Unix System Overview

Outline

- Why Unix
- File Systems
- Security and File Permission
- Text Editor
- Command Summary

Why UNIX?

- **Computer operating system** An operating system is the program that controls all the other parts of a computer system, both the hardware and the software. It allocates the computer's resources and schedules tasks. Every computer requires an operating system.
- **Portable** The built-in C compiler allows software to be developed for multiple platforms. All that is needed is the standard C compiler.
- **Multiuser** Allows multiple users to concurrently share hardware and resources and the same time, preventing any one user from locking out others.
- **Multitasking** Allows a user to run more than one job at a time
- **Networking** Allows users at one location to log into the system at other sites as if they were using a local system.
- **Organized File System** Can handle large disk capacity
- **Device Independence** UNIX treats input/output device like ordinary files.
- **Utilities** Hundreds of available productivity tools.
- **Services** Support utilities for system administrator to monitor system and help users

UNIX Structure

The Kernel - The heart of the Unix system

- Loaded at startup
- It contains process control and resource management
- (There is no need to know about the kernel in order to use a UNIX system. It is provided here for information only.)

The Shell - Part of UNIX that is most visible to the user.

- It receives and interprets commands entered by the user.
- The most commonly available shells are Bourne shell (sh), C shell (csh), TC shell (tcsh) and Bourne Again Shell (bash).
- At the UNIX/Linux prompt, type: **echo $SHELL** to check the environment shell.

Utilities - Standard UNIX programs that provide support services for users such as text editors, search programs, and printing.

Application - Programs that provide extended capabilities to the system.

Accessing UNIX

- The UISACAD5 server allows automatic creation of accounts for users who are enrolled at UIS. Your account and password are the same as the ones used to log into UIS email and Blackboard.

- You must have an SSH client to access the server. PuTTY is an open source SSH client. Run PuTTY and enter uisacad5.uis.edu as host, connection type SSH then click open. Type your UIS NetID and password when prompted. UNIX is case sensitive.

- If successfully logged in, a UNIX prompt will be displayed.

- For further information, read the Linux Login Primer available in Course Resources.

- Commands - A UNIX command is an action request given to the UNIX shell for execution.

Outline

- Why Unix
- File Systems
- Security and File Permission
- Text Editor
- Command Summary

Filenames

- Filenames

  - Can be up to 255 characters

  - Can be any sequence of ASCII characters. Except the greater than (>) or less than (<) characters since they are used for redirection.

  - A period in UNIX does not have a special meaning as it does in other OS. In many other OS, the period is used to separate a filename from its extension. In UNIX, it can be used anywhere in the filename.

- Filename rules:

  - Must start with alphabetic character

  - Use dividers to separate parts of the name. Good dividers are underscore, period, and hyphen.

  - Use an extension at the end of the filename to help classify its application, even thought UNIX does not require extensions. For example, .bin for binary, .c for C programs

  - Never start with a filename with a period unless you intend to hide it. Filenames beginning with a period are hidden files in UNIX.

Wildcards

- A wildcard is a token than specifies that one or more different characters can be used to satisfy a specific request.
- Three wildcards:
  - Single character (?) wildcard
  - Set ([...]) wildcard
  - Multiple character (*)

File	Matches	Does Not Match
c?	c1 c2	ac
c?t	cet cit c1t	cad
f[aeiou]d	fad fed fid fod fud	ftb fab
f[a-d]t	fat fbt fct fdt	fab fet
*	Every file	
f*	every file whose name begins with f, ex. f5c2	afile cat
.	every file whose name has a period, ex. File.dat	file cur

File Types

- Regular File contains user data that is available for future processing.
- Directory File contains the names and locations of all files stored on a physical device.
- Character Special File represents a physical device such as a terminal, that reads or writes one character at a time.
- Block Special File represents a physical device, such as a disk, that reads or writes one block at a time.
- Symbolic Link File is a logical file that defines the location of another file somewhere else in the system.
- FIFO File A first-in, first-out also known as a named pipe, is a file that is used for inter-procceses communication.
- Socket is a special file that is used for network communication.

Regular Files/Directories

Regular Files:

- Text Files UNIX uses the ASCII character set
- Binary File is a collection of data stored in the internal format of the computer

Directories:

- Root (/) is the highest level in the hierarchy. It is the root of the whole file structure. It belongs to the system administrator.
- Home (/home/*user*/) – Contains the files we created as a user on the system. This is our home directory.
- Working (.) – The working or current directory is the one that we are in at any point in a session.
- Parent (..) is immediately above the working directory.
- Absolute Pathname specifies the full path from the root to the desired directory or file.
- Relative Pathname is the path from the working directory to the file. For example, if we are in staff directory, we can refer file8 in the tran directory as tran/file8. If using the absolute path it would be /home/staff/tran/file8.

File System Structure

In UNIX, a File System has four structure sections known as:

Boot Block – When an OS is started, a small program know as a boot program is used to load the kernel into memory. The boot program, when present, is found at the beginning of a disk in the boot block.

Super Block - The next block on the disk, the super block, contains information about the file system. Stored here are such items as total size of the disk, how many blocks are empty, and the location of bad blocks on the disk.

Inode Block – Following the super block are inode (information node) blocks, which contains information about each file in the data block. An inode is like an index file that contains metadata about data files. They contain info about the files like: owner, file mode, file size, file address , permissions, time of last access, and time of last modification.

Data Block – This is where data is stored, it also contains the special files that are related to the user data: regular file, directory files, symbolic link file, and FIFO files, character special, block special and socket system files.

Links

A link is a logical relationship between an inode and a file that relates the name of a file to its physical location.

Two types of link in UNIX:

Hard Links - In a hard link structure, the inode in the directory links the filename directly to the physical file.

Symbolic Links – A symbolic (or soft) link is a structure in which the inode is related to the physical file through a special file know as symbolic link

The default link type is hard. To create a symbolic link, the symbolic option is used (-s).

Format:

# ln –s *source_file destination_file*

Outline

- Why Unix
- File Systems
- Security and File Permission
- Text Editor
- Command Summary

Security (1)

There are three levels of security in UNIX: system, directory and file. The system security is controlled by the system administrator, a super user. The directory and file are controlled by the users who own them.

Users – Anyone who logs on the system.

Superuser – Users who have more capabilities, also know as administrators .

Group – Users can be organized into groups. The system administrator can create a group and assign users to be members of such group.

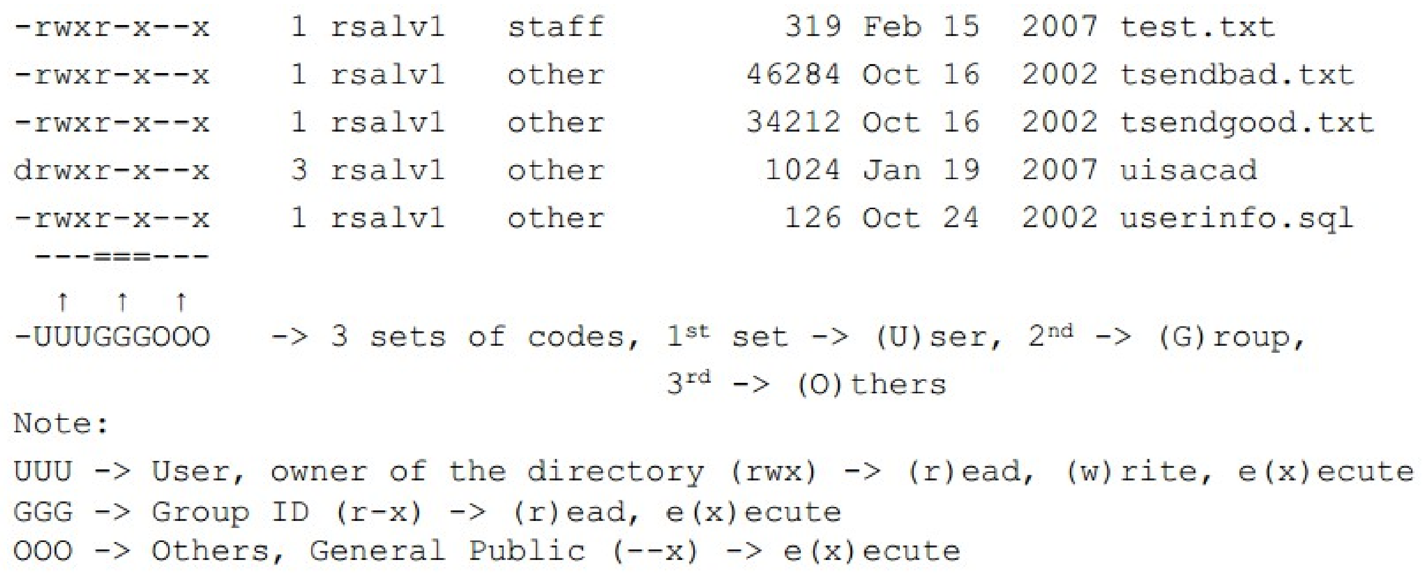
UNIX has system security controls that define who is allowed to access the system.

Generally, when an admin creates an account for you, the account information is written in /etc/passwd.

The passwd contains the login name, encrypted password, user id, group id, user information, home directory and login shell.

Security (2)

- Permissions exist both in the directory and the file security levels to determine who can access and manipulate a directory or file. The permission codes are divided into three sets of codes. The first set contains the permission of the owner of the directory. The second set contains the group permissions for members in a group as identified by a group id. The third set contains the permissions for everyone else (other users on the system).
- Example:



```
-rwxr-x--x    1 rsalvl   staff          319 Feb 15  2007 test.txt
-rwxr-x--x    1 rsalvl   other        46284 Oct 16  2002 tsendbad.txt
-rwxr-x--x    1 rsalvl   other        34212 Oct 16  2002 tsendgood.txt
drwxr-x--x    3 rsalvl   other         1024 Jan 19  2007 uisacad
-rwxr-x--x    1 rsalvl   other          126 Oct 24  2002 userinfo.sql
 ---===---
   ↑  ↑  ↑
-UUUGGGOOO   -> 3 sets of codes, 1st set -> (U)ser, 2nd -> (G)roup,
                                 3rd -> (O)thers
Note:
UUU -> User, owner of the directory (rwx) -> (r)ead, (w)rite, e(x)ecute
GGG -> Group ID (r-x) -> (r)ead, e(x)ecute
OOO -> Others, General Public (--x) -> e(x)ecute
```

Permissions (1)

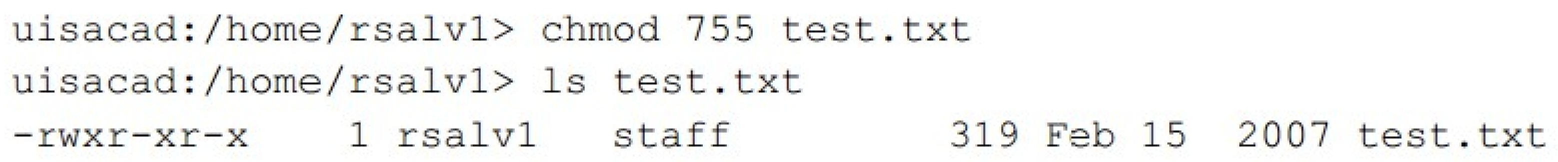
- Read Permission (r). The users can read the directory which contain the names of the files and subdirectories and all of their attributes.
- Write Permission (w). The users can add or delete entries.
- Execute Permission (x). Allows users to execute program files.

Command	**Interpretation**
chmod u=rwx *filename*	Sets read (r), write (w), execute (x) for user.
chmod g=rx *filename*	Sets read (r), and execute (x) for group; write (w) denied
chmod g+x *filename*	Adds execute (x) permission for group; read and write unchanged.
chmod a+r *filename*	Adds read (r) for all users,; write (w), execute (x) unchanged
chmod o-w file	Removes others’ write (w) permission , read and execute unchanged.

Permissions (2)

- You may also use Octal Notation, r=4, w=2, x=1.
  - 4+2+1 = 7 (rwx)
  - 4+1 = 5   (rx)

Example:



```
uisacad:/home/rsalv1> chmod 755 test.txt
uisacad:/home/rsalv1> ls test.txt
-rwxr-xr-x    1 rsalv1   staff          319 Feb 15  2007 test.txt
```

Outline

- Why Unix
- File Systems
- Security and File Permission
- Text Editor
- Command Summary

vi Text Editor

- The vi text editor is included in most UNIX and Linux distributions.
- You may use any available text editor; however, vi is recommended
- To learn how to use vi please study the following links:
  - http://www.washington.edu/computing/unix/vi.html
  - http://heather.cs.ucdavis.edu/~matloff/UnixAndC/Editors/ViIntro.html

- The following page provides a chart of common vi commands

vi Commands

- i –inserts text before the current character
- I –inserts text at the beginning of the current line
- a –appends text after the current character
- A –appends text at the end of the current line.
- o –opens an empty text line for new text after the current line
- O –opens an empty text line for new text before the current line.
- x –deletes the current character
- dd –deletes the current line
- u -to undo the most recent edit
- U –to undo all the edits on a single line
- [esc] :w –saves any edits made
- [esc] :wq –saves and exit
- [esc] :q! -exit without saving

Outline

- Why Unix
- File Systems
- Security and File Permission
- Text Editor
- Command Summary

Command Summary (1)

Command	**Description**
cat *file(s)*	Display contents of *file(s)* or echo standard input if no *file(s)*
cd *dir*	Change working directory to *dir*
cp *file1 file2*	Copy *file1* to *file2*
date	Display system date and time
echo *args*	Display *args*
ln *file1 file2*	Link *file1* to *file2*
ln *file(s) directory*	Link *files* into *directory*
ls *file(s)*	List *file(s)*
ls *dir(s)*	List files in *dir(s)* or in current directory if no *dir(s)*
mkdir *dir(s)*	Create directory(s) named dir(s)
mv *file1 file2*	Move *file1* to *file2* (or rename *file1* to *file2* if working in same dir)

Command Summary (2)

Command	Description
mv *file(s) dir*	Move *file(s)* into directory *dir*
ps	List information about active processes
pwd	Display current working directory
rm *file(s)*	Remove (delete) *file(s)*
rmdir *dir(s)*	Remove empty directory *dir(s)*
sort *file(s)*	Sort lines of *file(s)* or standard input if no *file(s)* specified
wc *file(s)*	Count the number of lines, words, and characters in *file(s)*or standard input of no *file(s)* specified
who	Display who is currently logged in
man *command*	Display manual page of *command* Example: man ps