Linux Crash Course – Extended Student Handout

1. Introduction to Linux and Filesystem

- Linux is a multiuser, multitasking operating system used widely in servers and development environments.
- The Linux filesystem is hierarchical, starting from the root directory `/`.
- - `/bin`: Essential binary executables
- - `/etc`: Configuration files
- - `/home`: Home directories of users
- - `/var/log`: System logs
- - `/usr`: User programs and libraries
- Use `pwd` to print the current directory and `ls -l` to list contents in detail.
- `cd` is used to navigate directories, and `mkdir`/`touch` to create directories and files.

2. Shell Basics and Navigation

- The shell is a command-line interpreter. The default shell is usually `bash`.
- Important shell commands include:
- - `echo`: Print text
- - `man`: View manual pages for commands
- - `alias`: Create command shortcuts (e.g., `alias ll='ls -la'`)
- - `history`: View command history
- `nano` and `vi` are text editors used to modify files.

3. File Permissions, Ownership & chmod

- Linux permissions follow the `rwx` model for Owner, Group, and Others.
- `ls -l` displays permissions: `-rwxr-xr--` means:
- - Owner has read (r), write (w), execute (x) = 7
- - Group has read (r), execute (x) = 5
- - Others have read (r) only = 4
- Use `chmod` to change permissions numerically or symbolically:
- - `chmod 755 file.sh` makes the file executable for owner and readable for others
- - `chmod +x script.sh` adds execute permission
- Use `chown` to change file ownership and `chgrp` to change group.
- Linux uses a permission model to control access to files and directories. Each file or directory has three sets of permissions:
- - User (owner)
- - Group (assigned group of users)
- - Others (everyone else)
- Each set of permissions is represented by three characters: r (read), w (write), and x (execute). For example: `rwxr-xr--` means:
- - User can read, write, and execute
- - Group can read and execute

- - Others can only read
- Each permission has a numeric value:
- - r = 4
- - w = 2
- - x = 1
- You calculate the permission number by adding these values:
- - rwx = 4+2+1 = 7 (full access)
- - rw- = 4+2 = 6 (read/write)
- - r-- = 4 = 4 (read only)
- These combinations form the numeric representation used in `chmod`. Some common ones:
- - 777 = rwxrwxrwx (everyone has full access)
- - 755 = rwxr-xr-x (owner has full, others can read/execute)
- - 700 = rwx------ (only owner has access)
- - 644 = rw-r--r-- (owner can read/write, others read-only)
- - 600 = rw------- (private file, used for SSH keys etc.)
- Public files (like web pages) are typically 644 or 755 depending on whether they're directories or static content.
- Group permissions are used to manage access among team members: assign users to a group and set group access accordingly.
- Use `chmod` to set permissions numerically: `chmod 755 file.sh`
- You can also use symbolic mode: `chmod g+w file.sh` to give group write access.
- `chown` changes the file owner: `chown alice file.sh`
- `chgrp` changes the group: `chgrp devteam file.sh`
- Always use the least amount of privilege needed. Avoid 777 unless absolutely required and in trusted environments.
- Here is a breakdown of the permission numbers from 0 to 7 and what each represents:
- - 0: --- (No permissions)
- - 1: --x (Execute only)
- - 2: -w- (Write only)
- - 3: -wx (Write and execute)
- - 4: r-- (Read only)
- - 5: r-x (Read and execute)
- - 6: rw- (Read and write)
- - 7: rwx (Full access)
- These numbers are used for each class: user, group, and others.
- For example, `chmod 751 file.sh` means:
- - User: 7 (rwx)
- - Group: 5 (r-x)
- - Others: 1 (--x)

4. Users, Groups, and Shell Environment

- User data is stored in `/etc/passwd`, and encrypted passwords in `/etc/shadow`.
- Each user has a UID (User ID), and root has UID 0.
- Use `whoami`, `id`, and `groups` to get information about users and their privileges.
- Shell environment variables include `$PATH`, `$HOME`, and are set using `export`.
- Startup configuration files:

- - `~/.bashrc`, `~/.profile` (user-specific)
- - `/etc/profile` (system-wide)

5. Process and Job Management

- Use `ps aux`, `top`, and `htop` to monitor processes.
- `kill ` sends signals to terminate processes.
- Background execution: use `&` at the end of a command.
- `jobs`, `fg`, `bg` are used to manage background and foreground jobs.
- Example:
- - `sleep 100 &` runs a sleep job in the background
- - `fg` brings it back to foreground

6. Disk, Memory and System Info

- Disk usage: `df -h` shows file system disk space, `du -sh *` shows directory sizes.
- Memory usage: `free -m` and `top` show system memory status.
- System info commands include `uname -a`, `uptime`, `hostname`, `who`, `w`, and `logname`.

7. Archiving, Searching and File Utilities

- `tar -xvzf archive.tar.gz` extracts a compressed archive.
- `zip` and `unzip` work for .zip files.
- Search tools:
- - `grep`: Searches inside files (e.g., `grep 'error' logfile.txt`)
- - `find`: Locates files in directories (e.g., `find / -name '*.sh'`)
- - `locate`: Uses an indexed database, faster than find
- `wc -l file.txt` counts lines in a file.

8. Scheduling Jobs with cron and at

- `cron` is used for recurring tasks, `at` for one-time tasks.
- Use `crontab -e` to edit user's crontab. Format:
- `* * * * * command` => min hour day month day_of_week
- Example:
- - `0 6 * * 1 /home/user/backup.sh` runs every Monday at 6 AM
- `at now + 1 minute` schedules a job to run once in a minute.
- `crontab -l` lists your cron jobs.