Introduction to UNIX

Operating Systems

- allocate resources and schedule tasks
- resources include CPU, memory, disk, tape, printers, terminals, modems, etc.
- only one user allowed at a time for each resource
- keeps track of filenames and directory structure
- multi-tasking (one task at a time per processor actually executing)
- multi-processing (schedules multiple processors)
- multi-user

History of UNIX

- developed at Bell Labs (AT&T) in 1969
- unlike most OS's at the time, UNIX was multi-user, interactive, and simplified sharing of data & programs
- became popular in industry as college and university graduates were trained in it
- open system, ported to many different hardware platforms (unlike IBM, Burroughs, Univac mainframes)
- software written for one UNIX system will often run on other systems with little or no modification
- still very weak in areas of system management - tape management, security, hardware accounting, capacity planning tools, performance management including prioritization of jobs
- hardware manufacturers modified UNIX to run on their systems and added enhancements
- standardization was begun in response to Windows NT threat
- System V Release 4 is one of the steps to address standardization
- SVR4 includes many of the modifications that were being done by hardware manufacturers, if they were universally useful and applicable

History of Linux

- GNU (Gnu's not Unix) was started in early 80's to create and promote free software
- Linux was created by Linus Torvalds in the 1990's, released to the Internet in 1994, mostly using GNU C compiler
- about 1/3 of Linux is GNU code from the Free Software Foundation - a Linux distribution consists of Linux kernel + GNU compilers/tools/utilities + other free software
- Linux uses the Open Source model for development - code is placed on the Internet, users download and test it, programmers improve it and place it back on the web
- there is competition among programmers to fix bugs and improve Linux

UNIX Structure

- hardware, surrounded by
- UNIX kernel (basic OS), surrounded by
- shell (user interface, command interpreter, and some built-in commands), surrounded by
- utilities (or commands)
- most common shells: Bourne shell (sh), C shell (csh), Korn shell (ksh), Bourne again shell (bash), TC shell (tcsh), Z shell (zsh)
- we'll mostly be concerned with the Bash shell, which is the most popular Linux shell
- the Korn shell is the most popular Unix shell, and is very similar to the Bash shell

Common Commands

- pwd will show your current directory
- cd is used to change current directory, eg. cd directory-name
- ls - lists information about files and directories
- ls -a - all files (including hidden)
- ls -l - long form, gives more information about files`
- ls -d - gives information about the directory itself, not contained files
- these options can be mixed and matched, eg. ls -ld
- touch - creates a new file, or updates stats (eg. timestamp) on an existing file
- mkdir - to create directories
- mkdir -p - to create a directory & any parent directories not already existing
- rmdir - to delete empty directories, eg. rmdir directory-list
- mv existing-file new-file - to rename or move files & directories
- cp source-file destination-file - to copy files (careful, overwrites destination with no warning)
- cp -r - to copy directories including files and subdirectories
- rm - to delete files, eg. rm file-list
- rm -r - to delete directories including files and subdirectories
- cat filename - display contents of file
- more or less - displays file contents one page at a time
- <space bar> - will go to next page
- b - will go to previous page
- /string - will search for string within document being viewed
- q - will quit
- file filename - gives info about the contents of the file
- man command - online manual (or help) for command, uses more to display information
- man -k keyword - eg. man -k calendar
- searches through man sections for keyword, displays one-line summary of each related command
- diff file1 file2 - displays differences between 2 files
- echo text or $variable (eg. $HOME)
- echo -n "Hello" - doesn't skip to a new line
- echo -e "Hello\nthere" - enables escapes, eg. \n (newline) \t (tab)
- eg. echo -ne "This is a menu\n\t1. Item 1\n\t2. Item2\n\t3. Item3\nEnter choice: "
- printf - similar to echo but can use C-style formatting
- allows escapes: eg. \n (newline) \t (tab)
- allows string length specifications
- eg. printf "%-20s%20s%10.2f\n\n" "$name" "$phone" "$price"
- date - gives date and time
- which utility - lists pathname that would be used to access this utility
- who - displays information about users logged on to system
- whoami - displays your userid

Permissions

- permissions can only be changed by file owner or superuser (system administrator)
- chmod is used to alter access permissions to an existing file or directory
- the 9 permission bits displayed in an ls -al listing are read/write/execute for user(owner)/group/other
- chmod xxx filename
- xxx is 3 octal digits representing the binary string rwxrwxrwx where the first three characters are read/write/execute permission for the user, the next three for the user's group, and the last three for all others
- eg. chmod 640 file1 - would give the user read and write permission, everyone in his group would have read permission, and all others would have no permission
- this is called the "octal" or "absolute" method of changing permissions
- chmod u+r filename
- u represents user, could also be g for group, o for other, a for all
- + represents addition of permission, could also be - for removal, = for set
- r represents read permission, could also be w for write, x for execute
- eg. chmod u+x file1 - would give the user execute permission in addition to whatever he had before
- eg. chmod g-w file1 - would take away write permission from the user's group if they had it before
- eg. chmod o=r file1 - would set all others' permission to read only regardless of what they had before
- this is called the "symbolic" or "relative" method of changing permissions

Directory Permissions

- r permission for a directory allows viewing of file names in the directory, but no access to the files themselves (regardless of the files' permission settings)
- x gives passthrough permission for a directory, which allows access to any files in the directory which have appropriate permissions set, but doesn't allow viewing of file names in the directory
- x also gives permission to cd to the directory, changing the pwd
- r and x permissions allow viewing of file names, and access to any files which have appropriate permissions set
- w and x permissions allow adding or removing of files, but don't allow viewing of file names
- r and w and x permissions allow viewing of file names, access to any files which have appropriate permissions set, and adding and removing of files

umask

- umask defines default permissions for newly created files, doesn't change permissions on existing files
- default permissions will be 777 minus umask for directories, remove any remaining executes for files
- eg. umask - by itself, shows current umask setting
- eg. umask 077 - new directories will be 700, new files will be 600
- eg. umask 023 - new directories will be 754, new files will be 644