Mo Tu We Th Fr Sa Su

# Linux

Memo No.

Date 27 / 07 / 2024

touch → creates new file

'cd' → change directory

'cat' → reads file

'grep' → search  grep [to-search] filename.

'ls' → lists all files in directory  'ls -a' for hidden files

'find' → find files  find path -name {to-search}

filter

→ link files with

Symbolic } ln -s [org.file-path] [link-path]

'man' → Gives data about usage of func"s  { man cat

↳ press 'q' to quit

↳ Search with '/' search backwards with '?'

↳ 'n' → next result, 'N' → prev. result.

is man -k [keyword] & Searches for manuals

- Some programs don't have manpage but description maybe given if we use --help, -help, -h, -?, help depending on program.

- for builtins we use: 'help [Built in]'

Mo Tu We Th Fr Sa Su

Memo No.
Date / /

# Practicing Piping

echo hi > asdf → default 1> → FD no.
↳ redirects o/p
↳ writes hi in a file asdf.

aggregate → run a bunch of commands.

→ '>>' will append

· File Descriptor (FD): no. the describes a communication channel in Linux.

FD: 0 → Standard i/p
" 1 → " o/p
" 2 → " error

some_command > output.log 2> errors.log
↳ command will redirect o/p to output.log & errors to errors.log

'<' → redirects i/p

'|' → Std o/p from left of operator will be piped into Std i/p of the command to the right.
pipe operator

Mo Tu We Th Fr Sa Su

Memo No.
Date / /

'>&' → redirects a file descriptor (fd) to another fd.
→ eg 2>&1 => redirects stderr to stdo/p

tee → dupes data flowing through pipe to any no. of files
echo hi | tee file1 file2 file3 .... file n.

Note

- /challenge/hack is a command, but if you want to pass it as a file use >(/challenge/hack)
  ↳ useful in tee as it only takes files as arguments
- $ should be prepended to variable (echo $VAR ✓ echo VAR ✗)
- variable=value NO SPACES IN B/W
- for multiple words use "" (for example")

'sh' → makes child shell
↳ Wont be able to access same variables as main shell
↳ To access we have to export variable

export VAR=53

env → Prints out every exported variable set in your shell.

→ Command Subs": Allows to capture o/p of any command as an argument to another command.

Eg: Flag = $(cat /flag)
echo "$Flag"
-o/p of cat

Mo Tu We Th Fr Sa Su

Memo No. _____
Date / /

read variable_to_store_in → inputs

read -p "prompt" → prompts a message before i/p

Eg: read -p "Name: " NAME

o/p → Name: Butcher
i/p

→ echo "test" > some_file o/p of echo → file

read VAR < some_file file data → VAR
read into

## Processes & Jobs

ps → process snapshot / status, & lists processes.

- PID → Process ID (A Numerical Identifier)
  - a no. that uniquely identifies every running process in a linux environment.
- Time → total cpu time the process has eaten
- TTY → terminal on which the commands are running

### ps -ef

-e → lists every process

-f → full format o/p

→ Standard Syntax

### ps aux

a → list processes for all users

u → " that are running in a terminal

x → for user-readable o/p

→ BSD Syntax

Mo Tu We Th Fr Sa Su

Memo No. _____
Date / /

STIME → Start time of process

PPID → PID of parent process

kill PID → kills process with that PID

- When a process clogs up the terminal (eg. by an application waiting for i/p) use ctrl+c to interrupt.
- Ctrl+2 suspends program
- fg resumes program in foreground
- bg suspend background
- When you ~~puta~~ a process its STAT in ps -o become T. (suspended due to Ctrl+2)

S → process is sleeping while waiting for i/p

R → Actively running

R+ → " in foreground

process & → Starts process in bg.

∴ Perceiving Permissions

ls -l → shows permissions of files/directories

↳ '-' represents normal file c → character device.

d " directory

chown → change Owner. chgrp → change Group {need write access for file & membership in that group

chown [username] [file]

Mo Tu We Th Fr Sa Su

Memo No.
Date / /

- in ls-l ; 1st character → file type
  next 3 " s → permissions for owning user
  " → " group } access perm
  " → all other access

Each char of the three represents permission of a diff type.
r - user/grp/other can read file/ list/dir
w - " modify file & create/delete files in dir
x - " execute the file as a program / can enter dir
- - nothing

Eg: rw-r--r-- diiodes,
r - user that owns the file can read it
w - " write to it
- - " cannot execute it
r - users in the grp that owns the file can read it
- - " cannot write to it
- x " cannot execute it
r - All other users can read it
- - " cannot write to it
- - " cannot execute it

chmod → change mode (to change file perm)
chmod [OPTIONS] MODE {FILE} Eg: chmod u+r /play
options allow to tweak perms with the mode format
who +/- what

Mo Tu We Th Fr Sa Su

Memo No.
Date / /

Eg: u+r → adds read access to users perm.

a-rwx → removes all permissions for users/grps/world

[u, g, o, a] +/- [r, w, x]

- You need write access to file to change perm.

→ Permission Setting

u=rw → sets read & write perm for user & wipe execute perm

o=x → sets only execute perm

- For multiple

u=rw, g=r

- Zero out perms by

u=rw, g=, o=

- Set User ID (SUID) allows user to run a program as the owner of that programs file.

-rws r-xr-x → The s part in executable bit means that the program is executable with SUID.

→ The program will execute as the owner user.

WARNING: giving SUID bit to an executable owned by root can give attackers a possible attack vector to become root.

Mo Tu We Th Fr Sa Su

Memo No.
Date / /

/etc/passwd → Gives full list of users. → globally readable
→ 1st field is username, 2nd is password
/etc/shadow → same as above, separately!
↳ * / ! means password login is disabled
↳ blank ⇒ no password

su → switch User

john [dir/file] → decrypts password

sudo → uses policies to determine user's authorization rather than password.

; → helps chain commands

- Use '.sh' files to store a list of commands in a file to execute at once.
  ↳ To execute bash x.sh
  ↳ You can pipe '|' these scripts to other commands too

- Nano is a text editor in terminal

→ You can execute without bash if you make the file executable

PATH → Stores a bunch of dirs
PATH="" → deletes all dirs

- if you add a script to PATH it can be launched with bare name (w/o whole path ./.../script).
  PATH= dir of script

Mo Tu We Th Fr Sa Su

Memo No.
Date / /

PATH = $PATH:(dir) $ `pwd` : if its current dir

is append

PATH = $(dir):$PATH → prepends