Linux Lab Session: eDAC & eKDAC OS Concepts

Lab Session Day-01:

- When we install Linux OS, we need to create two partitions:

1. swap partition – is used by linux system as a swap arear in which inactive running programs temporarily.

- conventionally the size of swap partition/swap area should be doubles the size of main memory.
- if the size RAM = 4 GB => size of swap partition should be 8 GB.
- if the size RAM = 2 GB => size of swap partition should be 4 GB.
- if the size RAM = 8 GB => size of swap partition should be 16 GB.
.
.
.

- linux system use swap partition which is a portion of HDD as an extension of the main memory in which inactive running programs can be kept temporarily.

2. data partition "/" root (/) partition – in which linux system keeps/stores all data in an organized manner => filesystem i.e. linux filesystem follows heirarchical structure (inverted tree like structure).

(/) root dir : starting point/dir of linux filesystem structure.
- linux filesystem structure starts from root (/) dir.

- root dir i.e. "/" dir contains sub-dir's like: "/boot", "/bin", "/sbin", "/etc", "/lib", "/usr", "/home", "/root" "/mnt", "/dev" etc....

1. "/boot": contains static files of the boot loader, kernel, initrd (initial ramdisk).
- it contains information about booting the system
- it contains linux kernel by the name vmlinuz.

2. "/bin": contains user commands in a binary format, e.g. commands like ls, cat, cp, mv etc...

3. "/sbin": contains admin/system commands in a binary format, e.g. lscpu, adduser, deluser, mkfs etc...

4. "/home": contains home dir's of all users for multi-user system.
- for any user on its creation of an account by default subdir by the name of the user got created by the operating system, in which that user can store data, user can have read, write as well execute perms in that dir.

5. "/root": root is a home dir for root user

6. **"/etc":** contains host specific system configuration files - it is like a **"control panel"** in windows.
- in computing configuration files (or config files) are files used to configure the parameters and initial settings for some computer programs. They are used for user applications, server processes and operating system settings.
- e.g. **YaST: Yet another Setup Tool** (Linux operating system setup and configuration tool) or **debconf** (for performing system-wide configuration taskson UNIX-like operating systems).
"/etc/opt": contains configuration files for **add-on-packages** that are stored in /opt.
7. **"/dev":** contains device files

8. **"/lib":** contains shared libraries required for /bin and /sbin.

9. **"/mnt":** contains temporarily mounted filesystems

10. **"/media":** mount point for removable media such as CDROMs.

11. **"/opt":** contains optional application software packages.

12. **"/proc":** virtual filesystem provides process and kernel information as a files. In Linux , corresponds to a **procfs** mount. Generally automatically generated and populated by the system, on the fly.

- In Linux filesystem hierarchy, each file is uniquely identified by its path
path : it is a unique location of any file in a filesystem structure represented/denoted in a **string format** and file names/sub dir name are seperated by **delimiter char (delimiter).**

- there are 3 delimiter chars:
1. **slash (/)** - used in UNIX based OS's/Linux
2. **backslash (\ ) - used in Windows**
3. **colon (:)**

example:
"/" => root (/) dir
"/bin" =>
"/home/sunbeam/feb2021/eDAC"

"C:\sachin\edac"

"https://us06web.zoom.us/support/down4j"

- any file in a linux filesystem can be accessed by using 2 ways:
and hence there are two types of path:
1. **absoulte path:** it is a full path of any file which starts from **root (/) dir.**

i.e. path of the file with respect to root (/) dir.

2. relative path: path of the file with respect to current directory.

- whenever we create new dir, bydefault two hidden entries gets created inside a dir file.
. (single dot) => current dir
.. (two dots) => parent dir

- root (/) dir is a parent of itself
inode =>

- In Linux filename which starts with dot (.) [the period] is considered as a hidden file.

 # Basic Linux Commands:

Command Name: "pwd" - print/present working directory
- this command displays an absolute path (i.e. full path) of the present/current working directory
- pwd command internally refers the value of shell variable by the name "PWD".

Command Name: "cd" – to change current working directory
- this command is used to navigate from one location to another location.

$cd <dirpath> -- change dir to the dirpath
$cd / -- goto the root directory -- "/" slash -- is user short hand variable
$cd ~ -- goto the user's home dir -- "~" tild symbol -- is user short hand variable
$cd -- goto the user's home dir
$cd - -- goto the prev accessed directory
$cd . -- remains in current dir
$cd .. -- goto the parent dir of the current working dir

Command Name: "ls" - to display contents of the directory/list directory contents.

- contents of the directory files are nothing but name of files and sub-dir's.

$ls <dirpath> -- it displays contents of the dir by the name "dirpath".
$ls -- bydefault it displays contents of the current directory columnwise
$ls -l -- its displays contents of the dir in a listing format.

total n
n = total no. of data blocks allocated for the files and sub dir's in a given dir

feild-1: first char of feild 1 denotes type the file next 3 chars of field 1 denotes access perms for user/owner
next 3 chars of field 1 denotes access perms for group members last 3 chars of field 1 denotes access perms for group others.

feild-2: denotes no. of links exists for that file,
for **regular file** bydefualt no. of links i.e. **link count = 1.**
for **directory file** bydefualt no. of links i.e. **link count = 2**

feild-3 : denotes name of an user/owner
feild-4 : denotes name of group
feild-5 : denotes size of the file in bytes
feild-6 : denotes **time stamps** - date of creation and last modified date & time in ISO format.
feild-7 : name of the file
- In UNIX/Linux filename starts with **(.)** is considered as a **hidden file**
- the period (".")

options/flags/args description:
-h : displays size of the files in human redable format
-l : displays contents of the dir in a listing format
-a : display all contents of the dir (including hidden files)
-A : display all contents of the dir (including hidden files but excluding entries for "." & "..").
-i : display inode number of the files inode number is an unique idenfier of the file.
etc..... explore more options from man pages

Command Name: "cat" -
- cat command is used for concatenation of contents of file/files and print it onto standard output.
- cat command can be used to display contents of the regular file.

$cat > file1.txt
sunbeam infotech
pune
karad
<cntl+d> => to stop file writing

file by the name “file1.txt” created into the cur dir
and

“~” tild – users short hand operator => user’s home dir

$cat file1.txt => to display contents of file1.txt

$cat >> file1.txt => to append contents into the file – file1.txt

- cat command is not used for file editing.

Command Name: "cp" - to copy file/files

$cp <source> <destination>

Command Name: "mv" - command used to move file/s from one location to another location.

- this command is also used to rename file

Command Name: man – display manual pages and info about commands, system calls, library functions etc....

section-1 : contains info about shell commands
section-2 : contains info about system calls
section-3 : contains info abour library functions

+ Command Name: rmdir – remove empty dir only
- this command is used to remove empty dir/s
- if directory is not empty, rmdir command fails, in this case we can use rm command with -r option.

+ Comamnd Name: rm – remove file/s

- process => unique identifier => pid
- file => unique identfier => inode number

- all shell commands are the programs in a binary format
e.g. ls, cp, mv, mkdir
and options/flags/params are the command line arguments which we pass to the program while execution.

$./program.out one two three

$ls -l -i -s dirpath

ls – program
-l, -i -s, dirpath command line args
- all shell command are applications of command line argument programming.

File & filesystem related commands: ls, cp, mv, mkdir, cat, tac, touch, rmdir, rm, alias, unalias

+ Command Name: touch – to update timestamps

$touch filename.txt
- if filename.txt not exists, empty file gets created

+ redirection operators:
- there are 3 redirection operators
1. input redirection operator (<) - left cheveron
2. output redirection operator (>) - right cheveron
3. error redirection operator (2>) -

stdin – input buffer (which is there into the main memory) associated with standard input device i.e. kbd/conosole input

stdout – output buffer (which is there into the main memory) associated with standard ouput device i.e. monitor/console output

stderr – error buffer (which is there into the main memory) associated with standard output device i.e. monitor/console output

- bydefault any process reads input from **stdin** file
- bydefault any process writes output into **stdout** file
- if there are some errors, then list of errors gets written bydefault into the **stderr** file and those list of errors gets displayed onto the console.

Lab Session Day-02:

Command Name: "chmod" - used to change file mode bits

- to change mode bit of the file, by means of chaning mode bits i.e. to assign/ to remove access permissions of the file for user/owner, group members and others.
- we can do this by two ways:
1. by using human readable format
    +x : to assign execute perm to all (user + grp members + others)
    +r : to assign read perm to all (user + grp members + others)
    +w : to assign write perm to all (user + grp members + others)
    u+r: to assign read perm to user/owner
    u+w: to assign write perm to user/owner
    u+x: to assign excecute perm to user/owner
    g+r: to assign read perm to group members

-x : to remove execute perm from all (user + grp members + others)
-r : to remove read perm from all
-w : to remove write perm from all
u-r: to remove read perm from user/owner
u-w: to remove write perm from user/owner
u-x: to remove excecute perm from user/owner
g-r: to remove read perm from group members
.
.
.

2. by using octal format:

hexadecimal values:

r -- read -- 4
w -- write -- 2
x -- execute -- 1
.
.
.

- we want to assign :
user => read, write & execute
grp members => read & write
others => read & execute

$chmod 0765 india.txt

10 users:

a b c d e f g h i j

sunbeam : [c d e f g]

user – c => file1.txt => ~~rw-rw-r--~~

c becomes owner/user of the file file1.txt
grp members: d e f g => rw
others: a b h i j

pipe command (|) : pipe command is used to pass an output of one command as an input to another command on same shell.
- internally pipe command maintains buffer temporarily, into which an o/p of one comand can be kept and gets passed to the next command as an input.

uniq => to apply uniq command on the file which contains numbers, prequisite contents must be present in a numerically sorted manner.

Cascadding of linux commands by using pipe command: less, head, tail, sort, uniq,

PATH => its a shell variable in which path of binaries required for execution gets stored.
SHELL =>

pwd command prints current/present working dir => it displays an absolute path i.e. full path of cur working, internally pwd command refers the value of PWD shell variable.

PWD => its a shell variable -> it always kept an absolute of cur/present working dir.

$PWD =>

- if we want to execute any binary =>
way-1: we have to mention either its **absolute path** or its **relative path**

way-2: we have to go the parent dir in which binary is exists
$./program.out

- if we want to execute any linux command at any location in a filesystem structur, no need to go in a parent dir in which linux command is there in a binary format.

- when we run any command, shell refers value of PATH variable in which it contains paths of binaries.

SHELL, PWD, PATH
$SHELL => its value
$PWD => its value
$PATH => its value

Lab Session Day-03:

+ grep command => globally search regular expression and print
- grep command is used to search string/pattern/expression in file/files/filesystem, grep command searches for string/pattern/expression and it prints lines where desired string/expression/pattern is found

repetation operator in extended regular expression:

- 2+ yrs experience :
linux comamands: grep

+ interview questions:
tell me any 10 linux commands:
ls => contents of the dir files are dir entries of its files & sub dir's
ps
lscpu
top
ln
grep
cascadding of commands with pipe command
cut
tr
.....

Lab Exam => Linux Commands + Shell Script Programming
Assignments + Class Works

Shell Script Programming (by using bash shell)

Q. What is Shell?
- It is an application program which acts as an an interpreter i.e. interface between user & system.
- Shell program takes input from the user in the text format (i.e. commands) and execute it i.e. pass it to kernel for execution.
- In Linux System multilple shell programs may exists, but only one shell program may be active at a time.

Examples of shell program in a Linux System: sh compatible shell programs open source:

bsh – bourne shell
bash - bourne again shell
ksh – korn shell
csh – c shell

- Command to check currently active shell program:
$echo $SHELL

- Command to check available shell programs in a system
$ls -l /bin/*sh

- Command to change current active shell:
$chsh <shellname>

- In Linux dot extension of any doesnt matter

shell script => it is a simple text file which contains set of commands, instructions in its own syntax and binaries gets executed by means of executing text file at once.

Shell Script file is same as Batch file in Windows

step-1: write as script/set of commands into the simple text file by using any editor and save the file by giving . dot extension as .sh for sake convenience.
step-2: assign execute permissions for that script
    $chmod +x scriptname.sh

step-3: execute the script
    $./scriptname.sh   OR $bash scriptname.sh

- we can write scripts simply in any editor => vi editor/vs code/gedit

- in shell script programming there are 3 types of quotes:
1. double quotes => " " => string
2. single quotes => ' ' => char constant
3. back quotes => ` ` => if we want to assign/store output of command into any variable then we need to mention that whole command in a back quotes.

=> We can write shell script programs by using any editor on Linux:
gedit => like notepad in windows
vi editor => if you know how to use/operate it
vs code editor => you need to install it on vb

- if currently active shell is bash – to mention shabang line is not mandatory

- in C if we want to use any variable/function, first we have to define it
- in shell script programming no need to define variable => shell programming language is typeless programming language, no need to mention data type of any variable explicitly.

Lab Session Day-04:

commands: chsh, echo, read, expr, bc
shebang line
variable
constants
assignment operator
shell script programming laguage typeless

if(yr % 4 == 0 && yr % 100 != 0 || yr % 400 == 0)

relational operators:
-eq : equal equal to operator
-ne : not equal to operator
-lt : less than
-le : less than or equal to
-ge : greater than or equal to
-gt : greater than

logical operators:
-a : logical AND
-o : logical OR

if statement:

```
if [ condition ]
then
        statement/s
fi
```

#if-else statement:

```
if [ condition ]
then
        statement/s
else
        statement/s
fi
```

test commands:
-e filepath : to check the filepath is valid or invalid
-d filepath : to check the filepath is directory file or not
-f filepath : to check the filepath is a regular file or not
-x filepath : to check the filepath has execute perms or not
-r filepath : to check the filepath has read perms or not
-w filepath : to check the filepath has write perms or not

if-else ladder :

```
if [ cond ]
then
elif [ cond ]
then
        statement/s
elif [ cond ]
then
        statement/s
else
        statement/s
fi
```

loops:

1. while loop:

```
while [ cond ] # if cond is true then only control goes inside while loop
do

        statement/s

done
```

- as soon as cond become false while loop gets break

2. until loop:

```
until [ cond ] # if cond is false then only control goes inside while loop
do

        statement/s

done
```

- as soon as cond become true until loop gets break

3. for loop:

```
for var_name in {collection}
do
        statement/s
done
```

- collection is like an array – collection of similar type of data elements
- in each iteration for loop takes value for var_name from the collection sequentially

```
for filepath in {file1.txt  script_01.sh  script_03.sh  script_05.sh  script_07.sh
file2.txt  script_02.sh  script_04.sh  script_06.sh  script_08.sh}
```

directory file => rwx

if dir file has read perm => we can apply “ls” command on it

if dir file has write perm => we can create sub dir’s and files inside it

if dir file has execute perm => we can apply cd command on it

positional parameters in shell script programming <==> command line arguments in C

C => arguments which we pass through command line to an executable file while executing it are called as command line arguments

Shell => arguments which we pass through command line to the script while executing it are called as positional parameters

C :
argv[0] : name of an executable file : string
argv[1] : arg1
argv[2] : arg2

.
.
.

argc : argument counter => total no. of command line args including name of an executable file

Shell:

$# : no. of positional parameters => no. of pos param (excluding name of the script).

$0 : name of the script
$1 : positional param-1
$2 : positional param-1
.
.
.

$* : all positional parameters

function => an independent block statement/s which does specific task and statements inside it gets executed only after giving call to that function.

```
function function_name( )
{
        statement/s
}
```

- we have to define function at the begining of the script and statements/instructions/commands inside function gets executed only after giving call to that function.

Inside any function definition:
$1, $2, $3, ... => arguments which we pass to the function

Outside the function definition(globally)
$1, $2, $3, ...=> positional params

```
tab={
        table of 20 is :
        20
        40
        60
        80
        100
        120
        140
        160
        180
        200
}
```

switch control block:

we are applying switch control block on value of var

```
case $var in
1)
        statement/s
        ;;      # its like a break statment of case

2)
        statement/s
        ;;

3)
        statement/s
        ;;

*)              # default case
        ;;

esac
```

***** If you are expert in C programming language you can become expert in any programming language.

UNIX => C & Assembly Language
Linux, Windows => C, C++ & Assembly language.