UNIX

* An **Operating System (OS)** is the software that manages the sharing of the resources of a computer and provides programmers with an interface used to access those resources.
* **Functions of operating system:**

1. Process Management
2. Main-memory management
3. Secondary-storage management
4. I/O System management
5. File management
6. Protection system
7. Networking
8. Command-Interpreter System

* History of Unix → evolved at AT&T Bell Labs in 1983
* Features of Unix
  1. Simple user interface
  2. Multi- user and multi-processing system
  3. Time sharing operating system
  4. Written in c
  5. Consistent ‘Byte Stream’ file format
  6. Hierarchical file system
* Services
  1. Process management →involves creation, termination, suspension, communication between processes.
  2. File management → involves aspects related to files like creation and deletion, file security and so on.
* System architecture ( Unix system is functionally organized in three levels)
  1. Kernel → schedules tasks and manages storage
  2. Shell → connects and interprets user’s commands, calls programs from memory and executes them. (Bourne shell, Korn shell, C shell)
  3. Tools and applications → offers additional functionality to the operating system.

***Commands***

1. Man ($man <command>) → provides online help.
2. Cal ($cal <command>) → display calendar from the year 1 to 9999.
3. Date ($ date [fmt]) → see current date and time, can be displayed in different formats
   * m-month, h-Month name, d-Day of month, y-Year(last 2 digits), H-Hour, M-Minutes, S-Seconds,T-Time in hh:mm:ss, A-Day of week(Sun to Sat),R-Time in AM/PM.
4. Lp ($ lp [option] [no of copies]<file name>) → printing files
   * Lp(listing), lpq(prints jobs in queue) ,lprm(kill job by job number)
5. Nl ($nl [option] <file name>) →print file contents with line numbers.
   * Options → -w(width of number), -v(indicate first line number), -i(increment line number by).
6. Tty ($tty) → display device name of terminal
7. Who ($who) → list all users who are currently logged in.
8. Who am I → to see current user.

**File system Structure**

* Regular file, Directory file, Device file
* File access permissions (rwx rwx rwx)(user(u) group(g) others(o))(r-read, w-write, x-execute).

***File Related Commands***

1. Pwd ($pwd) → check current directory
2. Cd ($cd <dir name>) → changes directories to specified directory.
   * Cd .. ( one level up), cd / (root directory), cd (home directory).
3. Logname ($logname) → check login directory.
4. Ls ($ls [option]) → list directory contents.
5. Cat ($cat <filename1 […filename n]>) → displaying and creating files.
   * Redirection
     1. < : input redirection.
     2. > : output redirection.
     3. 2> : Error redirection.
     4. ≫ : Append redirection.
6. Cp($cp [option] <file 1> <file 2>) → copies a file or group of files.
7. Rm($rm [option] <file 1> [….file n]) → deletes files.
8. Mv($mv <file1> <file2>) → rename file or group of files as well as directories.
9. Wc($wc [option] <file>) →counts lines, words, and character depending on option.
10. Cmp($cmp <file1> <file2>) → compare if two files are matching or not.
11. Comm($comm [option] <file1> [file2]) → compares two sorted files and gives 3 columnar output.(1st column- unique lines of first file, 2nd column- unique lines of second lines, 3rd column- common lines)
12. Diff($diff <list1> <list2>) → display the file differences.
13. Tr ($tr [option] <file1>) → first character set is replaced by the equivalent member in the second character set.($tr “[a-z]” “[A-Z]” < file1.txt)
14. More ($more <options> <+line number> <+/pattern> <filesnames(s)>) →to view one page at a time and particularly useful for viewing large files.
15. Chmod($chmod <category> <operation> <permission> <filenames>) → change the file permissions by using symbolic notations or octal numbers(4-read, 2-write, 1-execute).
16. Mkdir($mkdir <dir> [dir/subdir] [dir/subdir1]….[dir/subdir1/../subdir n]) →create a directory or a number of subdirectories.
17. Rmdir($rmdir <dir> [dir/subdir] [dir/subdir1]….[dir/subdir1/../subdir n]) → removes one or more directories. Only empty directory can be deleted.

***Filters***

1. Head ($head [number] <filename1> [… filename n]) → display the first 10 lines of a file or more files.
2. Tail ($tail [number] <filename> […filename n]) → display the last few lines or characters of the file.
3. Cut ($cut [options] <filename>) → retrieves selected fields from a file.
4. Paste ($paste <file1><file2>) → used for horizontal merging of files.
5. Sort ($sort [options] <filename>) → sort file in ascending order.
6. Uniq ($uniq [option] <filename>) → fetches only one copy of redundant records and writes the same to standard output.
7. Tee ($tee <filename>) → copies standard input to standard output and also to the specified file.
8. Find ($ find <path list> <selection criteria> <action>) → locates files.
9. Grep ($grep <options> <pattern> <filenames>) → used to locate a pattern / expression in a file /set of files.
10. Fgrep ($fgrep [-e pattern\_list] [-f pattern\_file] [pattern] [search file]) → useful to search files for one or more patterns, which cannot be combined together.
11. Egrep ($egrep [ -e pattern\_list] [-f file] [strings] [file]) → it uses extended regular expression matching.

***Shell types:***

* 1. Bourne shell : -sh (named after its founder steve bourne)
  2. K shell : -ksh (from the univ. of California, berkeley)
  3. C shell : -csh (by david korn of bell lab)
  4. Restricted shell : -rsh

***Building block primitives***:

* Pipe (|) → allows stream of data to be passed between reader & writer process.

***Command substitution:*** shell allows the argument of a command to be obtained from the output of another command. This is done by using a pair of backquotes(``).

***Shell scripting***

Eval( eval `<command>`) → used to assign values to variable.

***Vi editor***

Three modes of Vi editor

* input mode → insert, append, replace or change text.
* command mode → w(save),q(quit),q!(quit without saving),wq (or) :x (save and exit)
* ex mode → : colon to execute command.

The editor ed was developed by Ken Thompson.

***Navigation commands:*** h(left), j(down),k(up),l(right),^ (beginning of first), $(end of line), b(backwards to beginning of word), e(end of word), w(beginning of word), 0 or | beginning of line(no repeat factor with 0).

***Search and repeat commands:***

* /pat → searches forward for pat.
* ?pat → searches backward for pattern pat.
* n → repeats search in the same direction along which the previous search was made (no repeat factor).
* N → repeats search in the opposite direction along which the previous search was made (no repeat factor).
* -fch → moves cursor forward to first occurrence of ch in current line.

***Other features:***

* Set ($:set <command>)→ used to customize the behavior of vi editor.
* Sed (sed options sed\_script filename) → non-interactive stream oriented editor for filtering and transforming text, simplify task performing on multiple files, write conversion programs.
  + Substitute (sed ‘s/old/new’ file) → changes all occurrences of the regular expression into a new values.

***Process commands:***

* Ps (ps [options [arguments] …]) → displays characteristics of a process.
* Kill (kill [-signumber] pid ..) → used to terminate a process.

Scheduling policy:

* Time-sharing technique → allowed to run concurrently.

Processes

* I/O – bound → make heavy use of I/O devices and spend much time wasting for I/O operations to complete.
* CPU-bound processes → are number crunching applications that require a lot of cpu time.

Process scheduling

* Interactive processes → interact constantly with users and spend a lot of time for key presses and mouse operations.
* Batch processes → don’t need user interaction and often run in background.
* Real-time processes → should never be blocked by lower-priority processes, have short response time.
* Nice (nice [option] [command] [arg]…) →runs a program with modified scheduling priority.
* Wait (wait [process id …]) →waits for child process to complete.
* Cron , crontab

***AWK***

Named after Aho, Weinberger, Kernigham , as a powerful programming language that can access, transform and format individual fields in records.

Awk <options> ‘line specifier {action}’ <files>

Awk variables list:

* $0 → contains full current record.
* $1 … $n →contains individual fields in current record.
* NF(number of fields), NR(number of records), FS(field seperator), OFS(output field separator), RS(input record separator), filename(holds file name), ARGC(arguments on command line), ARGV(list of arguments).
* Logical operator (&& , ||)and Relational operator (<, <=,==,!=,>=,> ,~,!~).
* Arithmetic operators (+,-,\*,/,%)

Begin (to begin the section) and End( to end the section).

Positional parameters(‘’) and shell variables

* Numeric functions → int(x),sqrt(x),index(s1,s2),length().
* String functions → substr(s1,s2,s3),split(s,a)

***Control statements:*** If, while

* Pattern matching (awk ‘/<pattern>’ <filename>) → print all lines with pattern.

***Shell Script programming:***

* Echo($echo <”text to print”>) → to print the entered text.
* Read(read <variable\_name>) → to read value to the variable.

Expr is generally used but let is more user-friendly. It is used in Bash and Korn shell.

Command line arguments parameters

1. $0 → name of executed command
2. $\* → complete set of positional parameters.
3. $# → number of arguments.
4. $$ → PID of current shell.
5. $! → PID of background job.
6. $? → exit status of last command.
7. $@ → used with strings in looping constructs.

Relational Operators:

* Eq(Equal to), ne(not equal to), gt(Greater than), gc(greater than or equal to), lt(less than), lc(less than or equal to).

String Operators:

* -n (not null), -z(null)

Logical Operators

* -a(AND), -o(OR), !(NOT)

File related operations

* -f <file> → file exists and regular file.
* -d <file> → file exists and directory file.
* -r <file> → file exists and readable file.
* -w <file> → file exists and writable file.
* -x <file> → file exists and executable file.
* -s <file> → file exists and size > 0.
* -e <file> → file exists.
* -z <file> → file empty or not.

Functions → to modularize the script, called as script module.

* Functionname() { Commands }

Arrays → collection of values accessible by individuals or groups.

* Arrayname{subscript}

SVN (SubVersion) → software versioning and revision control system.

Repository → central store of that system’s data in form of file system tree.

Svn log <file:///repository_path>

Svn import user\_directory\_path files:///repository\_path –m ‘Message’

Working copy(Check-out) → ordinary directory tree on local system, contain collection of files

Complete check-out → svn checkout <file:///repository_path>

Partial check-out → svn checkout <file:///repository_path/Directory_Path>

Lock-modify-unlock solution

Addition of file to repository → Svn add file\_path/file\_name file\_Path/file\_name

Commiting changes → svn commit –m ‘Message’

Locking file → svn lock file\_path/file\_name file –m “Message”

Un-locking file → svn unlock file\_path/file\_name.