Sheridan

SYST13416
Introduction to Linux
Operating

**Linux Command
Line Environment**

LINUX WORK ENVIRONMENT

Objectives

- Understand what is the Unix account
- Describe what your account might be if created by SA
- Understand the significance of passwords
- Choose a strong password
- Describe what other account information is created with an account
- Log in to your Linux account remotely

Linux Work Environment

- Computer responds to correct commands.
- Command line is parsed by the shell into individual tokens and checks the syntax
- If you use correct semantics, the shell will produce meaningful results

Syntax
- spelling and order

Semantics
- usage and meaning

Example
- The following command displays the content of the file called "filename" on the monitor
  more filename

- An example of a **syntax** error would be to type the command as follows:
  mroe filename

- An example of a **semantic** error would be if the content of the file "filename" is not viewable.

The Unix Account

- In one respect, UNIX is like your local bank; just as you need a bank account to withdraw money from a bank, you also need an account to use a UNIX computer. Every bank has a manager who is responsible for setting bank policy, opening and closing customer accounts, balancing the books, and generally overseeing that the bank operates smoothly. Similarly, most UNIX installations have a system administrator (SA), who sees that the system runs smoothly. The responsibilities of an SA include
  - Setting up the hardware;
  - Installing software, including the operating system;
  - Starting up and shutting down the system;
  - Monitoring system usage;
  - Backing up users' files;
  - Creating and managing user accounts; and
  - Troubleshooting.

Your Account

- A large installation may serve hundreds or thousands of users, each having their own account. To identify these accounts, each is given a unique name, which is typically based on some way on the user's real name or nickname. For example, the following would all be likely account names for a user named John P. Jones:
  - jpjones
  - jonesjp
  - jones
  - johnp
  - jpj
  - sparky

[Module 2: Linux Command Line Environment] – Page 3 •

- The policies for assigning account names vary. The following rules are fairly typical:
  - ✓ Select a name that is at least three characters long, but no more than eight.
  - ✓ Be sure that no other account has the same name.
  - ✓ Choose an account name that is related in some way to the real name or nickname of the person.
  - ✓ Use only numbers and lowercase letters; do not include uppercase letters or punctuation.
- When you start a work session on the computer, you will be asked to log in by giving your account name:
  login:

- Because you use your account name when logging in, it is often referred to as your **user account**, **user ID**, **username** or **login name** or **login**.

Your Password

- Your login name is public knowledge. To prevent unauthorized use of your account, the computer also asks you for a secret password:
  password:

- By entering your password, you verify to the computer that you really are the person to whom the account belongs. The SA will probably choose your first password for you. After that, you can, and should, choose your own. A good password is one that is easy for you to remember, but difficult for someone else to guess. Here are some good general guidelines for selecting a password:
  - ✓ Choose a password that is at least six characters long. (Passwords can be as long as you like, but some systems only examine the first eight characters.)
  - ✓ Combine numbers, uppercase and lowercase letters, and punctuation.

"What do you mean Rumpelstiltskin is too long for a password?!"

[Module 2: Linux Command Line Environment] – Page 4 •

✓ Make the password memorable, but avoid common names, and any words that might be found in a dictionary.
✓ Do not use your social insurance number, telephone number, your login name, or any variation of your login (forward and backward).
✓ Make sure a new password differs significantly from the old one.

- A number of strategies exist for choosing a password. One is to misspell an easily remember word or name. Another is to form a password from the first letter of each word in an easily remembered phrase, including punctuation. For example:
  ✓ *NiMyyd!*   (Not in My yard you don't!)
  ✓ *Ihnybtf*   (I have not yet begun to fight)
  ✓ *H.hotr*   (Home, home on the range)
  ✓ *wtdatap*   (where the deer and the antelope play)

- Your password is the main line of protection for your account. Anyone who discovers your password can do nearly anything to your account, including deletion of your files. Therefore, it is extremely important that you keep your password secure.

Other Account Information

- When your account is created, the system administrator sets up the following information in addition to the login name and password:
  ✓ **Home Directory.** Your home directory is the place where all of your other files and directories reside. The home directory has a name, which is the same as your login name.
  ✓ **Group ID.** You may be assigned to a group of users. In some organizations groups are set up so that users in the same department or working on the same project can easily share files.
  ✓ **Login shell.** The system administrator will select a shell to start up automatically whenever you log on.

Sheridan

to Linux Operating

SYST13416
Introduction to Linux
Operating

The Essence of Unix

HUMAN-COMPUTER INTERACTION

Objectives

- Understand the work environment: terminal window, virtual console, remote access, the keyboard

Linux Work Environment

- There are two different Unix interfaces, text-based and graphical. The graphical user interface (GUI) is created by a combination of X Window, a window manager, and a desktop environment.
- The text-based interface is usually referred to as the command line interface (CLI). The basic text-based interface is simple. The shell (command processor) displays a prompt. You type a command. The shell does what's necessary to carry out the command. Once the command has been processed, the shell displays another prompt, you type another command, and so on.
- This process uses only text (plain characters), and the line on which you type your commands is called the command line. The term CLI is a counterpart of GUI. Often, when people talk, they don't say CLI, they say "command line".

Terminal Windows

- Within your desktop environment, all your work is done within windows, so it only makes sense that you would run a terminal emulation program within a window.
- All desktop environments come with a simple way to start a terminal program. (Gnome: look in the System Tools submenu for Terminal. KDE, look in the System submenu for Terminal or Konsole.) These programs will appear as a window on your screen. Within that window you have a standard CLI.

Virtual Consoles

- When you start Linux GUI desktop environment, Linux starts seven different terminal emulation programs, called virtual consoles. The first six are full-screen, text-based terminals for using a CLI. The seventh virtual console is a graphics terminal for running a GUI. When you are looking at your graphical desktop environment, you are looking at virtual console #7, the other six are invisible.
- To switch from one virtual console to another, use the key combinations Ctrl-Alt-F1 through Ctrl-Alt-F6 for corresponding console #1 through #6. To return to the GUI, press Ctrl-Alt-F7.
- When you switch away from a virtual console, the program that is running in it keeps running.

- The text-based virtual consoles use the entire screen and display characters using a mono-spaced font typeface.
- If you ever have a serious problem with your desktop environment, it's handy to be able to switch to a virtual console in order to solve the problem. For example, if your GUI freezes, you can usually press Ctrl-Alt-F1 to get to another terminal. You can then log in to fix the problem or reboot the system.

The One and Only Console

- One of the terminals, the console, is special because it was used by the system administrator to manage the system.
- Physically the console looks like other terminals, but there are two things that make it different. First, when Unix needs to display a very important message, that message is sent to the console. Second, when the system is booting into single-user mode, for maintenance or problem solving, only the console is activated.

Selecting and Inserting

- X Window allows you to select and insert text from one window to another, or within the same window.
- Many GUI-based programs support the Windows-type copy/paste facility (^C and ^V)

Remote Access

- You may be working directly with your Unix account using a terminal, a personal computer emulating a terminal, or a Unix workstation.
- When you are working on your Unix account remotely, usually you require additional hardware and software. Ask your Unix system administrator or Internet Server Provider (ISP) about remote access.

The Keyboard—Teletype Terminal

- Unix has been designed as a system in which people used terminals to access a host computer. Each of the terminals has its own characteristics and uses its own set of commands. For example, all display terminals have a command to **clear** the screen.
- Each terminal has its own name. The command to display the name of your terminal is **tty**. Whenever you read about printing data, it mostly refers to displaying data. When you give the command tty, it will display the internal name of your terminal.
- The **stty** ("set tty") command is used to display or change the settings of your terminal.
- The **getty** ("get tty") program is used to open communication with a terminal and start the login process
- The keyboard of a teletype contained keys for the 26 letters of the alphabet, the digits 0-9, the most common punctuation symbols, and a few special keys that were used to provide the functions necessary to send and receive messages. The most important such keys were <Esc>, <Ctrl>, <Shift>, <Tab>, and <Return>. Combined with other keys, you can send a signal such as <Ctrl-A> and so on.
- Thompson and Ritchie wrote Unix so that certain *signals* could be used to control the operation of a program as it was running. For example, the signal called *intr* (interrupt) is used to terminate a program. To send the intr signal, you pressed <Ctrl-C>.

Accessing Your Account

- If you make a mistake while typing, press either **Backspace** or **Delete** to erase your mistake. If neither key works, try **^H**, the control sequence for Backspace.
- Pressing **^C** usually cancels a login attempt.
- If you get the error message *incorrect password*, your login attempt failed. Try logging in and out again, perhaps you type something wrong.
- A common mistake is not realizing that **Caps Lock** is on.

Example

```
stty -a
speed 38400 baud; rows 24; columns 80; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; swtch = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R;
werase = ^W; lnext = ^V; flush = ^O; min = 1; time = 0; …
```

Sheridan

to Linux Operating

SYST13416
Introduction to Linux
Operating

**The Shell
Environment**

COMMAND LINE INTERFACE

Objectives

- Understand your shell environment
- Command line elements
- How Linux processes data
- How shell processes commands and handles errors
- Get help with man, apropos, whatis, info
- Utilities: cal, who, date, history

Command Line Interface (CLI)

Command Prompt: Once you log onto your computer network using Unix/Linux, you are running a shell which is a program that interprets your instructions for the Unix kernel. The shell prompts you for command entry.

Command line: Commands activate the shell to perform various tasks. Everything that follows the prompt forms the command line—your input to Unix. You can think of a command line as a sentence of instructions. Instead of ending your sentence with punctuation, you terminate command-line input by pressing **Return** or **Enter**. After entering the command line, Unix processes your instructions, reporting the results of its actions as **output**.

COMMAND LINE RULES! WHY?

From a system administration point of view, the command-line really does rule. It is always available. It is flexible. It enables you to work comfortably on remote systems thousands of miles away even over a slow modem line. Let's face it; all those graphics are a killer on a slow network connection.

Here's one reason that demands a quick bit of history. Friendly administrative interfaces have been around a lot longer than pretty, front-end GUIs and graphical desktops. Way back when, in the early days of UNIX system administration, there were plenty of menu-driven interfaces to admin tools. This was all very nice and it gave even a novice administrator the means to get the job done without being an expert. The **real power** through, comes from a real understanding of what is happening under the surface, below that friendly menu or graphical interface.

Learning to wield the command line is akin to getting a black belt in a martial art or earning a first aid certificate. It's not suddenly having the means to crush any obstacle (or opponent) that comes along, but rather having the confidence and the knowledge that you can protect yourself or handle an emergency when it arises. *The command line is power and it always there for you.*

Interactive Shell Use

When you use the shell interactively, you engage in a **login session** that begins when you log on and ends when you type exit or logout.

During a login session, **you type command lines to the shell**; these are lines of text ending in RETURN that you type in to your terminal or workstation.

By default, the shell prompts you for each command with an information string followed by a dollar sign, though the entire prompt can be changed.

[Module 2: Linux Command Line Environment] – Page 11 •

Elements of the Command Window

• Assuming that you have succeeded in logging in, Unix awaits your instructions. You may think that you've "logged into Unix", but in reality, you have logged onto your computer network using Unix. You are running a shell, which is a program that will interpret your instructions. It acts as an interface between the user and the Unix system kernel. It also acts as a command line interpreter. It interprets typed requests (**commands**), and then spawns and executes them.

Shell Prompt (Command Prompt, System Prompt)

• At this point, you should see a shell prompt, which is simply the shell's way of telling you that it is ready to receive your instructions. If your login shell is the C shell or TC Shell, the prompt is probably a percent sign:
%

• If you are using the Bourne Shell, Korn Shell, or Bash, (these belong to the same family) the usual prompt is the dollar sign:
$

• Other symbols are occasionally used for shell prompts, including the pound sign (#), the "greater than" sign (>), the asterisk (*), the "at" or "ampersand" symbol (@), and the colon (:). Some systems are set up to include the host name as part of the prompt, like this:
atlas %

• In this illustration, the shell prompt is the > symbol, which is commonly used by the C shell family:

Shell

Shell is a language with its own syntax and semantics.

[Module 2: Linux Command Line Environment] – Page 12 •
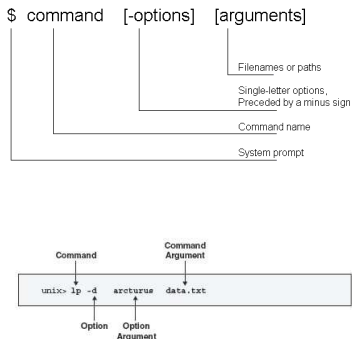
Cursor

- Cursors are symbols that indicate where to type commands after the prompt. Often your cursor might appear as a rectangle. To enable text entry, just point and click inside the desired window, which is called setting the focus.
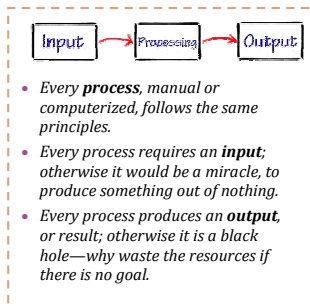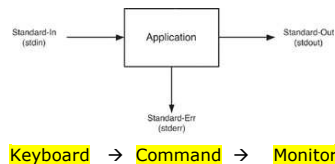
Command Line

- Everything that follows the prompt forms the command line. You can think of the command line as a sentence of instructions. Instead of ending your sentence with punctuation, you terminate command-line input by pressing Return or Enter. After entering the command line, Unix processes your instructions, reporting the results as output.

```
$  command    [-options]    [arguments]
                                Filenames or paths
                                Single-letter options,
                                Preceded by a minus sign
                                Command name
                                System prompt
```

Command-line Syntax

- Unix has some general rules that you should heed:
  - Unix is case sensitive! It matters whether you use uppercase or lowercase letters.
  - Separate command-line elements with spaces! Enter spaces with Spacebar. Use as many spaces as you wish.
  - Dashes and underscores are different!

- In general, A Unix command has the form
  command options arguments

- If the command seems complicated, note the following rules:
  - Unix reads the command line from left to right
  - The hyphen (-) indicates to Unix that an option appears in the command line. Do not separate the hyphen and the option.
  - Option arguments follow the option.
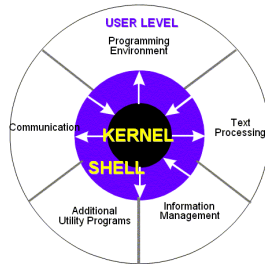 - The command argument is the last item in the command line.

```
       Command          Command
                         Argument
      unix> lp -d   arcturus   data.txt
         Option  Option
                 Argument
```

How Linux processes data

```
Input  →  Processing  →  Output
```

- Every **process**, manual or computerized, follows the same principles.
- Every process requires an **input**; otherwise it would be a miracle, to produce something out of nothing.
- Every process produces an **output**, or result; otherwise it is a black hole—why waste the resources if there is no goal.

- receives input from the standard input device—**Standard in (STDIN)** is where the system expects to find its input. This is usually the keyboard, although it can be a program or shell script. When you change that default, you call it redirecting from STDIN.
- sends output to the standard output device—**Standard out (STDOUT)** is where the system expects to direct its output, usually the terminal screen. Redirection of STDOUT is at the discretion of whatever command or script is executing at the time.
- STDIN is often referred to as **fd0**, or file descriptor 0, while STDOUT is usually thought of as **fd1**.
- the third is standard output error—There is also **standard error (STDERR),** where the system reports any errors in program execution. By default, this is also the terminal. To redirect STDOUT, use the greater-than sign (>). As you might have guessed, to redirect from STDIN, you use the less-than sign (<). But what exactly does that mean?
- The chain of events from STDIN to STDOUT looks something like this:

```
Standard-In          Application          Standard-Out
  (stdin)          →                →        (stdout)
                         │
                         ↓
                      Standard-Err
                        (stderr)
```

Keyboard → Command → Monitor

Operating System Tasks

The operating system tasks, in the most general sense, fall into six categories:

- **File system management**
- **Process management**
  - Ensuring that each process and application receives enough of the processor's time to function properly.
  - Using as many processor cycles for real work as is possible.
- **Memory storage and management**
  - Each process must have enough memory in which to execute, and it can neither run into the memory space of another process nor be run into by another process.
  - The different types of memory in the system must be used properly so that each process can run most effectively.
- **Device management**
- **Application interface**
- **User interface**

Classification of Linux Utilities (commands)

- **File processing** - afio, awk, cat, cmp, comm, cp, cpio, cut, dd, diff, dump, fdformat, find, fmt, grep, groff, gzip, head, ispell, less, ln, lpr, ls, man, mkdir, mkfs, mv, od, paste, pr, pwd, rdev, restore, rm, rmdir, sed, sort, tail, tar, touch, uniq, wc
- **System status** - chgrp, chmod, chown, date, df, du, file, finger, free, ed, quota, kill, ps, sleep, top, w, who
- **Network** - ftp, ifconfig, netstat, ping, rcp, rlogin, rsh, rwho, showmount, telnet, wvdial
- **Communications** - mail, mesg, pine, talk, wall, write
- **Programming** - configure, gcc, make, patch
- **Source code management** - ci, co, cvs, rcs, rlog
- **Miscellaneous** - at, atq, atrm, batch, cal, crontab, expr, fsck, tee, tr, tty, xargs

---

Operating system utilities let you

- create and manage files
- run programs
- produce reports
- generally interact with the system
- access a full range of services to monitor and maintain the system and recover from a wide range of errors

How the Shell Processes Commands

1. **The shell displays a prompt symbol on the screen**. The prompt tells you the shell is ready to receive your command.
2. **You type in a command**. As you type, the shell stores the characters and also echoes them back to the screen.
3. **You type return (Enter key).** This is the signal for the shell to interpret the command and start working on it.
4. **The shell interprets your command**. In most cases, the shell looks for the appropriate software to run your command. If it can't find the right software, it gives you an error message; otherwise, the shell asks the kernel to run it.
5. **The kernel runs the requested software**. While the command is running, the shell "goes to sleep". When the kernel finishes, it "wakes up" the shell.
6. **The shell displays a prompt symbol.** This indicates that the shell is once again ready to receive your command.

Handling Errors...

7. Occasionally, you may redirect standard output but find that the program still prints something on the terminal. This is standard error (stderr), an additional output stream for diagnostics and debugging. Try this command, which produces an error:
   ```
   ls /fffffffff > f
   ```
8. After completion, f should be empty, but you still see the following error message on the terminal as standard error:
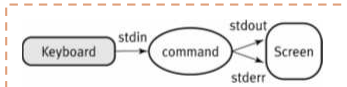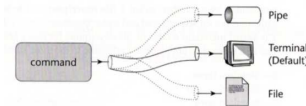   ```
   ls: /fffffffff: No such file or directory
   ```
9. You can redirect the standard error if you like. If you want to send standard output to f and standard error to e, use the following command:
   ```
   ls /fffffffff > f 2> e
   ```

   The number 2 specifies the stream ID that the shell modifies. Stream ID 1 is standard output (the default), and 2 is standard error.
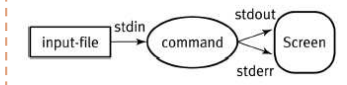
## Redirection





**cal 2009**

- *By default, the shell expects stdin from the keyboard, and sends both stdout and stderr to the scree.*



**mail jones  < my message**

- *By default, the shell expects stdin from the keyboard, and sends both stdout and stderr to the scree.*

## Handling stdin and stdout

10. The standard output is usually the terminal screen (monitor).  However, the shell allows you to redirect the standard output so that th eoutput goes into an ordinary file instead. For example, to use the **redirection operator (>)** to create a file containing a calendar:
```
cal 2007 > calendar.file
```

11. You can add output to a file using the **append operator (>>).**
```
cal 2008 >> calendar.file
```

12. It is also possible to redirect the standard input so that a process takes its input from an ordinary file rather than the keyboard. For example, the mail command allows you to send and read electronic mail. To send the contents of the file my.message to the user jones, you could enter the command line:
```
mail jones < my.message
```

13. You can combine input and output redirection in the same command line. For example, the following command line invokes the wc utility to count the lines, words, and characters in the file input, then redirects results into the file output.
```
wc < input > output
```

14. Now type the cat command and redirect its STDOUT to a file called "randomNames"
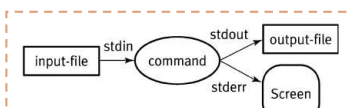```
cat > randomNames
```

15. Your cursor will just sit there and wait for you to do something, type the following names, pressing <mark>Enter</mark> after each one.
    Marie Curie
    Albert Einstein
    Mark Twain
    Wolfgang Amadeus Mozart
    Stephen Hawking
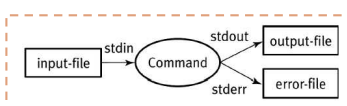   **Isaac Newton**
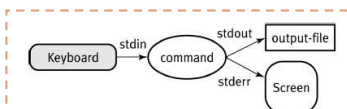
cal 2009 > calendar.file

- *Redirects the stdout to a file.*

ls /nosuchdir   2> error.file

- *Redirects the stderr to a file.*

wc < infile > outfile 2> err.file

- *Redirects stdin from a file, stdout to a file, and stderr to a file.*

What's happening here is that cat is taking its input from STDIN and writing it out to your new file. You can also write the command like this:
cat - 1> randomNames

The hyphen literally means standard in to the command. The 1 stands for file descriptor 1. This is good information, and you will use it later. Finish the random names list and press CTRL+D to finish. **CTRL+D, by the way, it stands for EOF, or end of file.**

Handling pipes

17. Rather than sending stdout into a file, you can redirect the output of one command as the stdin of the next command. To piple the output of command1 into command2, use the syntax
```
command1 | command2
```

18. A common use of piping is paging through output:
```
ls –R / | less
```

19. Note that the output of first command becomes the input of the second command, and the output of the second command becomes the input of the third command, and so on.
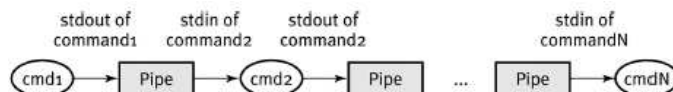
MS-DOS vs. Linux

MS-DOS	Linux / Unix
attrib	chmod
backup	tar
dir	ls
cls	clear
copy	cp
del	rm
deltree	rm -R rmdir
edit	vi pico
format	fdformat mount umount
move / rename	mv
type	less <file>
cd	cd chdir
more < file	more file
md	mkdir
win	startx

Grouping Commands

20. Normally you type one command at a time, following each command by a return (Enter key), which is the signal for the shell to begin its work. However, it is possible to put multiple command son the same line, if you separate the commands with semicolons. Thus the command line
```
who; ls; cal
```
Has the same effect as the three separate command lines
```
who
ls
cal
```

21. Grouping commands can be especially useful when you want to redirect the output to a file. For example,
```
cal 6 2008 > summer.2008
cal 7 2008 >> summer.2008
cal 8 2008 >> summer.2008
```
You can accomplish the same thing with just one line:
```
(cal 6 2008; cal 7 2008; cal 8 2008) >> summer.2008
```
Note: you use the parentheses to redirect all three calendars into the same file, otherwise only the August calendar would be redirect into the file

[Module 2: Linux Command Line Environment] – Page 19 •

Redirecting to multiple targets

22. The output from a command resembles flow through a pipe. Real plumbing pipes have all sorts of shapes. The T-pipe forms a junction that splits flow into vertical and horizontal flows. With Linux, use the tee command for similar purpose. The tee command stores intermediate results, enabling the output to be sent to both the standard output and an output file or files. To see and store active users, you can give the command
```
who | tee users.txt
```
If you wish to append to a file, use the –a option:
```
who | tee –a users.txt.
```

[Module 2: Linux Command Line Environment] – Page 20 •

Getting Help

So you want to know more about the who command…

- All Unix commands are documented online. You can specify which command you want the manual page to view:

```
man who
```

- See what happens if you don't specify any command:

```
man
```

- To learn more about the online manual, try the following:

```
man man
```

- Spend some time browsing through the file and the information that it provides about the Online Manual. When you are viewing the manual pages, you will notice that you are viewing only one page at a time. To move to the next page, press the [**spacebar**]. To go back to the previous page, press [**b**] key, and to quit and return to the command prompt, press [**q**].
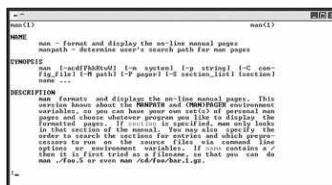
- Note the different levels of pages provides for different types of users. Try

```
man 5 passwd
```

- What if you know what you want to do, but you ar not sure which command to use? The solution is to use man with the –k option (the k stands for "keyword"). For example:

```
man –k manual
```

- Note that all pages have the same format and organization. Each page contains the following sections: NAME, SYNOPSIS, DESCRIPTION, FILES, SEE ALSO, DIAGNOSTICS, BUGS, and AUTHOR

BE WARNED!

- *A common way of getting help is finding someone who knows, and however patient and peace-loving your social or work community will be, almost everybody will expect you to have tried to explore available resources before asking them, and the ways in which this viewpoint is expressed may be rather harsh if you prove not to have followed this basic rule.*

Organization of a manual page.

Section	Description
NAME	The name of the command and a brief description
SYNOPSIS	How to use the command and command-line options
DESCRIPTION	An explanation of the program and its options
FILES	A list of files used by command, and their location
SEE ALSO	A list of related man pages
DIAGNOSTICS	A description of unusual output
BUGS	Known problems
AUTHOR	The program's main author and other contributors

- As you work through the course, you will use a number of new commands, familiarize yourself with their pages, even if it does not specifically instructs you. Don't worry if you don't understand it all. The more you use it, more will make sense. Remember, PRACTICE MAKES PERFECT!

```
man history
man date
man cal
```

Other Commands to Get Help

- Getting command summaries with **whatis** command, a synonym for **man -f**

```
whatis cal
```

- The **apropos** command is a synonym for **man –k**. The word comes from the French expression à propos meaning "related to". In English, "apropos" is a preposition meaning "concerning" or "with reference to".

```
apropos cal
```

- Getting more information with **info** command:

```
info cal
```

INFO is an online help system, separate from the online manual, which is used to document the GNU utilities. Info files contain not just information, but links to other Info files, so reading in Info page is similar to reading a web page in the sense that you can use a link to jump to another file. The man pages will often refer you to Info for more information.

INFO System

Although the Info system is complicated, there are seven really important commands:

[space] — display next screenful
[backspace] — display previous screenful
[tab] — move cursor down to next link
[return] — follow a link
[d] — jump to the directory node
[l] — jump to last node you visited
[q] — quit

Use these commands to get started and commit yourself to learning more as the need arises.

COMMAND cal

On what day of the week were you born?

- The cal command without any option or argument will output the current month's calendar. Try:
  ```
  cal
  ```

- Provide one argument, specifying which year you wish to display:
  ```
  cal 1923
  ```
- Provide two arguments, specifying the month followed by the year:
  ```
  cal 2 1968
  ```
- Examine the manual page for other arguments and options.
  ```
  man cal
  ```

COMMAND who

See if Unix knows who you are.
- Try the command who:
  ```
  who
  ```

- The computer will respond with a list of the users who are currently logged into the system. To display your login name, type:
  ```
  who am I
  ```
- Examine the manual page for other arguments and options.

date displays the current date and time.
A superuser can set the date and time.
Syntax

   date [options] [+format]

Common Options

-u	use Universal Time (or Greenwich Mean Time)
+format	specify the output format
%a	weekday abbreviation, Sun to Sat
%h	month abbreviation, Jan to Dec
%j	day of year, 001 to 366
%n	<new-line>
%t	<TAB>
%y	last 2 digits of year, 00 to 99
%D	MM/DD/YY date
%H	hour, 00 to 23
%M	minute, 00 to 59
%S	second, 00 to 59
%T	HH:MM:SS time

What's today?

- Display the manual page for the command date and read the description and what options are available. Explore some of the options.
man date

- Display the current time and date.
  date

- You can show formatted date and time. Check the manual page for the formatting options.
date +'Date: %m/%d/%Y%nTime: %r'

- On some systems you can get more details on formatting:
man strftime

   If this is not available on your system, use your favourite search engine to browse the Internet.

Accessing your command history

- Perhaps you're having trouble remembering all the commands you entered. Or perhaps you would like to re-enter a command that you're tired of typing. Here is Unix to the rescue!
- Unix will show a list of commands. The length to which Unix remembers is a variable that you can customize—later, wait until you learn how to modify your configuration files.
history

- List last 12 commands
history 12

- Repeat last find command but pipe it to the soft command
!find | sort

- Print without executing event number 24
!24:p

- Redo command line 10 assuming that history shows at least 10 commands.
!10

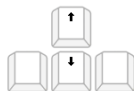- Try the following command. Note what the result is. Use the man page to confirm your findings.
history 7

- Try the following command. Note what the result is. Use the man page to confirm your findings.
!!

Browsing your command history at the prompt

- Recall the last command by using the up-arrow ↑
- Move through the history back several commands by repeatedly pressing the up-arrow ↑ and see how the commands are recalled on the command line.
- Then use the down-arrow ↓ to return to the end of the history, until the command line is blank.

Contents