PWB/UNIX Documentation Roadmap

J. R. Mashey

Bell Laboratories
Piscataway, New Jersey 08854

1. INTRODUCTION

A great deal of documentation exists for PWB/UNIX. It has different formats, is contributed by many different people, and is modified frequently. New users are often overcome by the volume and distributed nature of the documentation. This ''roadmap'' attempts to be a terse, up-to-date outline of crucial documents and information sources.

Numerous people have contributed comments and information for this ''roadmap,'' in order to make it as helpful as possible for PWB/UNIX users. *However, many of these comments are accurate only with regard to PWB/UNIX and may well be totally inapplicable to other versions of UNIX.*

1.1 Things to Do

See a local PWB/UNIX system administrator to obtain a ''login name'' and get other appropriate system information.

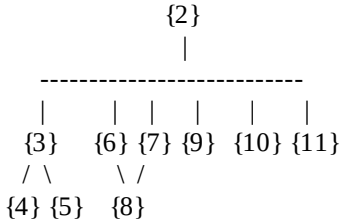
1.2 Notation Used in this Roadmap

{*N*} → Section N in this ''roadmap.''
++ → item required for everyone.
\+ → item recommended for most users.

All other items are optional and depend on specific interest (a list of relevant documents appears in the Table of Contents of *Documents for the PWB/UNIX Time-Sharing System*).

Items in Section *N* of the *PWB/UNIX User's Manual* are referred to by *name*(N).

1.3 Prerequisite Structure of Following Sections



2. BASIC INFORMATION

Don't do anything else until you have learned most of this section. You must know how to log onto the system, make your terminal work correctly, enter and edit files, and perform basic operations on directories and files.

2.1 PWB/UNIX User's Manual + +

- Read *Introduction* and *How to Get Started.*
- Look through Section I to become familiar with command names.
- Note the existence of the *Table of Contents* and of the *Permuted Index.*

Section I will be especially needed for reference use.

2.2 UNIX for Beginners + +

2.3 A Tutorial Introduction to the UNIX Text Editor + +

2.4 Advanced Editing on UNIX +

2.5 PWB Papers from the Second International Conference on Software Engineering +

Gives an overview of the Programmer's Workbench.

2.6 Things to Do

- Do all the exercises found in {2.2} and {2.3}, and maybe {2.4}.
- Create a file named ''.mail'' in your login directory,* so that other people (such as system administrators) can send you mail. This can be done by:

  cp /dev/null .mail

- If you want some sequence of commands to be executed each time you log in, create a file named ''.profile'' in your login directory containing the commands you want executed. For more information, see *Initialization* in *sh*(I).
- Files in directory ''/usr/news'' contain recent information on various topics. To see what has been updated recently, type:

  ls –lt /usr/news

  and then print any files that look interesting. Other useful actions include:

mail –f /usr/news/.mail	gives recent history from primary system mailbox.
cat /usr/news/helpers	gives contacts and telephone numbers for counseling, file restorals, trouble reporting, and other services.
nroff –mm /usr/news/roadmap	prints current copy of this ''roadmap.''
cat /usr/news/terminals	gives recommendations on selection of computer terminals.

2.7 Manual Pages to Be Studied

The following commands are described in Section I of the *PWB/UNIX User's Manual*, and are used for creating, editing, moving (i.e., renaming), and removing files:

cat(I)	concatenate and print files (no pagination).
chdir(I)	change working (current) directory; a.k.a. *cd*(I).
cp(I)	make a copy of an existing file.
ed(I)	text editor.
ls(I)	list a directory; file names beginning with ''.'' are not listed unless the ''–a'' flag is used.
mkdir(I)	make a (new) directory.
mv(I)	move (rename) file.
pr(I)	print files (paginated listings).
rm(I)	remove (delete) file(s).
rmdir(I)	remove directory(ies).

The following help you communicate with other users, make proper use of different kinds of terminals, and print manual pages on-line:

login(I)	sign on.
mail(I)	send mail to other users; inspect mail from them, or contents of the system mailbox.
man(I)	print pages of *PWB/UNIX User's Manual.*
stty(I)	set terminal options; i.e., inform the system about the hardware characteristics of your terminal.
tabs(I)	set tab stops on your terminal.
terminals(VII)	gives descriptions of commonly-used terminals.
who(I)	print list of users currently logged in.
write(I)	communicate with another (logged in) user.

* The directory you are in right after logging into the system.

3. BASIC TEXT PROCESSING AND DOCUMENT PREPARATION

You should read this section if you want to *use* existing text processing tools to write letters, memoranda, manuals, etc.

3.1 PWB/MM—Programmer's Workbench Memorandum Macros + +

This is a reference manual, and can be moderately heavy going for a beginner. Try out some of the examples, and stick close to the default options.

3.2 Typing Documents with PWB/MM + +

A handy fold-out.

3.3 NROFF/TROFF User's Manual +

Describes the text formatting language in great detail; look at the REQUEST SUMMARY, but don't try to digest the whole manual on first reading.

3.4 Documentation Tools and Techniques +

This overview of UNIX text processing methods is one of the papers from the Second International Conference on Software Engineering. (See {2.5} above).

3.5 Manual Pages to Be Studied

mm(I)	makes it easy to specify standard options to *nroff*(I).
nroff(I)	read to see formatter option flags.
spell(I)	identifies possible spelling errors.
tmac.name(VII)	list of text-formatting macro packages.
typo(I)	identifies possible typographical errors.

To obtain some special functions (e.g., reverse paper motion, subscripts, superscripts), you must either indicate the terminal type to *nroff* or post-process *nroff* output through one of the following:

450(I)	newer Diablo printer terminals, such as the DASI450, DIABLO 1620, XEROX 1700, etc.
col(I)	terminals lacking physical reverse motion, such as the Texas Instrument 700 series.
gsi(I)	older Diablo printer terminals, such as the GSI300, DASI300, DTC300, etc.
hp(I)	Hewlett-Packard 2640 terminals (HP2640A, HP2640B, HP2644A, HP2645A, etc.).

4. SPECIALIZED TEXT PROCESSING

The tools listed in this section are of a more specialized nature than those in {3}.

4.1 TBL—A Program to Format Tables +

Great help in formatting tabular data (see *tbl*(I)).

4.2 Typesetting Mathematics—User's Guide (2nd Edition) +

Read this if you need to produce mathematical equations. It describes the use of the equation setting commands *eqn*(I) and *neqn*(I).

4.3 A TROFF Tutorial

An introduction to formatting text with the phototypesetter.

4.4 Manual Pages to Be Studied

diffmark(I)	marks changes between versions of a file, using output of *diff*(I) to produce ''revision bars'' in the right margin.
eqn(I)	preprocessor for mathematical equations (phototypesetter).
neqn(I)	preprocessor for mathematical equations (terminals).
tbl(I)	preprocessor for tabular data.
troff(I)	formatter for phototypesetter.

5. ADVANCED TEXT PROCESSING

You should read this section if you need to *design* your own package of formatting macros or perform other actions beyond the capabilities of existing tools; {3} is a prerequisite, and familiarity with {4} is very helpful, as is an experienced advisor. It takes a great deal of effort to write a good package of macros for general use. Don't reinvent what you can borrow from an existing package (such as PWB/MM).

5.1 NROFF/TROFF User's Manual + +

Look at this in detail and try modifying the examples. If you are going to use the phototypesetter, do the same for *A TROFF Tutorial* ({4.3} above).

5.2 Things to Do

It is fairly easy to use the text formatters for simple purposes. A typical application is that of writing simple macros that print standard headings in order to eliminate repetitive keying of such headings. It is extremely difficult to set up general-purpose macro packages for use by large numbers of people. If possible, try to use an existing package or modify one as needed. Look at existing packages first—see *tmac.name*(VII).

5.3 Manual Pages to Be Studied

All pages mentioned in {3} and {4}.

6. COMMAND LANGUAGE (SHELL) PROGRAMMING

The Shell provides a powerful programming language for combining existing commands. This section should be especially useful to those who want to automate manual procedures and build data bases.

6.1 The UNIX Time-Sharing System + +

6.2 PWB/UNIX Shell Tutorial + +

6.3 Things to Do

If you want to create your own library of commands, create a ''.path'' file in your login directory, as described in *sh*(I).

6.4 Manual Pages to Be Studied

Read *sh*(I) first; the following pages give further details on commands that are most frequently used within command language programs:

echo(I)	echo arguments (typically to terminal).
equals(I)	Shell assignment command (for variables).
exit(I)	terminate command file.
expr(I)	evaluate an algebraic expression.
fd2(I)	redirect diagnostic output.
if(I)	conditional command.
next(I)	read command input from named file.
nohup(I)	run a command immune to communications line hang-up.
onintr(I)	handle interrupts in Shell files.
pump(I)	Shell data transfer command.
sh(I)	Shell (command interpreter).
shift(I)	adjust Shell arguments.
switch(I)	Shell multi-way branch command.
while(I)	Shell iteration command.

7. FILE MANIPULATION

In addition to the basic commands of {2}, many UNIX commands exist to perform various kinds of file manipulation. Small data bases can often be managed quite simply, by combining text processing (from {5}), command language programming {6}, and commands listed below in {7.2}.

7.1 Things to Do

This ''roadmap'' notes only the most frequently used commands. It is wise to scan Section I of the *PWB/UNIX User's Manual* periodically—you will often discover new uses for commands.

7.2 Manual Pages to Be Studied

The following are used to search or edit files in a single pass:

grep(I)	search a file for a pattern; more powerful and specialized versions include *egrep*(I), *fgrep*(I), and *rgrep*(I).
sed(I)	stream editor.
tr(I)	transliterate (substitute or delete specified characters).

The following compare files in different ways:

cmp(I)	compare files (byte by byte).
comm(I)	print lines common to two files, or lines that appear in only one of the two files.
diff(I)	differential file comparator (minimal editing for conversion).

The following combine files and/or split them apart:

ar(I)	archiver and library maintainer.
cpio(I)	general file copying and archiving.
csplit(I)	split file by context.
split(I)	split file into chunks of specified size.

These commands interrogate files and print information about them:

file(I)	determine file type (best guess).
od(I)	octal dump (and other kinds also).
wc(I)	word (and line) count.

Miscellaneous commands:

find(I)	search directory structure for specified kinds of files.
gath(I)	gather real and virtual files; alias for *send*(I).
help(I)	ask for help about a specific error message.
reform(I)	reformat ''tabbed'' files (often used to truncate lines).
sort(I)	sort or merge files.
tee(I)	copy single input to several output files.
uniq(I)	report repeated lines in a file, or obtain unique ones.

8. C PROGRAMMING

Try to use existing tools first, before writing **C** programs at all.

8.1 Programming in C—A Tutorial + +

Read; try the examples.

8.2 C Reference Manual + +

Terse but complete reference manual.

8.3 A New Input-Output Package +

Describes a new I/O package that is superseding many of the existing routines; write any new code using this package.

8.4 UNIX Programming +

8.5 YACC—Yet Another Compiler Compiler

8.6 LEX—Lexical Analyzer Generator

8.7 Make—A Program for Maintaining Computer Programs

8.8 Things to Do

The best way to learn **C** is to look at the source code of existing programs, especially ones whose functions are well known to you. Much code can be found in directory ‘‘/sys/source’’. In particular, directories ‘‘s1’’ and ‘‘s2’’ contain the source for most of the commands. Also, investigate directory ‘‘/usr/include’’.

8.9 Manual Pages to Be Studied

adb(I)	**C** debugger; more powerful (but more complex) than the older *cdb*(I).
cc(I)	**C** compiler.
cdb(I)	**C** debugger (for post-mortem core dumps and other debugging).
ld(I)	loader (you must know about some of its flags).
lex(I)	generate lexical analyzers.
make(I)	automate program regeneration procedures.
nm(I)	print name (i.e., symbol) list.
prof(I)	display profile data (used for program optimization).
regcmp(I)	compile regular expression.
strip(I)	remove symbols and relocation bits from executable file.
time(I)	time a command.
yacc(I)	parser generator.

9. IBM REMOTE JOB ENTRY (RJE)

This section is for those who use PWB/UNIX to submit jobs to remote computers.

9.1 Guide to IBM Remote Job Entry for PWB/UNIX Users +

9.2 Manual Pages to Be Studied

bfs(I)	big file scanner (scans RJE output).
csplit(I)	split file by context (often used to split RJE output).
fspec(V)	format specification in text files.
reform(I)	reformat files (often used to convert source programs from non-UNIX systems).
rjestat(I)	RJE status and enquiries.
send(I)	submit RJE job.

10. SOURCE CODE CONTROL SYSTEM (SCCS)

SCCS can be used to maintain, control, and identify files of text as they are modified and updated. Its most common use is for maintaining source programs, as well as for keeping track of successive versions of various documents; in combination with *diffmark*(I), this allows one to automatically generate ‘‘revision bars’’ in successive editions of such documents.

10.1 SCCS/PWB User’s Manual + +

10.2 Manual Pages to Be Studied

Of the following, *get*(I), *delta*(I), and *prt*(I) are most frequently used.

admin(I)	administer SCCS files (including creation thereof).
chghist(I)	change the history entry of an SCCS delta.
comb(I)	combine SCCS deltas.
delta(I)	make an SCCS delta (a permanent record of editing changes).
get(I)	get a version of an SCCS file.
prt(I)	print SCCS file.

rmdel(I)	remove a delta from an SCCS file.
sccsdiff(I)	get the differences between two SCCS deltas.
what(I)	find and print SCCS identifications in files.

11. NUMERICAL COMPUTATION

11.1 DC—An Interactive Desk Calculator

11.2 BC—An Arbitrary Precision Desk Calculator Language

11.3 RATFOR—A Preprocessor for a Rational Fortran

11.4 Manual Pages to Be Studied

bas(I)	BASIC interpreter.
bc(I)	interactive language, acts as front end for *dc*(I)
dc(I)	desk calculator.
fc(I)	Fortran compiler/interpreter.
rc(I)	RATFOR preprocessor.