Unix Basics

&

Shell Programming

Operating System

- Is a System software
- Can be defined as organized collection of software consisting of procedures for operating a computer
- Provides an environment for execution of programs
- Acts as an interface between the user and the hardware of the computer system.
- Operating system interacts with the user in two ways
  - Operating system calls provides an interface to a running program and the operating system. System calls in UNIX is written in C.
  - Operating system commands enables the user to interact directly with the operating system.

History of UNIX

- Ken Thompson of AT&T ,BELL laboratories originally designed UNIX in late 1960s which evolved from timesharing operating system called multics.

- Originally written in assembly language but with the development of ‘c’ programming language in 1973,it was then written in ‘c’.

- Two versions of UNIX emerged are AT&T Unix and BSD Unix 1989 AT&T and Sun Micro system joined together and developed system V release 4 (SVR4)

- Two of the main standards mainly in use are POSIX(Portable Operating System Interface) and X/open standard

- 1988 MIT formed Xconsortium which developed vendor Neutral Xwindow System

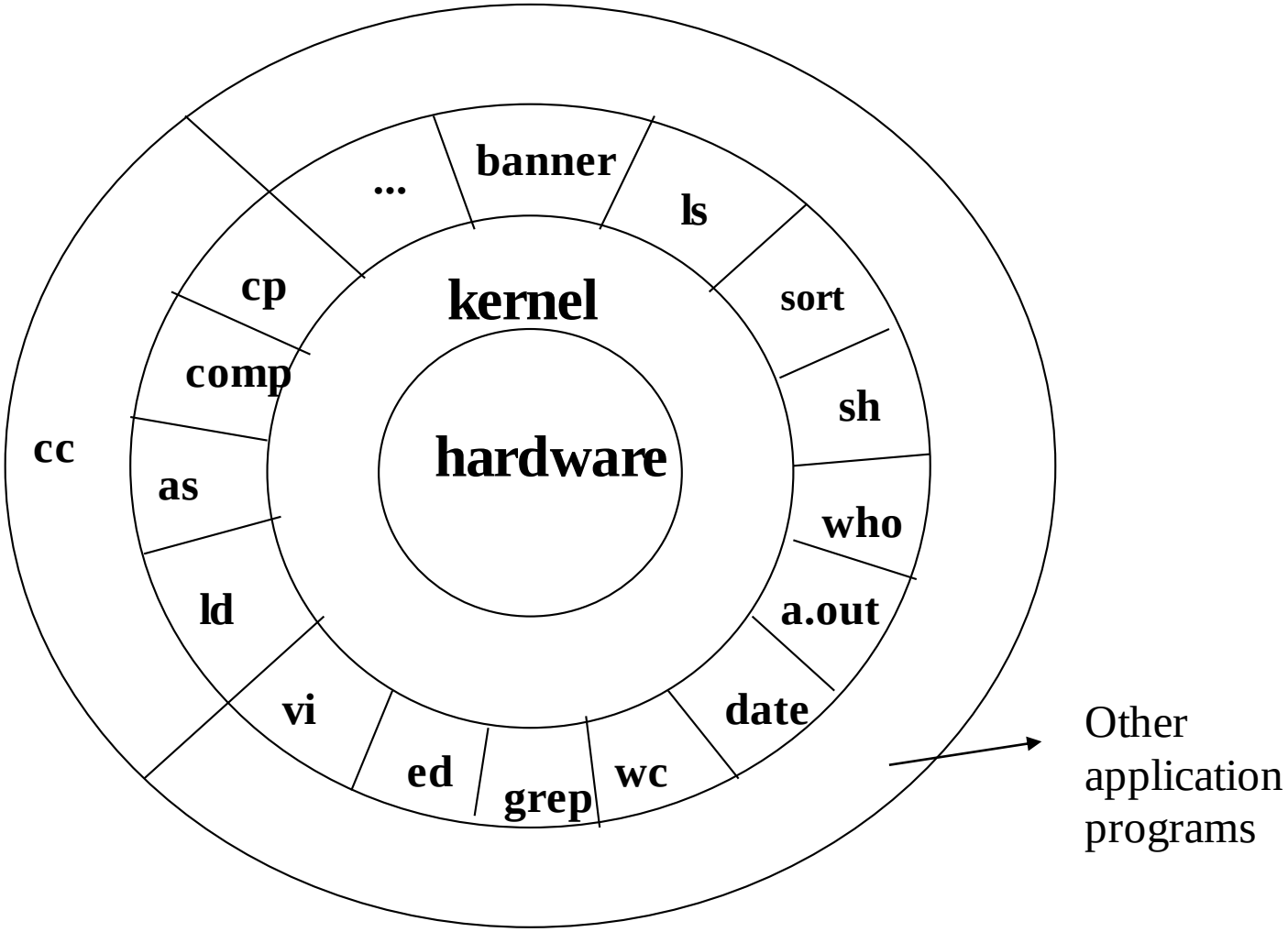
Features of UNIX

- Multi-user /multitasking/timesharing concept
- Portability
- Modularity
- File structure
- Security
- Strong networking support
- Advance graphics
- Multi-user, Multiprocessing Operating System
- Consists of several dumb terminals attached to it.
- Written in high-level language C
- Uses hierarchical file system - allows easy maintenance and efficient implementation

UNIX System Architecture

- Unix System follows a layered Approach it has four layers
- The innermost layer is the hardware layer
- In the second layer, the kernel is placed.
- The utilities and the other application programs form the third layer
- Fourth layer is the one with which the user actually interacts.

Layered Architecture



Kernel

- Kernel is that part of the OS which directly makes interface with the hardware system.
  - Provides mechanism for creating and deleting processes
  - Provides processor scheduling, memory and IO management
  - Does interprocess communication.

Kernel Architecture

The Linux kernel is monolithic. It is a large, complex.

Composed of several logically different components.

Most commercial Unix variants are monolithic.

Traditional Unix kernels are compiled and linked statically.

Most modern kernels can dynamically load and unload some portions of the kernel code (typically, device drivers), which are usually called *modules*.

contd..

- ***Kernel threading:*** Some modern Unix kernels, like Solaris 2.x and SVR4.2/MP, are organized as a set of *kernel threads*.
- A kernel thread is an execution context that can be independently scheduled; it may be associated with a user program, or it may run only some kernel functions.
- Context switches between kernel threads are usually much less expensive than context switches between ordinary processes.
- Linux uses kernel threads in a very limited way to execute a few kernel functions periodically; since Linux kernel threads cannot execute user programs, they do not represent the basic execution context abstraction.
- ***Multithreaded application support:*** Most modern operating systems have some kind of support for multithreaded applications, that is, user programs that are well designed in terms of many relatively independent execution flows sharing a large portion of the application data structures.

contd..

- A multithreaded user application could be composed of many *lightweight processes* (LWP), or processes that can operate on a common address space, common physical memory pages, common opened files, and so on.

- Linux defines its own version of lightweight processes, which is different from the types used on other systems such as SVR4 and Solaris.

- While all the commercial Unix variants of LWP are based on kernel threads,

- Linux regards lightweight processes as the basic execution context and handles them via the nonstandard clone( ) system call.

contd..

- Linux is a nonpreemptive kernel.
- This means that Linux cannot arbitrarily interleave execution flows while they are in privileged mode.
- Several sections of kernel code assume they can run and modify data structures without fear of being interrupted and having another thread alter those data structures.
- Usually, fully preemptive kernels are associated with special real-time operating systems.
- Currently, among conventional, general-purpose Unix systems, only Solaris 2.x and Mach 3.0 are fully preemptive kernels.
- SVR4.2/MP introduces some *fixed preemption points* as a method to get limited preemption capability.

The Shell

- A Utility program that comes with the unix system.
- Features of Shell are:
  - Interactive Processing
  - Background Processing
  - I/O Redirection
  - Pipes
  - Shell Scripts
  - Shell Variables
  - Programming Constructs

CPU Scheduling

- Unix uses “Round-Robin” scheduling to support its multi-user and time-sharing feature.
- Round-Robin fashion of scheduling is considered to be oldest,simplest and widely used algorithm.
- Here, every process is given a time slice(10-100 millisec.)

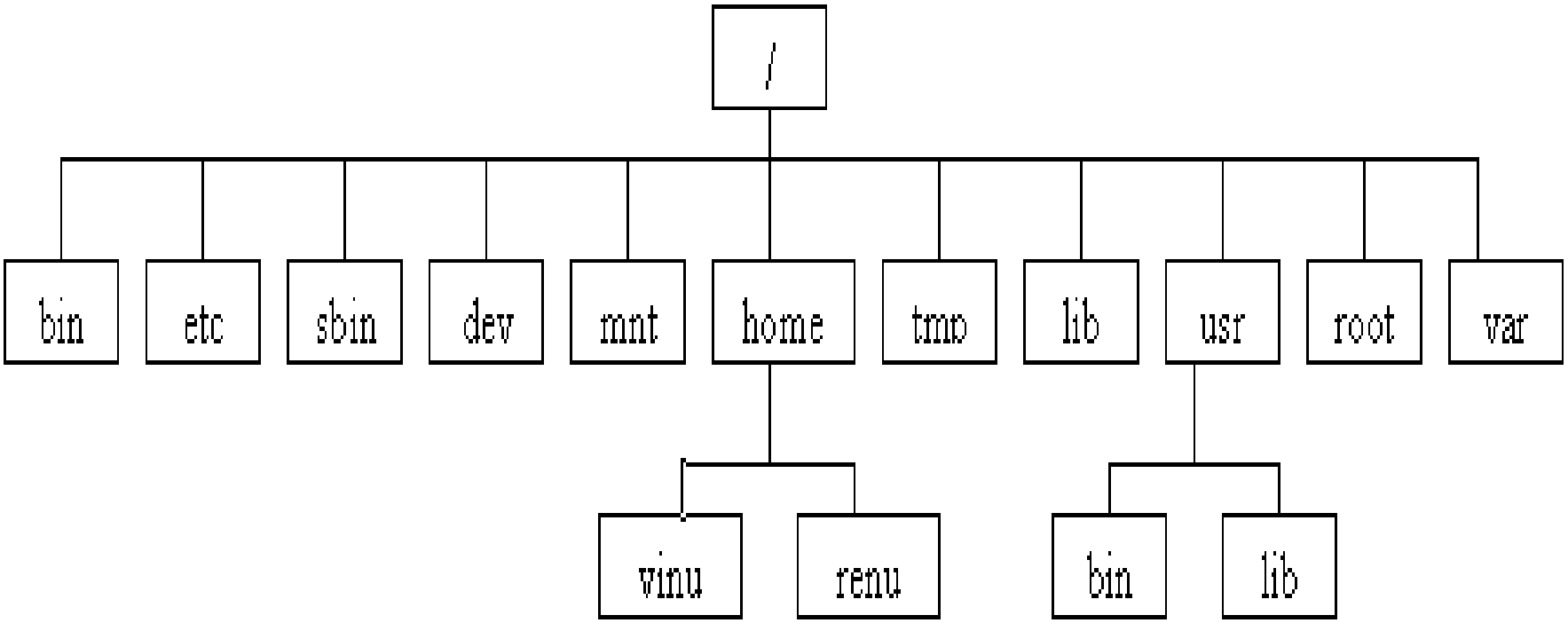
Memory Management

- Swapping
- Demand Paging

File system

- The role of the file system
- Various rules for naming files
- Absolute and relative pathnames
- Working directory and home directory
- Unix uses a hierarchical file system with "/" as its root.
- Every non-leaf node of the tree is called as a directory file.
- Every leaf node can either be a file or an empty directory

File Organization in Unix / Linux



contd..

- A brief description of the purpose of the most important directories follows:
- /home This directory contains the home directories of all the user accounts in the system. The home directory of a user will have the same name as his login name.
- /bin Most of the executable files, like command files.
- /sbin Important executable files needed by the system.
- /lib Essential library files are kept here. These libraries are required by the system.
- /etc All system configuration script files are kept in this directory, and in sub-directories under this directory.
- /dev This directory contains special device files i.e., special files that represent devices.
- /mnt This directory generally provides the mount point for external file systems, like the floppy disk or the CD-ROM.

contd..

- /var Files with variable data are kept here. These files are maintained by Linux, and their contents keep changing to reflect the current state of the system.
- /usr/bin This directory generally contains the executable files of different utilities that are not part of the core system.
- /usr/lib Library files needed by the external utilities mentioned above are kept here.

Common UNIX flavours

- BSD: Berkeley, BSD
- Solaris: Sun Microsystems, Sys 5/BSD
- Ultrix: Digital Equipment Corporation, BSD
- OSF 1: Digital Equipment Corporation, BSD/sys 5
- HPUX: Hewlett-Packard, Sys 5
- AIX: IBM, Sys 5 / BSD
- IRIX: Silicon Graphics, Sys 5
- GNU/Linux: GNU, BSD/Posix

Working with UNIX

- Log in
- Log out

UNIX COMMANDS

Internal and external commands

- Some commands in Unix are internal, built into the shell.
  - Example: cd, pwd, read, history, alias
  - The shell doesn't start a separate process to run internal commands.
- The command which are not built in shell is called external commands.
  - Example : /bin/cat
  - External commands require the shell to run a new sub process.

General purpose commands

- man – Unix manual
- info help
- pwd Identifies the current working directory.
- date Displays the current date and time
- who Displays the names of all the users who have currently logged in.

General purpose commands

- cal Prints calendar of specified month and year
- who am i Displays the name of the current user.
- printf An alternative to echo
- bc The calculator
- passwd To change password
- echo
  - Print the string
- tty
  - To know your terminal
- stty
  - To display and set terminal character
- uname
  - To know machine name / type

contd..

- **Examples**
- *man* command is used to provide a one line synopsis of any commands that contain the keyword that you want to search on with the "-k" option.
- For e.g. to search on the keyword password, type:
- $ man -k password
- passwd (5) – password file
- passwd (1) – change password information
- The number in parentheses indicates the section of the man pages where these references were found.

Displaying calendar

- ***cal*** displays calendar.
- **Syntax**
  - Cal [-smjy13] [[month] year]
- **Examples**
  - **$cal 2007 [Displays calendar of the year 2007]**
  - $cal [- displays current months calendar]
- **Options**
  - -1 display single month output
  - -3 display prev/current/next month output
  - s display Sunday as first day of the week
  - m display Monday as first day of the week
  - y display calendar for the current year

Printing date and time

- ***date*** displays the current data and time. A super user 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
  - %**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

Contd….

- **Examples:**

```
$date
      Mon Jun 10 09:01:05  EDT 2006

$date -u
      Mon Jun 10 13:01:33 GMT 2006

$date +%a%t%D
    Mon 06/10/06

$date '+%y:%j'
      96:162
```

echo a statement

- The ***echo*** command is used to repeat, or echo, the argument user gives it back to the standard output device.
- **Syntax**
  - ***echo*** [string]
- **Common Options**
  - **-n** don't print <new-line> (BSD, shell built-in)
  - **c** don't print <new-line> (SVR4)
  - **0n** where **n** is the 8-bit ASCII character code (SVR4)
  - **t** tab (SVR4)
  - **f** form-feed (SVR4)
  - **n** new-line (SVR4)
  - **v** vertical tab (SVR4)

$ echo Hello Class or echo "Hello Class"

- To prevent the line feed:
- $ echo -n Hello Class or echo "Hello Class \c"

An alternative to echo printf

- printf – format and print data
- Syntax:
  - printf FORMAT [ARGUMENT]...

Options:

print arguments according to Format

\"	double quote
\\	backslash
\a	alert
\b	backspace
\n	new line
\r	carriage return
\t	horizontal tab
\v	vertical tab
%%	a single %

The calculator : bc

- bc – It is an intaractive tool for performing calculator like computations.
  - Example:
    - $bc

      10 + 20

      30

      ^d to exit bc

      While computing bc truncates the decimal portion in the result. To enable floating point computation, the number of digits of precision is set.

      $bc

      scale=5

      10 / 3

      3.33333

      ^d

Changing the passwords

- Passwd - update a user's authentications
- syntax
  - Passwd [-k][-l][-u[-f]][-d][-S] [username]

  Options:

  -k is used to indicate that the update should only be for expired authentication tokens

  -l is used to lock the specified account, it is available to root only.

  -u This is the reverse of the –l option.

  -d This is a quick way to disable a passwd

  -x This will set the maximum password life time, in days.

  -w This will set the number of days in advance user will begin receiving warnigs that his passwd will expire.

who - list current users

- ***who*** reports who is logged in at the present.
- **Syntax**
  - ***who*** [am i]
- **Examples**

  $who

  rashmi ttyp1 Apr 21 20:15 (apple.acs.ohio-s)
  ravi ttyp4 Apr 21 22:27 (slip1-61.acs.ohi)
  sudhir ttyp5 Apr 21 23:07 (picard.acs.ohio-)
  leela ttyp6 Apr 21 23:00 (ts31-4.homenet.o)

  $who –q

  Rashmi ravi sudhir leela
  #users = 3

- Who am I

  $who am i

- Guest ttypc Apr 21 23:38 (mars)

Print system information

- ***uname*** has additional options to print information about system hardware type and software version.
- Option:
  - a option prints all the information.
  - s kernel name
  - r kernel release
  - v kernel version
  - m machine
  - p processor
  - i hardware platform
- E.g:
  - $uname –a

  Linux mars 2.4.20-8 #1 Thu Mar 13 17:54:28 EST 2003 i686 i686 i386 GNU/Linux

Printing the name of the terminal

- Tty prints file name of the terminal connected to standard input

- Syntax:

  - tty [option]

  Example:

  $tty

  /dev/tty4

terminal control

- ***stty*** reports or sets terminal control options.

- the ***stty*** command provides an invaluable tool for configuring many aspects of I/O control for a given device like:

  - erase and line-kill characters
  - data transmission speed
  - parity checking on data transmission
  - hardware flow control
  - newline (NL) versus carriage return plus linefeed (CR-LF)

- **Syntax**
  - ***stty*** [options]

- **Options**
  - (none) report the terminal settings
  - **all** (or **-a**) report on all options
  - **echoe** echo ERASE as BS-space-BS
  - **kill** set the LINE-KILL character
  - **erase** set the ERASE character
  - **intr** set the INTERRUPT character

- **Examples**
  - % stty -a
  - speed 38400 baud, 24 rows, 80 columns

… …. …. ….

  - erase kill werase rprnt flush lnext susp intr quit stop eof
  - ^H ^U ^W ^R ^O ^V ^Z/^Y ^C ^\ ^S/^Q ^D

- You can change settings using ***stty***, e.g., to change the erase character from **^?** (the delete key) to **^H**:
  - $ stty erase ^H
- To turn off keyboard input
  - $ stty –echo
- To restore keyboard input
  - $stty echo
- To erase character while doing backspace
  - $ stty echoe
- To reverse the above setting
  - $stty –echoe
- To set terminal characteristics to work with most of the terminals
  - $stty sane

whereis - report program locations

- *whereis* reports the filenames of source, binary, and manual page files associated with command(s).

- **Syntax**
  - *whereis* [options] command(s)

- **Common Options**
  - -b report binary files only
  - -m report manual sections only
  - -s report source files only

- **Examples**

- $ whereis mail
  - Mail: /usr/ucb/Mail /usr/lib/Mail.help /usr/lib/Mail.rc / usr/man/man1/Mail.1

- $ whereis -b mail
  - mail: /usr/ucb/Mail /usr/lib/Mail.help /usr/lib/Mail.rc

- $ whereis -m mail
  - Mail: /usr/man/man1/Mail.1

which - report the command found

- *which* will report the name of the file that is be executed when the command is invoked.

- This will be the full path name or the alias that's found first in your path.

- **Syntax**
  - *which* command(s)

- **example**
  - $ which Mail
    - /usr/ucb/Mail

cat

- Cat (concatenate) reads data from the file and gives their content as output. It helps us to create, view, concatenate files.
- 1) To view a single file
- $cat filename
- It will show content of given filename
- 2) To view multiple files
- $cat file1 file2
- This will show the content of file1 and file2.
- 3) To view contents of a file preceding with line numbers.
- $cat -n filename
- It will show content with line number
- example:-cat-n samples.txt
- 4) Create a file
- $ cat >newfile Will create and a file named newfile

contd..

- 5) Copy the contents of one file to another file.
- $cat [filename-whose-contents-is-to-be-copied] > [destination-filename]
- The content will be copied in destination file
- 6) Cat command can suppress repeated empty lines in output
- $cat -s samples.txt
- Will suppress repeated empty lines in output
- 7) Cat command can append the contents of one file to the end of another file.
- $cat file1 >> file2
- Will append the contents of one file to the end of another file
- 8) Cat command can display content in reverse order using tac command.
- $tac filename

contd..

- 9) Cat command can highlight the end of line.
- $cat -E "filename"
- Will highlight the end of line
- 10) If you want to use the -v, -E and -T option together, then instead of writing -vET in the command, you can just use the -A command line option.
- $cat -A "filename"
- 11) Cat command to open dashed files.
- $cat -- "-dashfile"
- Will display the content of -dashfile
- 12) Cat command if the file has a lot of content and can't fit in the terminal.
- $cat "filename" | more

contd..

12) Cat command to merge the contents of multiple files.

$cat "filename1" "filename2" "filename3" > "merged_filename"

Will merge the contents of file in respective order and will insert that content in "merged_filename".

13) Cat command to display the content of all text files in the folder.

$cat *.txt

Will show the content of all text files present in the folder.

awk

grep

- The grep filter searches a file for a particular pattern of characters, and displays all lines that contain that pattern. The pattern that is searched in the file is referred to as the regular expression (grep stands for globally search for regular expression and print out).

Syntax:

grep [options] pattern [files]

1. Case insensitive search : The -i option enables to search for a string case insensitively in the given file. It matches the words like "UNIX", "Unix", "unix".

$grep -i "UNix" samplefile.txt

2. Displaying the count of number of matches : We can find the number of lines that matches the given string/pattern

$grep -c "unix" samplefile.txt

contd..

3. Display the file names that matches the pattern : We can just display the files that contains the given string/pattern.
$grep -l "unix" * or $grep -l "unix" f1.txt f2.txt f3.xt f4.txt

4. Checking for the whole words in a file : By default, grep matches the given string/pattern even if it found as a substring in a file. The -w option to grep makes it match only the whole words.
$ grep -w "unix" samplefile.txt

5. Displaying only the matched pattern : By default, grep displays the entire line which has the matched string. We can make the grep to display only the matched string by using the -o option.
$ grep -o "unix" samplefile.txt

contd..

6. Show line number while displaying the output using grep -n : To show the line number of file with the line matched.

$ grep -n "unix" samplefile.txt

7. Inverting the pattern match : You can display the lines that are not matched with the specified search sting pattern using the -v option.

$ grep -v "unix" samplefile.txt

8. Matching the lines that start with a string : The ^ regular expression pattern specifies the start of a line. This can be used in grep to match the lines which start with the given string or pattern.

$ grep "^unix" samplefile.txt

contd..

9. Matching the lines that end with a string : The $ regular expression pattern specifies the end of a line. This can be used in grep to match the lines which end with the given string or pattern.
$ grep "os$" samplefile.txt

10.Specifies expression with -e option. Can use multiple times :
$grep –e "Agarwal" –e "Aggarwal" –e "Agrawal" samplefile.txt

11. -f file option Takes patterns from file, one per line.
$cat pattern.txt Agarwal Aggarwal Agrawal $grep –f pattern.txt samplefile.txt

head

- The head command, as the name implies, print the top N number of data of the given input. By default, it prints the first 10 lines of the specified files. If more than one file name is provided then data from each file is preceded by its file name.

Syntax:

head [OPTION]... [FILE]...

1. -n num: Prints the first 'num' lines instead of first 10 lines. **num** is mandatory to be specified in command otherwise it displays an error.

$ head -n 5 state.txt Andhra Pradesh Arunachal Pradesh Assam Bihar Chhattisgarh

2. -c num: Prints the first 'num' bytes from the file specified. Newline count as a single character, so if head prints out a newline, it will count it as a byte. **num** is mandatory to be specified in command otherwise displays an error.

$ head -c 6 state.txt Andhra

contd..

3. -q: It is used if more than 1 file is given. Because of this command, data from each file is not precedes by its file name.

Without using -q option ==> state.txt capital.txt <== Hyderabad Itanagar Dispur Patna Raipur Panaji Gandhinagar Chandigarh Shimla Srinagar

With using -q option $ head -q state.txt capital.txt Andhra Pradesh Arunachal Pradesh Assam Bihar Chhattisgarh Goa Gujarat Haryana Himachal Pradesh Jammu and Kashmir Hyderabad Itanagar Dispur Patna Raipur Panaji Gandhinagar Chandigarh Shimla Srinagar

4. -v: By using this option, data from the specified file is always preceded by its file name.

$ head -v state.txt ==> state.txt <== Andhra Pradesh Arunachal Pradesh Assam Bihar Chhattisgarh Goa Gujarat Haryana Himachal Pradesh Jammu and Kashmir

Editors

Vi editor

Vi is a visual editor used to enter and edit text files.

- A screen-oriented text editor
- Included with most UNIX system distributions
- Command driven

Categories of commands include

- General administration
- Cursor movement
- Insert text
- Delete text
- Paste text
- Modify text

Editing commands

The vi editor is invoked by the following command

To edit a file

- vi [*filename*]

- To recover an editing session
  - vi -r [*filename*]

- Text insertion / replacement
  - i - inserts text to the left of the cursor
  - a - inserts text to the right of the cursor
  - I - inserts text at the beginning of the line
  - A - appends text at end of the line
  - o - opens line below
  - O - opens line above
  - R - replaces text from cursor to right
  - s - replaces a single character with any number of characters
  - S - replaces entire line

Editing commands

- Deletion
  - -x - to delete character at cursor position
  - -3x - to delete 3 characters at cursor position
  - dw - to delete word
  - -2dw - to delete 2 word
  - dd - to delete a line
  - 2dd - to delete 2 lines

Editing commands

- Yanking
  - Y - copy line into buffer
  - 3Y - copy 3 lines into buffer
  - p - copy buffer below cursor
  - P - copy buffer above cursor
- Save and quit
  - :w - to save
  - :w! - to name a file (:w! filename -> *save as*)
  - :x - save and quit
  - :q - cancel changes
  - :q! - cancel and quit

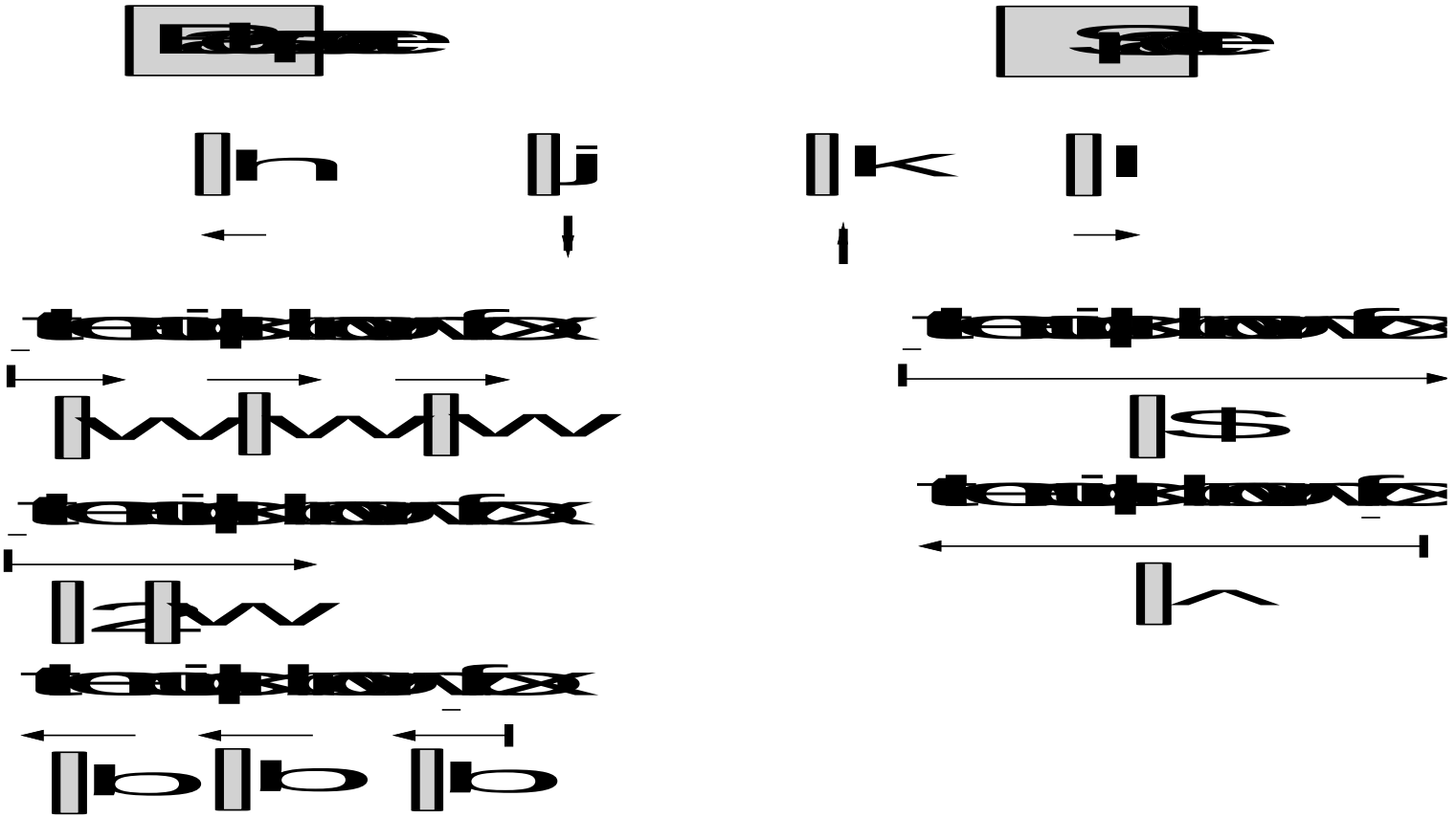
Search & replace commands

- **Searching Through Text**
- Backward for pattern ?*pattern*
- Forward for pattern /*pattern*
- Repeat previous search n
- Reverse direction of previous search N
- Show *all* lines containing pattern :beg,end g/pattern/p
  - **:1,$g/compiler/p** Will print all lines with the pattern **compiler**.
  - Substitute patt2 for all patt1 found. :beg,ends/patt1/patt2/g
  - **:%s/notfound/found/g** Will change all occurences of notfound to found.

Some Useful ex commands for use in vi

- Some useful set options for your ~/.exrc file:
- :set all Display all Set options
- :set autoindent Automagically indent following lines to the indentation of previous line.
- :set ignorecase Ignore case during pattern matching.
- :set list Show special characters in the file.
- :set number Display line numbers.
- :set shiftwidth=n Width for shifting operators << and >>
- :set showmode Display mode when in Insert, Append, or Replace mode.
- :set wrapmargin=n Set right margin 80-n for autowrapping lines (inserting newlines). 0 turns it off.

Navigation



Files & Directories

Objectives

- File Related Commands
- File Permissions
- Directory Related Commands

File related commands

FILE OPERATIONS	COMMANDS
Listing contents of the directory	ls
Copying a file	cp
Moving a file	mv
Removing a file	rm
Displaying a file	Cat
Changing directory	Cd
Present working directory	pwd

- Mkdir
- Rmdir
- Find
- More
- File
- Wc
- Cmp
- Comm
- Diff
- ln
- Touch

Listing the directory contents

ls

Syntax :ls [options] [file….]

options: -l list in long format

-a list all file including those beginning with a dot

-i list inode no of file in first column

-s reports disk blocks occupied by file

-R recursively list all sub directories

-F mark type of each file

-C display files in columns

Listing files and directories

$ ls –l

total 6

-rwxr-xr-x 1 user1 training 12373 Dec 15 14:45 a.out

drwxr-xr-x 2 user1 training 4096 Dec 22 14:00 awkpro

-rw-r--r-- 1 user1 training 12831 Dec 12 13:59 c

-rw------- 1 user1 training 61440 Dec 15 11:16 core

-rw-r--r-- 1 user1 training 255 Dec 20 14:29 cs

$ ls –a (All files including those beginning with (.) and (..))

$ ls –F (Marks directories with a/ , executables with a* and symbolic links with a@)

$ ls –r (Sort file names in reverse order)

Command - cp

Used to copy files across directories

Syntax

cp <source file> <new file name>

Example

cp file1 file2

Note:

cp -r /dev/tty myfile (copy directories and sub directories)

Command - cp

- -p preserve following file attributes
  - owner id
  - Group id
  - permissions
  - Last modification time

- -r

  Recursive copy; copy subdirectories under the directory if any

cat

cat is concatenate files and print on the standard output

cat command takes the input from the keyboard and send the output to the monitor

We can redirect the input and output using the redirection operators

- $ cat>file1
  - Type the content here
  - press <ctrl d>
- $ cat file1
  - Displays the content of the file
- $cat>>file1
  - Will append to the content of the file

Command - mv

Used to move a file or rename a file

Preserves the following details

- owner id
- group id
- permissions
- Last modification time

-f suppresses all prompting

-i prompts before overwriting destination file

Command - rm

Used to remove a file

- Syntax : rm file(s)

-f suppresses all prompting

-i prompts before overwriting destination file

-r will recursively remove the file from a directory (can be used to delete a directory along with the content)

Caution: Use “i” option along with “r” to get notified on deletion

Directory creation

Command Syntax

mkdir [OPTION] DIRECTORY

$mkdir <path>/<directory>

$mkdir –m <directory>

$mkdir –p <directory1>/<directory2>/<directory3>

Directory removal

rmdir command removes directory

Syntax

- rmdir <directory name>

Example

Removes project1 directory in the current directory

- rmdir project1

Remove multiple directories

rmdir pos1 pos2

Remove the directory recursively

rmdir –p dir1/dir2/dir2

Rule: *rmdir can be executed to a directory if it is empty and not the current directory*