A Practical Introduction to Unix

Michael B. Spring
Department of Information Science and Telecommunications
University of Pittsburgh
spring@imap.pitt.edu
http://www.sis.pitt.edu/~spring

Overview

- File system
  - Directory structure
  - Directory and file commands
  - File access control
- Process control
- Commands and Options
- The Shell
  - The different shells
  - Piping, redirection, and variables
  - Aliases and functions

The File System – Physical Disks

- Physically disks are formatted into blocks
  - The first two* blocks of a disk are:
    - bootstrap block: machine language instructions for startup
    - super block: list of available disk resources
  - next comes the inode blocks
  - next come the data blocks
- Additional physical disks can be associated with directories

* In modern Systems, both of these can be multiple blocks

Files

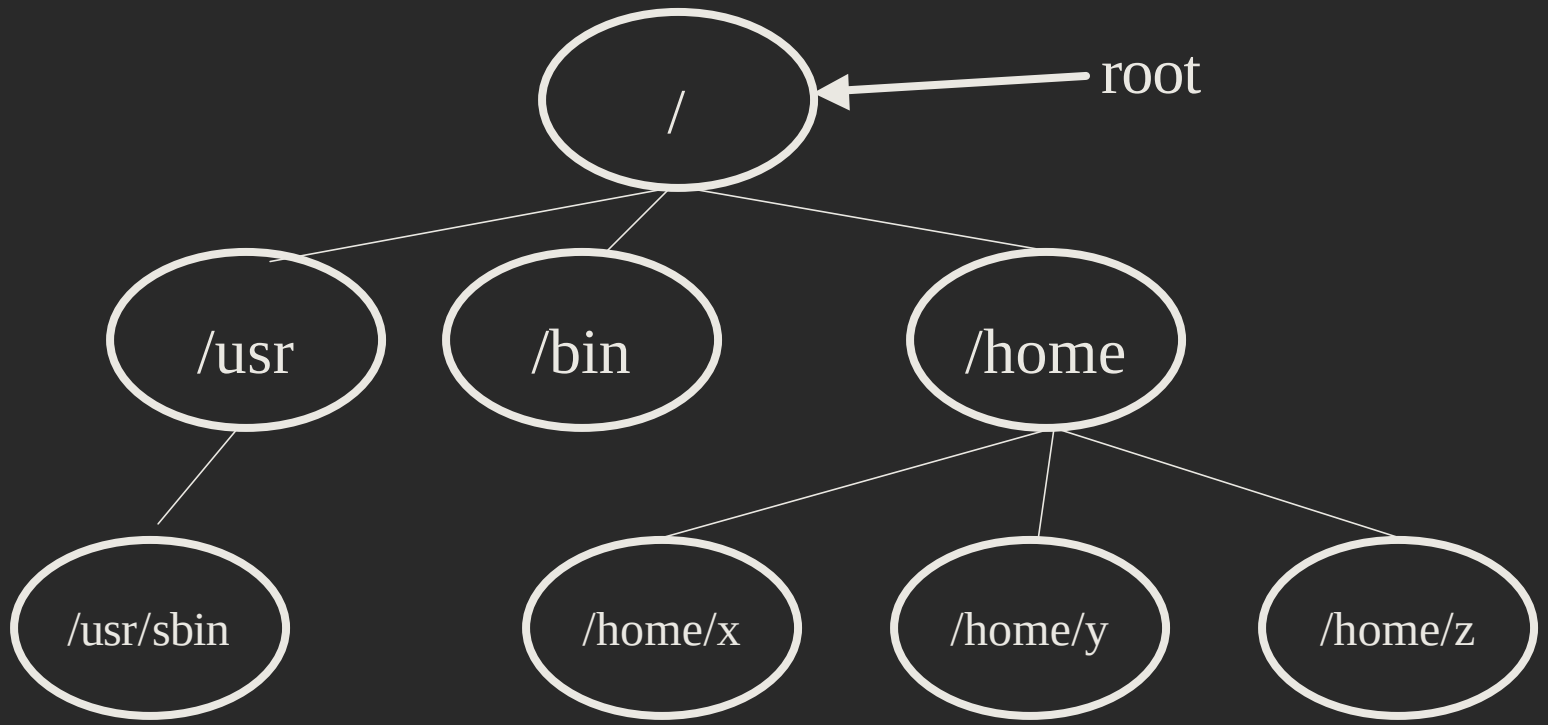
- May be of any size allowed by administrator
- Are not structured in any way by the system
- Files are of multiple types:
  - Ordinary files – be they text or binary
  - Files representing character and block devices
  - Files representing directories
- Security is provided on a file and directory basis
- Devices are treated as special files
  - They provide a level of indirection for devices
  - Read and write to device files just as to ordinary files

Directory files

- Controlled by the OS – not accessible to the user
- Associates a file name with an inode
- inode specifies:
  - owner
  - group
  - type
  - permissions
  - last access
  - last modification
  - size
  - disk addresses
  - last inode modification
- This indirection allows for links (see below)

Hierarchical File System

- Must have a root directory – /
- A user's home directory is ~username



Important Directories

- Some standard directories include
  - /bin – binaries
  - /sbin – system binaries
  - /dev – device files
  - /etc – system admin
  - /home – user file systems
  - /spool – temporary files
  - /var – files that vary in length
  - /usr – binary files (unix system resources)

Special File Structure References

- Directory separator is “/”, not “\” as in DOS
- Absolute and relative directory names
  - Absolute: /home/spring/bin/source
  - Relative, assuming cur. dir. is /home/spring/bin: source
- Special directory references:
  - Root directory: /
  - Home directory: ~
  - Parent directory: ..
  - Current directory: .

Directory Commands

- There are a relatively small set of commands related to directory access and manipulation
  - mkdir makes a directory
  - rmdir removes a directory
  - cd changes to the named directory
  - pwd prints the fully qualified name of a the current working directory
  - ls lists the contents of a directory
- There will be additional commands introduced later

File Commands

- Creating a file
  - Creating a file is easy with an editor
  - Using the “touch” command
  - Using echo with redirection
- Renaming a file is the same as moving it
  - mv source destination
  - if across disks, mv will not work. Copy the file to create a new inode then remove the original file and inode
- Copying files – cp
- Removing files – rm
  - Beware rm –rf – one of many very dangerous Unix commands

Setting File Protection

- Each file, ordinary, directory, or device has a set of protections – access protections
- The protection can be changed using either of two forms – conversational and absolute
- The command is chmod
- Access controls are specified for the:
  - Owner
  - Group
  - World
- Access rights relate to:
  - Reading
  - Writing
  - Executing (X)
- cp mv ln don't change the status of files

Using chmod “conversationally”

- chmod category=+/-function file
  - eg chmod g=+r filename
  - categories are u = user, g = group, o = others
    - NB—others(world does not include user)
  - operators inlcude = + -
  - functions include r,w,x
  - Functions also include
    - t = sticky bit
    - l = mandatory locking

Using chmod “absolutely”

Sticky			Owner			Group			World		
			R	W	X	R	W	X	R	W	X
			1	1	1	1	0	1	1	0	0

- Access rights are actually stored as a 12 bit number.
- Chmod generally accesses 9 of the twelve bits.
- Each of the cells shows which right is enabled – 1
- The rights may be read as the octal equivalent of the binary number. For example:
  - $111_2=7_8$
  - $101_2=5_8$
  - $010_2=2_8$

Using chmod “absolutely”

- The three numbers represent the owner, group, and world
- The three bits of the octal number define protections – the 4 position represents read, the 2 position write, and the 1 position eXecute.
- absolute mode, chmod knnn file
  - first n = owner, 2nd = group, 3rd = other
  - if k = 4, setuid flag, 2 = setgid, 1 = sticky bit
- Thus chmod 700 filename
  - Gives the owner access to all functions and not anyone else

Access to directory files

- chmod may be used on directory files as well, but the meanings are slightly different
  - read allows the files in the directory to be listed
  - write allows files to be added or deleted from a directory
  - eXecute allows traversal of the directory structure assuming that the file name is known--x means search not execute

umask value

- When an ordinary file or directory is created, the system variable umask is used to set protections
  - The default umask value is 077
  - The protection set is the logical opposite of the umask
  - Thus, owner gets all rights, world and group none
- Files created by shell redirection are set to 666 minus umask
  - basically, it removes executable permission

The Process Control System

- Each program run on Unix is a process or task, and has a processID or PID
- Processes start other processes via system calls
- The process which starts the process is the parent
- Unix processes are generally speaking lightweight compared to other systems
- The kernel is the first process that is started
  - The kernel starts a scheduler and some other processes
  - The scheduler starts shells when users login
  - The shell is itself a process

Watching processes

- ps can be used to check processes
- kill pid to kill a process
- Write a simple shell script

```
#! /usr/bin/ksh # or appropriate path
while true
do
   sleep 1;
   echo ‘’hello’’;
done
```

- Chmod on the script and run it in the background
- Use ps to find the process id and kill it

A Simple Command – with options

- The general form of commands is:
  - cmd –options filenames
  - Options are normally specified right after the command name, and normally preceded by a -
  - Where a command requires a filename, filename “metacharacters” are often allowed – I.e. *, ?, and []
- rm removes a file
  - -r removes files recursively
  - -f forces removal
  - -i requests confirmation
  - rm –rf /* is a disaster

Another Simple Command – with options

- ls to list the contents of a directory
  - -a all files
  - -l long listing
  - -R(recursive)
  - -d list directory information, not contents
  - -m merge the list to a comma separated set of names
  - -p mark directories with a /
  - -t list files by modification time
  - -u list files by access time
- ls –ltR *.c recursively lists all c files in the current directory and below in long form

A couple more

- cat is used to display a file
  - -t prints tabs as ^I and formfeeds as ^L
  - -e prints $ at the end of each line
- But there are other options to display a file
  - head, tail to look at the first or last lines of a file
    - -n specifies the number of lines, default is 10
  - more pages through a file
    - -d display a prompt
    - -c display by screen refresh rather than scroll
    - Within more /xyz will search for xyz
  - od produces an octal dump
    - -ha produces a hex and ascii dump

The Shells

- All commands are given through a shell
- In general, shells provide the following services
  - Interactive command execution
  - Process control
  - The ability to tailor your environment
  - The ability to execute programs
  - A variety of special features
    - History
    - Command line editing
- Your “shell” is simply a program.
  - There are many different shells

The Shell and Processes

- Because the shell is a process, it can start other processes
- Many of the commands given in Unix are really directives to the shell to start a child process.
- When a process starts another process, the parent process suspends execution until the child process completes
  - It is possible to direct the shell to start processes in the background
- Processes communicate with each other using a variety of mechanisms:
  - Variables set in the environment
  - I/O Channels
  - InterProcessCommunication

Issuing Commands

- The shell serves as a command interpreter
  - It has some “built in commands” and capabilities
  - Other capabilities are achieved by executing programs – “external commands”
- When the shell forks a process to run the program, it offers three primary process controls:
  - ; - allows commands to be sequenced
  - & - puts a command in the background
  - | - “pipes” the output from one command to another
- Sophisticated grouping of commands is possible

Which shell to use

- The default shell for you is set by the sys admin
- The more common shells include
  - csh
  - sh(bourne)
  - ksh(Korn)
  - bash(bourne again shell)
- You can use any of the shells on your system by simply typing the shell name
- It takes a long time to understand everything your shell can do

Learning about your shell

- The best way to learn about the shell is to use it and to experiment with it
- Books and man pages have to be read
  - man ksh
  - more, less, xman, and xless can also be important aids
- Keep in mind that there is a man on man which will help you to use the man pages
  - /topic when using man allows you to search
  - / by itself repeats the last search

Features Common to All Shells

- Filename metacharacters
  - * allows 0 or more characters to be matched
  - ? Allows 0 or one characters to be matched
  - [ ] allows any from a group of characters to be matched
  - Korn offers +, @, ! as well
- Process control
  - & allows a process to be run in the background
  - ; allows multiple processes on a single line
- Piping and redirection
  - the | pipe command allows process I/O to be tunneled
  - The <,>, and >> redirection commands for file I/O

One Simple Example

- the basic idea behind the Unix tools is that they work best in unison
  - Create a file
    - Type vi
      - Type I
      - Type 5 or six lines of words separated by tabs
      - Type “ESC” – the key
      - Type :w filename
      - Type :wq

A Simple Example Continued

- Now type the file out
  - cat filename
- Now type the file out with a sort
  - cat filename | sort +1
  - This will sort of field 1 – tab separated by default
  - Try sorting on different fields
- Now cut a field out of the sorted file
  - Cat filename | sort +1 | cut 1
- Now save the result as a new file
  - Cat filename | sort +2 | cut 2 > newfile

Std I/O and the Shell

- Each process, the shell being no exception, can have up to 256 file descriptors associated with it (actually, this is system dependent)
- For each process, the first three file descriptors are set by the system – they are automatic:
  - 0=stdin
  - 1=stdout
  - 2=stderr
- To redirect stderr use the I/O channel number:
  - cat filename 2> errors.dat

Variables and the Shell

- Variables play a big role in how the shell works.
- To set a variable, simply type, with no spaces
  - xys=abc
  - xyz is now a variable with the value abc
- Variables that are set within a shell process can be made available to children processes. These variables are called ENVIRONMENT variables
- To create an ENVIRONMENT variable, i.e. make it available to child processes, export the variable
  - export xyz

Some Standard Shell Variables

- All the shells rely on certain variables being defined. These are normally named using all caps. Because they are almost always exported, they are often referred to as “the” ENVIRONMENT variables
  - HOME is your home directory
  - LOGNAME is your login name
  - SHELL is the shell you are running
  - PATH is a : separated list of directories to be searched when you issue a command
  - MANPATH is a : separated list of directories that contain man pages
  - DISPLAY is the IP address and display:screen id of the device to display X Windows

Aliases and Functions

- Aliases can be used to change the shell interface

```
alias dir="ls –l"
alias type=cat
alias print=lpr
```

- Functions can be used to allow for the use of variables

```
function re {
<tab>echo $2;
<tab>echo $1;
}
```

- Re 2 1 will produce 1 2