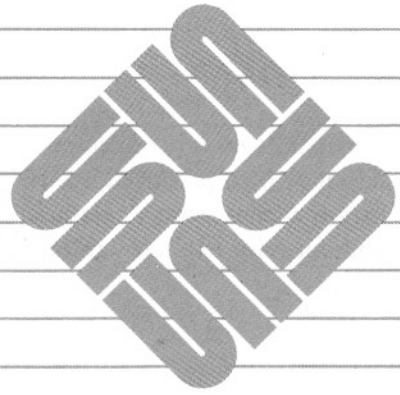
sun microsystems

Getting Started *with* UNIX: Beginner's Guide



Part No: 800-1284-03
Revision A of 17 February 1986

Contents

5

Commands

5

Commands

You type *commands* to tell the system what to do. In past chapters, you learned commands like `cat`, `cp`, and `mv`. In this chapter, you will learn the structure of commands, so that you can understand many more of them without learning all of the structural details each time.

5.1. Command Grammar

Since you've been using some commands, you may have noticed that commands have a certain grammar. One could draw an analogy between UNIX command syntax and English, or other *natural languages*.[22]

Note: Technically, the *command* is part of a *command line* that includes the command and its options, filenames, and other expressions.

All parts of UNIX *command lines* correspond to English parts of speech. A command line consists of the actual command and its *arguments*, the rest of the command line. The command is similar to a verb. Of the arguments, *options* are similar to adverbs, and *filenames* or other *expressions* are similar to objects of the verb.[23]

Two Simple Commands

`date` is a useful command that types the current date and time associated with your geographical location.[24]

If you decided to ask for the current date and time at the stroke before midnight on the last New Year's Eve of the century, you would type `date`:

Figure 5-1 *The* `date` *Command*

```
venus% date
Fri Dec 31 23:59:59 PST 1999
venus%
```

[22] Natural languages are languages that people commonly speak, read, or write within a society or culture.

[23] Interestingly, the noun as subject of the UNIX "sentence" doesn't exist. One may assume that commands are interrogative, so it isn't necessary to preface commands with "you," meaning the system.

For further information on the analogy between UNIX and natural language, see *Thoughts on an All-Natural User Interface*, by Marion O. Harris, pp. 343-347 in the USENIX Conference proceedings for Summer 1985.

[24] Don't ask what happens in outer space! If your date and time don't reflect the appropriate date and time for your location, contact your system administrator or see *System Administration for the Sun Workstation*.

`PST` is an abbreviation for Pacific Standard Time, from the time zone in which this manual was written. To get the Greenwich Mean Time, or u*niversal time*, type the command `date`, followed by the option `-u`:

Figure 5-2 *The* `date` *Command with Option for Universal Time (GMT)*

```
venus% date -u
Sat Jan 1  7:59:59 GMT 2000
venus%
```

Another useful command, `cal`, has one required argument, one optional argument, and no command options. The required argument is the *year* for which you want to construct a calendar. The optional argument is the *month*. So, you can construct a calendar for the whole year of 2001 by typing:

Figure 5-3 *Constructing a Yearly Calendar*

```
venus% cal 2001


                                2001

        Jan                     Feb                     Mar
 S  M Tu  W Th  F  S     S  M Tu  W Th  F  S     S  M Tu  W Th  F  S
    1  2  3  4  5  6                 1  2  3                 1  2  3
 7  8  9 10 11 12 13     4  5  6  7  8  9 10     4  5  6  7  8  9 10
14 15 16 17 18 19 20    11 12 13 14 15 16 17    11 12 13 14 15 16 17
21 22 23 24 25 26 27    18 19 20 21 22 23 24    18 19 20 21 22 23 24
28 29 30 31             25 26 27 28             25 26 27 28 29 30 31

        Apr                     May                     Jun
 S  M Tu  W Th  F  S     S  M Tu  W Th  F  S     S  M Tu  W Th  F  S
 1  2  3  4  5  6  7              1  2  3  4  5                 1  2
 8  9 10 11 12 13 14     6  7  8  9 10 11 12     3  4  5  6  7  8  9
15 16 17 18 19 20 21    13 14 15 16 17 18 19    10 11 12 13 14 15 16
22 23 24 25 26 27 28    20 21 22 23 24 25 26    17 18 19 20 21 22 23
29 30                   27 28 29 30 31          24 25 26 27 28 29 30

        Jul                     Aug                     Sep
 S  M Tu  W Th  F  S     S  M Tu  W Th  F  S     S  M Tu  W Th  F  S
 1  2  3  4  5  6  7                 1  2  3  4                    1
 8  9 10 11 12 13 14     5  6  7  8  9 10 11     2  3  4  5  6  7  8
15 16 17 18 19 20 21    12 13 14 15 16 17 18     9 10 11 12 13 14 15
22 23 24 25 26 27 28    19 20 21 22 23 24 25    16 17 18 19 20 21 22
29 30 31                26 27 28 29 30 31       23 24 25 26 27 28 29
                                                30

        Oct                     Nov                     Dec
 S  M Tu  W Th  F  S     S  M Tu  W Th  F  S     S  M Tu  W Th  F  S
    1  2  3  4  5  6                 1  2  3                    1
 7  8  9 10 11 12 13     4  5  6  7  8  9 10     2  3  4  5  6  7  8
14 15 16 17 18 19 20    11 12 13 14 15 16 17     9 10 11 12 13 14 15
21 22 23 24 25 26 27    18 19 20 21 22 23 24    16 17 18 19 20 21 22
28 29 30 31             25 26 27 28 29 30       23 24 25 26 27 28 29
                                                30 31


venus%
```

To construct a calendar for a specific month, type the number of the *month* and the *year*.

Figure 5-4 *Constructing a Monthly Calendar*

```
venus% cal 6 1969
   June 1969
 S  M Tu  W Th  F  S
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30
venus%
```

Type Ahead or Read Ahead

UNIX has an interesting feature called *type ahead* or *read ahead.* Type ahead allows you to type new commands while the system is executing the current command. When you know that you want to type several commands no matter what the outcome of each command, you can type ahead and the system will execute each command in order as it completes the previous command.

Type ahead is most useful when the system requires a lot of time to complete execution of a command. For example, it usually takes the system a bit of time to execute the `lpr` command. You can type `lpr` (followed by RETURN), then `ls`, without waiting for the system to print the command prompt that signals completion of the `lpr` command. This is what it looks like:

Figure 5-5 *Type Ahead or Read Ahead*

```
venus% ls
actors          alaska          hinterland      wilde
venus% lpr wilde
ls
venus% actors          alaska          hinterland      wilde
venus%
```

One side effect of type ahead is that the system types out the results of the `ls` command just after the command prompt that occurs between the `lpr` command and the `ls` command. That looks funny because the RETURN that you pressed after `ls` appears directly after `ls`, not after the command prompt.

Commands as Verbs

As mentioned above, commands are the "verb" of the command line. Some commands have no arguments, many have optional arguments (like the `cal` command's optional month argument), and some require arguments to execute without an error (`lpr` requires a *filename* argument).

Options as Adverbs

An option is one possible argument to a command. The option, sometimes known as a *flag*, tells the command to execute in a certain way. For instance, the `date` command, with its `-u` option, types the Greenwich Mean date and time, rather than the local date and time.

Usually, a `-` precedes the option to a command. `-` is a *marker* for options, and precedes most options on the command line.

Filenames as Objects

Some commands accept filenames as arguments, or "objects," in the command line. `cat`, for example, types out the file contained in its *filename* argument.

Putting the Grammar Together

`ls` is an example of a command with options and filename arguments. When you want to list the files in directory `hinterland`, with the files appearing from most recent file modification to oldest time of modification, use the `-t` option to `ls`:

Figure 5-6 *Command with Option and Filename Argument*

```
venus% ls -t hinterland
north.america   asia
venus% ls hinterland
asia            north.america
venus%
```

Because you created the directory `north.america` after creating the directory `asia`, `north.america` appears first in the file listing when using the `-t` option, and last in the file listing without the option.

To make it absolutely clear, `ls` is the *command*, `-t` is the *option*, and `hinterland` is the *filename* (for a directory) in the above example. Options usually appear before other arguments in the command line.

5.2. Wild Card Characters

At the end of Chapter 4 on files and the file system, you learned abbreviations for the home directory, the working directory, and the parent directory. Sometimes, you can abbreviate filenames too, using certain special characters called *wild card* characters.

The most common wild card characters are: `?` and `*`. `?` is the wild card character representing a one-character value; `*` represents an arbitrary number of characters.

For example, when you want to list all of the files that begin with any two characters and end with a certain *string*, or character combination(in this case `tors`):

Figure 5-7 *Use of ? Wild Card Character*

```
venus% cat > tutors
Socrates
Voltaire
venus% ls
actors       alaska     hinterland   tutors       wilde
venus% ls ??tors
actors  tutors
venus%
```

When you want to list all of the files that contain the string `la`:

Figure 5-8 *Use of * Wild Card Character*

Note: `ls` labels the files it lists in the directory `hinterland` with the directory name, so that there is no confusion about the listing.

```
venus% ls
actors       alaska     hinterland   tutors       wilde
venus% ls *la*
alaska

hinterland:
asia            north.america
venus%
```

These wild card characters are especially useful when you have to search through large directories full of files.

5.3. Redirecting Output

When you created your first file in the section of Chapter 4 on creating files, you used a technique called *redirecting output*, or sometimes *output redirection*, without really knowing what you were doing.

You typed `cat > alice` followed by the quotation you inserted into the file. By typing the > symbol, you ''redirected'' the output from the `cat` command into the file `alice`.

In other words, when you type > after a command, you redirect the output of that command into the filename after the > symbol. You're telling the system to take what a command would have typed out to the screen and put it into a file instead.

You can also redirect the output of `spell`, a program that checks your spelling. First, create a file full of spelling mistakes with `cat` by redirecting the output of `cat` to place it in the file `mispell`. Then, run `spell` on the file `mispell`,

redirecting the output from `spell` into the file `errors`.

Figure 5-9 *Spelling Correction and Redirecting Output*

```
venus% cat > mispell
"In ordinary composition, use orthodox spelling.  Do not
write nite for night, thru for through, pleez for please,
unless you plan to introduce a complete system of simplified
spelling and are prepared to take the consequences."
venus% spell mispell > errors
venus% cat errors
nite
pleez
thru
venus%
```

`spell` finds the spelling errors in a file and types them in alphabetical order.[25] As in the above example, redirecting the output from `spell` into a file allows you to store the spelling errors until you no longer need to refer to them.

Be careful when redirecting the output of a command to a file. When you redirect output onto a file that already exists, the output redirection will remove the current contents of that file and replace them with the redirected output. You could lose some valuable work.[26]

Figure 5-10 *Caution: Redirecting Output Onto an Existing File*

```
venus% ls
actors          alaska        hinterland      tutors
venus% cat tutors
Socrates
Voltaire
venus% date > tutors
venus% cat tutors
Sat May 30 23:00 GMT 1778
venus%
```

[25] This file is a quotation from *The Elements of Style*, third edition, by William Strunk, Jr., and E.B. White.

[26] To prevent redirected output from "clobbering," or replacing the contents of your files, see the section on the `noclobber` environment variable in *Setting Up Your UNIX Environment: Beginner's Guide*.

Appending Redirected Output

You can avoid "writing over" the contents of a file by *appending* output onto the file, rather than replacing the file. So, if you want to add the date to the end of a file, use the append redirecting output symbol >> .

Figure 5-11 *Appending Output to a File*

```
venus% ls
actors       alaska     hinterland   tutors       wilde
venus% cat actors
James Dean
Humphrey Bogart
venus% date >> actors
venus% cat actors
James Dean
Humphrey Bogart
Mon Sep 14 10:00 PST 1936
venus%
```

When you append output to a file that doesn't exist, the append output symbol, >>, acts exactly like the first redirecting output symbol, >, creating the file and redirecting the command output into it. Try duplicating the example in Figure 5-9 , describing spelling correction and redirecting output, by removing the file `error`, then replacing the redirecting output symbol, > , in the example, with the append output symbol, >> .

5.4. Interactive Programs

So far, the distinction between a *command* and a *program* is blurry. But one kind of program, called an *interactive program*, is never called a command.

An interactive program is a program that expects further instructions after it begins executing.

Note: `bc` may be picky about requiring spaces around arithmetic operators.

`bc`, the basic calculator program, is an interactive program. To begin execution of `bc`, type `bc`. Then, type basic arithmetic expressions, each followed by RETURN, and `bc` will type out the result of the arithmetic calculation. Use any of the following *arithmetic operators*: + for addition, - for subtraction, * for multiplication, and / for division. The RETURN acts as an equals sign. To *quit*, or stop execution of `bc`, type CTRL-D on a line by itself.

Figure 5-12 *Basic Calculations with* `bc`

```
venus% bc
3 + 7
10
3 - 7
-4
3 * 7
21
3 / 7
0 (To quit, type CTRL-D on next line.)
venus%
```

As you can see, `bc` types out results with a certain number of decimal places. When you want `bc` to type results with more decimal places, type `scale =` followed by the *number* of decimal places you want.

Figure 5-13 *Changing Scale in* `bc`

```
venus% bc
3 / 7
0
scale = 5
3 / 7
0.42857 (To quit, type CTRL-D on next line.)
venus%
```

`bc` offers a variety of other mathematical functions, base conversion, and variable assignment and manipulation.[27]

You've learned a lot about commands in this chapter, and you've encountered interactive programs. One of the most frequently used interactive programs on the system is `mail`. See *Mail and Messages: Beginner's Guide* for more information. Another widespread interactive program is `vi`, the UNIX text editor, described in the next chapter.

[27] For information on the other capabilities of `bc`, see *Games, Demos, and Other Pursuits: Beginner's Guide* and the *Commands Reference Manual* entry for `bc`.

Getting Started With UNIX: Quick Reference

This quick reference lists the commands presented in this manual concisely by function. Each listing includes a syntax diagram, and a brief description of the command.

1. Work Session

1.1. Log In

Type username to system login prompt.
Type password to password prompt.

1.2. Change Password

Type `passwd`, followed by old password, and repeat new password.

1.3. Log Out

Type `logout` or CTRL-D depending upon system setup.

2. File System

2.1. Create File

Type `cat >`*filename*, then text ending with CTRL-D, or see **Editing Files**.

2.2. Make (or Create) Directory

Type `mkdir` *directory-name*.

2.3. Look at File

Type `cat` *filename*
or `more` *filename*.

2.4. Print File

Type `lpr` *filename*.

2.5. List Files and Directories

Type

`ls` for listing of current directory

`ls` *directory-name*
for listing of another directory

`ls` *filename*
for listing of a single file

`ls -t` or

`ls -t` *filename* or

`ls -t` *directory-name*
to get a listing reverse sorted by time of last modification

`ls -F` or

`ls -F` *directory-name*
to get a listing that marks directory names by appending a / character to them.

2.6. Move (or Rename) Files and Directories

Type

`mv` *source-filename destination-filename*
to rename a file

`mv` *source-filename destination-directory*
to move a file into another directory

`mv` *source-directory-name destination-directory-name* to rename a directory, or move it into another directory.

2.7. Copy Files

Type

`cp` *source-filename destination-filename*
to copy a file into another filename

`cp` *source-filename destination-directory*
to copy a file into another directory

2.8. Remove (or Delete) File

Type

`rm` *filename*
to remove a file

`rmdir` *directory-name*
to remove an empty directory

`rm -r` *directory-name*
to remove a directory and its contents.

2.9. Change Working Directory

Type

`cd` to change directories to your home directory

`cd` *directory-name*
to change directories to another directory.

2.10. Find Name of Current Directory

Type `pwd`.

2.11. Pathnames

simple:	One filename or directory name to access local file or directory.
absolute:	List of directory names from root directory (first /) to desired filename or directory name, each name separated by /.
relative:	List of directory names from current position to desired filename or directory name, each name separated by /.

2.12. Directory Abbreviation

~	Home directory.
~*username*	Another user's home directory.
.	Working directory.
..	Parent of working directory.

3. Commands

3.1. Date and Time

Type `date`. For universal time (Greenwich Mean Time), type `date -u`.

3.2. Calendar

Type

`cal` *year*
for yearly calendar

`cal` *month-number year*
for monthly calendar

3.3. Wild Cards

`?` Single character wild card.

`*` Arbitrary number of characters.

3.4. Redirecting Output

System types output of command to file rather than screen, replacing current contents of file, if any. Type *command-name* > *filename*.

System types output of command to file rather than screen, appending to current contents of file, if any. Type *command-name* >> *filename*.

3.5. Basic Calculator

Type `bc` to enter interactive program. Type arithmetic expressions, using `+`, `-`, `*`, and `/` symbols, followed by [RETURN]. To change number of decimal places, type `scale` = *number*.

4. Editing Files

Type `vi` to enter text editor, then any of following commands (in command mode, unless preceded by a `:`):

`a` to add text

`cc` to substitute a line with a string (enters insert mode)

`cw` to substitute, or change, a word with a string (enters insert mode)

`dd` to delete the entire line the cursor is on

`dw` to delete the word, or portion of word, under and after the cursor

`h` to move left, or "west," one character

`i` to insert text under the cursor (enters insert mode)

`j` to move down, or "south," one line

`k` to move up, or "north," one line

`l` to move right, or "east," one character

`o` to insert text on a new blank line after the current line (enters insert mode)

`O` to insert text on a new blank line before the current line (enters insert mode)

`s` to substitute a character with a string (enters insert mode)

`x` to delete the character under the cursor

`:q` to quit `vi`

`:q!` to quit `vi`, without writing changes

`:w` to save, or write a file

5. Formatting Files

Construct source file to run through `nroff` formatter, including any of the following commands:

`.LP` to *left-justify* a paragraph

`.IP` to created an *itemized* paragraph (like this one)

`.ce` to center text on the page

`.ul` to underline portions of text

`.sp` to create a blank line space

`.br` to force the end of a line, a line break

To format the source file, type `nroff -ms` *source-filename*. You will probably want to redirect the output of `nroff` into a *destination-filename*, so you can print it out afterward.

6. Search Files

Type

`grep` *search-string filename*
to type out lines containing the string in a specific file

`grep` *search-string filename(s)*
to type out lines containing the string in more than one file

`grep -v` *search-string filename(s)*
to type out lines that **don't** contain the string

7. Timesavers

7.1. Aliases

To "alias," or abbreviate a command string with an alias string, type `alias` *alias-string command-string*.

8. History: Command Repetition

`!!` Repeat the entire last command line at any point in the current command line.

`!$` Repeat the last word of the last command line at any point in the current command line.

9. Run Command in Background: Job Control

To run a command in the background, as opposed to the more common method of running commands in the foreground, type a `&` after the command line. Then, you can type more commands to the command prompt, or even run more commands in the background for simultaneous command execution.

10. Online Documentation

To see online Man Pages, type `man` *command-name*.