The alphabetic version is called the "domain name system" and

the numeric the "numeric name system". Sometimes a local network will not be up-to-date with additions to the domain names and an address may not work. If this happens, try the numeric address before giving up. Sometimes the numeric name system address will be changed without notice and in that case the alphabetic domain name should be tried.

G.2 Internet Services

G.2.1 Overview

With a little practice, the above-described functions (`ftp`, `telnet`) will be simple and open the electronic door to the global reach of the Internet. An introduction to the Internet services can be found in [10]. A comprehensive listing of services is given in [14]. Its table of contents is listed below:

1. Library Catalogs & Campus Information Systems
2. Databases
3. Electronic Discussion Groups/Forums
4. Directories
5. Information Resources
6. FTP Archives
7. Fee-Based Information Services
8. Software/Freeware
9. Bulletin Board Services
10. Miscellaneous

G.2.2 archie

One of the most useful Internet services, acquisition of public domain software, can be the most frustrating. There are now hundreds of servers with thousands of software titles spread throughout the Internet. Often the searcher knows that the needed software is somewhere out there but finding the software title through this maze can take a long time. After checking 10 or 20 host sites, one is tempted to give up. Archie is a unique system devised to make locating software on public archives simple. Instead of searching the remote hosts one at a time, the user can enter the search on "archie" and find out where copies exist across 712 (at this writing) hosts.

The results of the search may be viewed online or sent automatically via e-mail for later viewing. Search results identify host domain name and IP address and the exact path and filename to the requested file making it easy to ftp. The search engine has many powerful features to aid in retrieving those hard-to-find titles.

Access: telnet quiche.cs.mcgill.ca login as archie. Important commands are:

help	a list of all commands
help command	description of command
quit	exit archie
whatis	search for keyword in the software description database
prog	search the database for a file

For example `prog xclock` will cause archie to search all the archives for the string "xclock". At the end of the search, archie will present the results back to the screen.

xarchie

If you have access to an X-window terminal, you can use `xarchie` for a menu-guided search through the public-domain ftp sites. For a more detailed description use man xarchie.

G.2.3 NetNews

Usenet is the set of people who exchange articles tagged with one or more universally-recognized labels, called "newsgroups". The groups distributed worldwide are divided into seven broad classifications: "news", "soc", "talk", "misc", "sci", "comp" and "rec". Each of these classifications is organized into groups and subgroups according to topic.

- comp** Topics of interest to both computer professionals and hobbyists, including topics in computer science, software source, and information on hardware and software systems.
- sci** Discussions marked by special and usually practical knowledge, relating to research in or application of the established sciences.
- misc** Groups addressing themes not easily classified under any of the other headings or which incorporate themes from multiple categories.
- soc** Groups primarily addressing social issues and socializing.
- talk** Groups largely debate-oriented and tending to feature long discussions without resolution and without appreciable amounts of generally useful information.

news Groups concerned with the news network and software themselves.

rec Groups oriented towards hobbies and recreational activities.

To start a NetNews reader with X11 interface type `mrxrn` on the IBM or HP workstations.

Bibliography

- [1] Adobe Systems Incorporated, *The PostScript Language Reference Manual*, Addison-Wesley, (1990), ISBN 0-201-18127-4
- [2] D. Cameron and B. Rosenblatt, *Learning GNU Emacs*, O' Reilly & Associates, Inc, Sebastopol, USA, (1991)
- [3] J.D. Foley, A. van Dam, S.K. Feiner, J.F. Hughes, *Computer Graphics: Principles and Practice*, Addison-Wesley, Amsterdam, (1990), ISBN 0-201-12110-7
- [4] M. Goossens, A. Samarin, *T_EX at CERN - Local Guide*, CERN CN/US/136, (1992)
- [5] Hewlett-Packard Company, *The Ultimate Guide to the vi and ex Text Editors*, Benjamin/Cummings Publishing Company, Inc., Redwood City, USA, (1990)
- [6] T.L.J. Howard, W.T. Hewitt, R.J. Hubbard, K.M. Wyrwas, *A Practical Introduction to PHIGS and PHIGS PLUS*, Addison-Wesley, Amsterdam, (1991), ISBN 0-201-41641-7
- [7] R.O. Jenks, R.S. Sutor, *axiom, The Scientific Computation System*, Springer
- [8] D.E. Knuth, *The T_EXbook*, Addison-Wesley, Reading, (1990)
- [9] D.E. Knuth, *Computers and Typesetting*, Vol. A – E, Addison-Wesley, Reading, (1986)
- [10] E. Krol, *The Whole INTERNET, Users's Guide and Catalog*, O' Reilly, (1992), ISBN 1-56592-025-2
- [11] L. Lamport, *L^AT_EX, A Document Preparation System*, Addison-Wesley, 2nd ed (1994)
- [12] L. Lamb, *Learning the Vi*, O' Reilly & Associates, Inc, Sebastopol, USA, (1990)
- [13] F. Mittelbach, R. Schöpf, *The New Font Selection - User Interface to Standard L^AT_EX*, TUGboat 10,2 (1989) 222-238
- [14] *NYSERVNet, New User's Guide to Useful and Unique Resources on the Internet*, stored in `rzri6b:/usr/local/doc/internet/nysernet.guidev2.txt`
- [15] St. Talbott, *Managing Projects with make*, O' Reilly & Associates, Inc
- [16] St. Wolfram, *Mathematica, A System for Doing Mathematics by Computer*, Addison Wesley
- [17] M.Goossens,F.Mittelbach,A.Samarin *The LaTeX2e Companion*, Addison Wesley (1994)

Index

|, 2-1, 2-3
~, 2-3
*, 2-3
+ppu option, 8-14
-q extname option, 8-14
., 1-8, 2-3, 3-1
.., 2-3, 3-1
.Z file, 2-3
.elm/aliases.text file, 4-4
.emacs file, 5-5
.exrc file, 5-3
.forward file, 4-4
.kshrc file, 1-9, 1-10
.mailrc file, 4-4
.mwmrc file, 1-3
.netrc file, 8-17
.profile file, 1-10, 8-12
.profile-common file, 1-10
.rhosts file, 4-16
.sh.history file, 1-11
.vueprofile file, 1-10, 8-12
.xinitrc file, E-2
.xsession file, E-2
/, 2-3
/cern, 8-12
/d, 3-1
/d file system, 8-15
/etc/profile file, 1-10
/s, 3-1
/tmp, 3-1
/u, 3-1
/usr/local/bin/.kshrc file, 1-9, 1-10
/usr/local/doc/tex, 6-4
/usr/local/lib/tex/inputs, 6-1, 6-4, 6-5
<, 2-1, 2-3
<Ctrl-c>, 2-2
>, 2-1, 2-3
>>, 2-1
&, 2-3
:, 1-10
~, 1-9
3270 emulation, 4-11
3270, 1-4
a.out file, 7-1
access
 mvs, 4-11
 to Unix
 text oriented, 1-3
 via X-windows, 1-5
 via XDMCP, 1-1
 with HP VUE, 1-3
account
 how to obtain, 1-1
adb, 7-2
aixterm, 1-4, 1-6, 4-8
alias
 mail, 4-4
AMSTEX, 6-2
amstex option, 6-2
anonymous login, 4-3, F-1
APE, 8-15
applymap, 8-5
ar, 7-1, 8-14
archie, F-1
archive, 7-1
Archiving, 4-12
article documentstyle, 6-1
ascii subcommand of ftp, 4-10
at, 4-7
Athena widget, 1-8
ATL Automatic Tape Library, 8-15
aux file, 6-2
auxiliary
 aux file, 6-1
avant option, 6-6
avg4, 8-5
awk, 6-1
AXIOM, 8-1
axiom, 8-1
background job, 2-3
backup, 4-12
bash, 1-6, C-1
batch system, 4-14
bbl file, 6-2
Benutzerberatung, i
bibliography
 bbl file, 6-2
binary subcommand of ftp, 4-10
bison, C-1
BLAS, 8-2
bmp file, 8-9
book documentstyle, 6-1
bookman option, 6-6
bounding box, 6-7, 6-9
 specify for epsfig, 6-9
Bourne again shell, 1-6
Bourne shell, 1-6
C (compiler), 7-1

- C shell, 1-6
- C++, 7-1
- cancel
 - background job, 2-2
 - batch job, 4-14
 - print job, 4-5
 - process, 2-2
- cancel, 4-5
- car file, 8-12, 8-13
- case subcommand of ftp, 4-10
- case sensitive
 - file name, 3-1
- cat, 2-2, 3-3
- cc, 7-1, 7-3
- cd subcommand of ftp, 4-9
- cd, 1-7, 2-2, 3-1
- cdf file, 8-14
- CERN program library, 8-10
 - new version, 8-12
 - old version, 8-12
 - pro version, 8-12
- cerninfo, 8-12
- cernlogin, 8-12
- cernmins documentstyle, 6-12
- change
 - directory, 2-2, 3-1
 - password, 2-2
 - permission, 2-2, 3-4
- character
 - count, 2-2
- chmod, 2-2, 3-4
- class
 - job, 4-14
- click mouse button, 1-8
- close subcommand of telnet, 4-8
- cmz file, 8-13
- cmz, 7-2, 8-11, 8-14
- cmzlogon.kumac, 8-14
- colors
 - with TeX, 6-10
- COMIS, 8-13
- command history, 1-11
- command line editing, 1-10
- commands, 2-1
 - options, 2-1
 - switch, 2-1
- compiler, 7-1, 7-3
 - option, 7-1
- compress, 2-3, C-1
- compress, 2-3
- concatenate
 - file, 3-3
- conversion
 - image formats, 8-5, 8-7
- copy
 - file, 2-2, 3-3
- copy station
 - for exabyte tapes, 4-16, B-1
- count
 - characters, 2-2
 - lines, 2-2
 - words, 2-2
- cp, 2-2, 3-3
- create
 - directory, 2-2, 3-2
- crop, 8-5
- cross-references
 - toc file, 6-1
- csh, 1-6
- CSPACK, 8-13
- current directory, 3-1
- CVS, C-1
- cvtgdf, 8-9
- data analysis
 - CERN Software, 8-10
 - in parallel, 8-11
- dbx, 7-2
- debug program, 7-2
- decterm, 4-8
- delete
 - directory, 2-2, 3-2
 - file, 2-2, 3-3
- desktop environment, 1-6
 - Visual User Environment, 1-6
- device independent file, *see* dvi
- \ding, 6-6
- dingbat option, 6-6
- \dingfill, 6-6
- \dingline, 6-6
- \dinglist, 6-6
- directory
 - change, 2-2, 3-1
 - create, 2-2, 3-2
 - current, 3-1
 - home, 3-1
 - list, 2-2, 3-1
 - move, 3-2
 - parent, 3-1
 - remove, 2-2, 3-2
 - root, 2-3, 3-1
 - working, 3-1
- display
 - file, 2-2, 3-3

- display address, 1-4
- display server, *see* X-server
- distortion of picture
 - epsfig, 6-10
- document style, 6-1
- documentation
 - CERN software, 8-10
 - emacs, 5-6
 - LaTeX, 6-4
 - online, 2-2, 4-1, 8-10
 - TeX, 6-4
- DOMAIN, 4-12
- dpcolor option, 6-10
- draft mode (epsfig), 6-9
- drag graphical object, 1-8
- drivers
 - documentation, 6-4
- DSM, 4-12
- DSM-clients, 4-12
- dvi
 - driver, 6-2, 6-9
 - dvips, 6-5, 6-7
 - previewer, 6-2
 - xdvi, 6-6
- dvi file, 6-1, 6-2, 6-4, 6-7
- dvips, 6-2, 6-4-6-8, 6-10
- dxdb, 7-2
- dxnotepad, 5-12
- dxterm, 1-4, 1-6

- e, 5-11
- echo, 1-9, 2-3
- ed, 5-11
- edit
 - goto line, 5-8
 - multiple files, 5-7
- edit mode
 - sofedit, 5-9
- editing in command line, 1-10
- EDITOR, 1-9
- editors, 5-1
 - Notepad, 5-12
 - edt, 5-6
 - emacs, 5-3
 - dxnotepad, 5-12
 - e, 5-11
 - ed, 5-11
 - edt+, 5-6
 - INed, 5-11
 - LPEX, 5-11
 - sofedit, 5-7
 - softlpex, 5-11
 - uni-XEDIT, 5-12
 - vi, 5-1
 - vuepad, 5-12
 - xe, 5-12
 - xedit, 5-10
- edt, 5-6
- edt+, 5-6
 - GOLD key, 5-6
 - keypad functions, 5-6
 - KEYPAD mode, 5-6
- eeepic option, 6-7
- elm, 4-4
- elmalias, 4-5
- emacs
 - documentation, 5-6
- emacs, 5-3, C-1
 - basic keystrokes, 5-5
 - commands, 5-3
 - modes, 5-4
- encapsulated PostScript, 6-7
- ENV, 1-9
- environment variable, 1-8, 1-9, 2-3
 - DISPLAY, 1-9
 - EDITOR, 1-9
 - ENV, 1-9
 - HOME, 1-9
 - LPDEST, 1-9, 4-5
 - PATH, 1-10
 - PRINTER, 4-5
 - PS1, 1-10
 - PWD, 1-10
 - TERM, 1-10
 - VISUAL, 1-10
- epic option, 6-7
- EPIO, 8-13
- eps file, 8-10
- epsfig option, 6-6, 6-8
 - \epsfig, 6-8-6-10
- equivalent hosts, 4-7
- ESSL, 8-2
- ex, 5-2
- exit, 1-5, 1-6
- experiment data
 - access from Unix, 8-15
 - access via ftp, 8-16
 - access via nfs, 8-17
 - analysis, 8-15
 - Fortran read, 8-15
 - MVS naming conventions, 8-16
- export, 1-4, 1-8, 1-9

- f77, 7-1

fancyheadings documentstyle, 6-12
 fant, 8-5
 FFREAD, 8-13
 file
 change permission, 2-2, 3-4
 concatenate, 3-3
 copy, 2-2, 3-3
 display, 2-2, 3-3
 move, 2-2, 3-3
 name, 3-1
 case sensitive, 3-1
 print, 4-5
 remove, 2-2, 3-3
 system, 3-1
 transfer, 4-9
 file name completion, 1-10
 file transfer protocol, 4-9
 files, produced by LaTeX, 6-1
 find, 2-3
 finger, 2-2
 fmt file, 6-2
 fmt file, 6-2
 font, 6-2
 files
 gf, 6-2
 pk, 6-2
 pxl, 6-2
 large size, 6-7
 fontutil, C-1
 format, 6-2
 fort77, 7-1, 8-14
 Fortran, 7-1
 forward
 mail, 4-4
 Free Software Foundation, C-1
 ftp
 background processing, 8-17
 experiment data transfer, 8-16
 ftp, 4-3, 4-9, 8-16, F-1
 anonymous login, 4-3, F-1

 gcc, C-1
 gdb, C-1
 GDDMXD, 4-11
 GDF file, 8-9
 GDFIP TSO command, 8-9
 GEANT, 8-11, 8-13, 8-14
 example Makefile, 8-14
 interactive main program, 8-13
 geant315.o, 8-13
 geant316.o, 8-13
 get subcommand of ftp, 4-10

getting help
 Unix Commands, 4-1
 getx11, 8-5
 gf file, 6-2, 6-4
 Ghostscript, 6-6, C-1
 ghostview, 4-2, 6-4, 6-6, 6-9, 6-10, C-1
 gif file, 8-5-8-7, 8-9, 8-10
 giftorle, 8-5
 GNU, C-1
 C Lib, C-1
 tar, C-1
 GNU project, 6-6
 gnuchess, C-1
 gnuplot, C-1
 GOOSY raw data, 8-15
 gopher server, 4-3
 GRAFLIB, 8-13
 graphical tools, 8-4
 graphics
 GDDM display, 4-11
 merge with TeX, 6-7
 graPHIGS, 8-9
 graytorle, 8-5
 grep, 2-2
 gs, 6-6, C-1
 gsibrief documentstyle, 6-11
 gsifoil documentstyle, 6-11
 gsirep documentstyle, 6-11
 gunzip, C-1
 gxint315.o, 8-13
 gzip, C-1

 hardware
 installed, E-1
 HBOOK, 8-13
 hdf file, 8-7
 height parameter (epsfig), 6-10
 help, 2-2, 4-1
 help subcommand of ftp, 4-10
 help subcommand of telnet, 4-8
 help desk, i
 helv option, 6-6
 HIGZ, 8-13
 higz_windows.dat, 8-12
 hmtl file, 4-2
 holding, A-1
 HOME, 1-9
 home directory, 1-10, 3-1
 \$HOME/higz_windows.dat, 8-12
 host
 equivalent, 4-7
 hostname, 1-7

- HP VUE, *see* Visual User Environment
- HPLLOT, 8-13
- hpterm, 1-4, 1-6, 4-8
- html file, 4-2
- HTTP, 4-3
- hypertext, 4-1

- IBM Mainframe Access, 4-11
- icon, 1-6, 1-7
- icon file, 8-7
- IDL, 8-4
- idx file, 6-2
- iff file, 8-7
- image display, 8-9
- image formats, 8-7
- image processing, 8-5
- imconv, 8-7
- inactive versions, 4-12
- inc, 4-5
- ind file, 6-2
- indent, C-1
- indexing
 - idx file, 6-2
 - ind file, 6-2
 - makeindex, 6-2
- INed, 5-11
- info, 2-2, 4-1
- information system, 4-1
- input
 - standard, 2-1
- internet
 - address, 1-4
 - name, 1-4
- Internet address, F-1
- Internet services, F-1
- interrupt TeX, 6-5
- ispell, C-1

- job
 - class, 4-14
- jobs, 2-2
- jpeg file, 8-9
- jpg file, 8-9

- KAPACK, 8-13
- KERNLIB, 8-13
- kill
 - background job, 2-2
 - process, 2-2
- kill, 2-2
- Korn shell, 1-6
 - profile files, 1-10

- ksh, 1-6
- KUIP, 8-13
- kuipc, 8-14

- LAPACK, 8-4
- LaTeX, 6-1, 6-2
 - documentation, 6-4
 - error messages, 6-1
 - format, 6-2
 - logfile, 6-1
 - merge graphics, 6-7
 - style file, 6-1
 - with colors, 6-10
- latex, 6-4, 6-5
- LaTeX2e, 6-12
- launcher, 1-3
- launcher menu, 1-2, 1-3
- lcd subcommand of ftp, 4-9
- learn, 2-2
- less, C-1
- letter documentstyle, 6-1
- libg++, C-1
- libgeant315.a, 8-13
- libgeant316.a, 8-13
- libgrafGKS.a, 8-13
- libgraflib.a, 8-13
- libgrafX11.a, 8-13
- libkernlib.a, 8-13
- libmathlib.a, 8-13
- libpacklib.a, 8-13
- libpawlib.a, 8-13
- library, 7-1
 - CERN program, 8-10
 - ESSL, 8-4
 - LAPACK, 8-4
 - mathematical, 8-1
 - NAG, 8-1
- line
 - count, 2-2
- linker, 7-1
- list
 - directory, 2-2, 3-1
 - process, 2-2
 - users, 2-2
- list-of-figures
 - lof file, 6-1
- list-of-tables
 - lot file, 6-1
- llcancel, 4-14, 4-15
- llq, 4-14, 4-15
- llstatus, 4-14, 4-15
- llsubmit, 4-14, 4-15

- LoadLeveler, 4-14, 4-15
 - using NQS scripts, 4-15
- lock terminal screen, 1-4, 1-7
- lof file, 6-2
- log file, 6-1
- login
 - remote, 4-8
 - via telnet, 1-3
- login server, 1-1
- logout, 1-5
- lot file, 6-2
- lp, 2-1, 4-5
- LPDEST, 1-9
- LPDEST, 1-9, 4-5
- LPEX, 5-11
- lpr, 4-5
- lprm, 4-5
- lpstat, 4-5
- ls, 2-1, 2-2, 3-1

- macro, 6-2
- magnetic tape, 4-15
- magnification, 6-2
- mail
 - address, 4-4, 4-5
 - alias, 4-4
 - forward, 4-4
 - program
 - elm, 4-4
 - mail, 4-4
 - xmh, 4-5
- mail, 4-4
- Mail/components file, 4-5
- mailx, 4-4
- make, 7-2, 7-3, C-1
- Makefile file, 7-2, 8-14
- makefile file, 7-2, 7-3
- makeindex, 6-2
- man, 2-1, 2-2, 4-1
- math, 8-1
- mathbook, 8-1
- Mathematica, 8-1
- mathematical library, 8-1
- MATHLIB, 8-13
- mcut, 8-6
- menu
 - option -, 1-8
 - popup -, 1-8
 - pulldown -, 1-8
- merge text and graphics, 6-7
- mergechan, 8-6
- metafont, 6-2, 6-5

- mf file, 6-2, 6-4
- mf, 6-2
- mget subcommand of ftp, 4-10
- MINUIT, 8-13
- mkdir subcommand of ftp, 4-10
- mkdir, 2-2, 3-2
- more, 2-2
- Motif
 - widget, 1-8
 - window, 1-7, D-1
 - window manager, 1-5, 1-7
- mount, 8-17
- mountmvs, 8-17
- mouse, 1-7
- move
 - directory, 3-2
 - file, 2-2, 3-3
- mpnt file, 8-7
- mput subcommand of ftp, 4-10
- mt, 4-16, B-1
- mv, 2-2, 3-2, 3-3
- mvs
 - access, 4-11
 - login to Unix, A-1
- mwm, 1-7
 - start with HP VUE, 1-3
 - start with launcher, 1-3
 - start with XDMCP, 1-1
- mwm, *see* Motif window manager
- mxrn, F-2

- NAG FORTRAN Library, 8-1
- naghelp, 8-2
- nagtest, 8-2
- ncs option, 6-6
- NetNews, 4-3, F-2
- newsgroups, 4-3, F-2
- NFSS, 6-5, 6-6
- nmap subcommand of ftp, 4-10
- NODENAME, 4-12
- Notepad, 5-12
- NQS, 4-15
 - using scripts with LoadLeveler, 4-15
- ntrans subcommand of ftp, 4-10
- Ntuples
 - paw, 8-11
- numerical subroutines, 8-1

- on-line tutorial, 2-2
- online documentation, 4-1
- open subcommand of telnet, 4-8
- open, 1-4

- option
 - style, 6-1
- option menu, 1-8
- output
 - standard, 2-1
- PACKLIB, 8-13
- palatino option, 6-6
- parent directory, 3-1
- passwd, 1-7, 2-2
- password
 - change, 1-7, 2-2
- patch, C-1
- patchy, 8-11
- PATH, 1-10
- path name, 2-3
 - show, 2-2, 3-2
- path name , 3-1
- pattern
 - matching, 2-2
- pattern matching, 2-1
- PAW, 6-8, 8-13, 8-14
 - example Makefile, 8-14
- paw, 8-11, 8-12
- PAWLIB, 8-13
- pawlogon.kumac, 8-12
- pbm file, 8-7, 8-9
- pcx file, 8-7, 8-9, 8-10
- permission
 - change, 2-2, 3-4
 - determine, 3-3
- pgm file, 8-7, 8-9
- PHIGS, 8-7
- PIAF, 8-11
- pic file, 8-7
- pict file, 8-7, 8-10
- PiCTeX package, 6-7
- picture environment, 6-7
- picture data formats, 8-5
- pipe, 2-1, 2-3
- pix file, 8-7
- pk file, 6-2, 6-5
- PL/I, 7-1
- plain, 6-2
 - TeX format, 6-1
- pm file, 8-9
- pnm file, 8-7
- pointer device, 1-7
- popup menu, 1-8
- PostScript, 6-4, 6-5
 - encapsulated, 6-7
 - fonts with TeX, 6-6
 - previewer, 6-6
- ppm file, 8-7, 8-9, 8-10
- press mouse button, 1-8
- previewer, 6-2
- primitive, 6-2
- printenv, 1-9
- PRINTER, 4-5
- printers
 - table of, 4-6
- printing, 4-5
 - help on, 4-5
- process
 - kill, 2-2
 - list, 2-2
- profile files, 1-10
- program development, 7-1
- prompt subcommand of ftp, 4-10
- ps file, 6-4, 8-7
- ps, 2-2
- PS1, 1-10
- \psdraft, 6-9
- psfig option, 6-8
- \psfull, 6-9
- \psnoisy, 6-9
- \pssilent, 6-9
- public domain software
 - find, F-1
- pull-down menu, 1-8
- put subcommand of ftp, 4-10
- PWD, 1-10
- pwd subcommand of ftp, 4-10
- pwd, 1-9, 2-2, 3-2
- pxl file, 6-2
- pyrmask, 8-6
- queues
 - table of print, 4-6
- quit subcommand of ftp, 4-9
- quit subcommand of telnet, 4-8
- quit, 1-5
- ras file, 8-7
- rawtorle, 8-6
- rccp, 4-9, 4-10
- RCS, C-1
- redirection
 - input, 2-1
 - output, 2-1
- regular expression, 2-1
- release mouse button, 1-8
- remote copy, 4-10
- remote login, 4-8

- remote processing, 4-8
- remote shell, 4-9
- remove
 - directory, 2-2, 3-2
 - file, 2-2, 3-3
- remsh, 4-8, 4-9
- report documentstyle, 6-1
- repos, 8-6
- rexe, 4-8, 4-9
- rgb file, 8-7
- r1a file, 8-7
- RLE file, 8-5, 8-6
- rle file, 8-7, 8-9
- RLE-format, 8-5, 8-7
- rleaddcom, 8-6
- rleaddeof, 8-6
- rlebg, 8-6
- rlebox, 8-6
- rleClock, 8-6
- rlecomp, 8-6
- rledither, 8-6
- rleflip, 8-6
- rlehdr, 8-6
- rlehisto, 8-6
- rleldmap, 8-6
- rlemandl, 8-6
- rlenoise, 8-6
- rlepatch, 8-6
- rleprint, 8-6
- rlequant, 8-6
- rlescale, 8-6
- rleselect, 8-6
- rresetbg, 8-6
- rlespiff, 8-6
- rlesplice, 8-6
- rlesplit, 8-6
- rleswap, 8-6
- rletoascii, 8-6
- rletogif, 8-6
- rletogray, 8-6
- rletops, 8-6
- rletoraw, 8-6
- rletotiff, 8-6
- rlezoom, 8-6
- rlogin, 4-8
- rm, 2-2, 3-3
- rmdir, 2-2, 3-2
- root
 - directory, 2-3, 3-1
- rpbm file, 8-7
- rpgm file, 8-7
- rpnm file, 8-7
- rppm file, 8-7
- rsh, 4-9
- run LaTeX, 6-4
- SATAN raw data, 8-15
- SCCS, 7-2
- screen dumps, 8-9
- script, 1-7, 2-3
 - dot script, 2-3
- search
 - directory tree, 2-3
- sed, 6-1
- sh, 1-6
- shell, 1-4, 1-6
 - command, 1-4
 - remote, 4-9
- shell parameter
 - global, *see* environment variable
 - local, 1-8
- shell script, 1-7, *see* script
- SIGMA, 8-13
- smush, 8-6
- sofedit, 5-7
- softlpex, 5-11
- sort, 2-2
- sort, 2-2
- source-code management system, 8-11
- \special, 6-7, 6-9
- special characters, 2-3
- standard input, 2-1
- standard LaTeX style, 6-1
- standard output, 2-1
- status
 - background jobs, 2-2
 - batch job, 4-14
 - print job, 4-5
- string
 - type, 2-3
- sty file, 6-1, 6-4
- style
 - article, 6-1
 - book, 6-1
 - documentation, 6-4
 - file, 6-1
 - letter, 6-1
 - major, 6-4
 - minor, 6-4
 - report, 6-1
 - standard LaTeX style, 6-1
- submit
 - batch job, 4-14
 - print job, 4-5

- symbolic calculations, 8-1
- symbolic debugger, 7-2
- synu file, 8-7

- tape, 4-15
 - allocation, 4-15
 - copy station, 4-16, B-1
 - stacker for exabyte tapes, 4-16
- tape, 4-15
- tape stacker, B-1
- tar, 2-3, F-1
- tc shell, 1-6
- tcopy, 4-16
- tcsh, 1-6
- telnet, 1-3
- telnet, 1-1, 4-8, A-1, F-1
 - 3270 emulation, 4-12
- telnet window, 1-3, 1-6
 - close, 1-5
- TERM, 1-10
- terminal emulation, 1-5
 - 3270, 4-11
- terminal emulation program, 1-4
- terminal type, 1-4
 - aixterm, 1-4
 - dxterm, 1-4
 - hpterm, 1-4
 - xterm, 1-4
- terminal window, 1-5, 1-7
 - close, 1-5
 - open, 1-2, 1-5
 - for MVS, 1-5
- TeX, 6-1
 - documentation, 6-4
 - with colors, 6-10
- tex, 6-1, 6-5
- Texinfo, C-1
- text processing, 6-1
- tfm file, 6-2, 6-4
- tif file, 8-7
- tiff file, 8-6, 8-9
- times option, 6-6
- to8, 8-6
- tobw, 8-6
- toc file, 6-2
- type
 - file, 2-2, 3-3

- umountmvs, 8-17
- uncompress, 2-3, C-1
- uncompress, 2-3, F-1
- unexp, 8-6

- uni-XEDIT, 5-12
- uniq, 2-2
- Unix
 - file system, 3-1
 - machines, E-1
- unix
 - account, 1-1
- Unix commands, 2-1
- unslice, 8-6
- URL, 4-3
- urt, *see* Utah Raster Toolkit
- Usenet, F-2
- user subcommand of telnet, 4-8
- users
 - list, 2-2
 - show information, 2-2
- Utah Raster Toolkit, 8-5

- vi
 - .exrc file, 5-3
 - EXINIT variable, 5-3
 - ex command mode, 5-1, 5-2
 - basic keystrokes, 5-2
 - command mode, 5-1
 - insert mode, 5-1
 - operating modes, 5-1
- VISUAL, 1-10
- Visual User Environment, 1-1, 1-3, 1-5, 1-6
 - close, 1-5
- vt100, 1-4
- vt220, 1-4
- VUE, *see* Visual User Environment, 1-3
- vuepad, 5-12

- wais server, 4-3
- wc, 2-2
- who, 2-2
- who am i, 2-2
- whoami, 1-7
- widget, 1-8
- width parameter (epsfig), 6-10
- window, 1-5, 1-7
 - active, 1-8
 - background, 1-8
- window manager, *see* Motif window manager
- word
 - count, 2-2
- working directory, 3-1
- World Wide Web, 4-1
- write string, 2-3
- WWW, 4-1
- www, 4-2

x file, 8-7
X-server, 1-5
X-terminal, 1-1, 1-3
X-window server, *see* X-server
X-windows, 1-1, 1-5
x3270, 4-11
xarchie, F-2
xbm file, 8-7, 8-9
xdb, 7-2
xde, 7-2
xdm, E-2
XDMCP, 1-1
XDMCP, E-2
xdmcp, 1-1
xdvi, 6-4, 6-6
xe, 5-12
xedit, 5-10
xinit, 1-5
xlf, 7-1, 8-14
xloadl, 4-14, 4-15
xmh, 4-4, 4-5
xpick, 6-8, 8-9
xt, 1-5, 4-7, 4-8, 4-11
xterm, 1-4, 4-8
XtoPS, 8-7
xv, 4-2, 8-9, 8-10
xwd file, 8-7
xwd, 8-7
xwww, 4-2, 8-10

z shell, 1-6
ZBOOK, 8-13
ZEBRA, 8-13
zsh, 1-6