CSE 362: Operating Systems Lab

Fun with Linux Shell Commands

"The command line is your superpower."

Introduction

- A good **command line interface** is a marvelously expressive way of communicating with a computer in much the same way the written word is for human beings.

- **Graphical user interfaces** make easy tasks easy, while **command line interfaces** make difficult tasks *possible*.

- The **shell** is a program that takes keyboard commands and passes them to the operating system to carry out.

- Almost all Linux distributions supply a shell program from the **GNU Project** called `bash`.

Starting up

Launch terminal. You'll see something like:

```
username@machinename~$
```

Here `!~ indicates the current working directory and \cmd{$} indicates a normal user. If the last character is \cmd{#} then it means a superuser .` (*These will be clear to you very soon!*)

Hello World

```
echo Hello World
echo ‘Hello World’
echo "Hello World"
```

It is not essential to surround the string with quotes. But doing so also doesn't repeat them on the screen.
Actually `echo` can do lots of other powerful things. They will come later.

More Examples

```
echo "Today is $(date)"
echo "User: $USER, Home: $HOME"
echo -e "Line1\nLine2\nLine3"
echo -n "No newline here"
```

- Single quotes '' → literal string
- Double quotes "" → allow variable/command substitution
- `-e` enables backslash escapes (`n`, `t`)
- `-n` suppresses newline

Tasks

1. Print your full name and current date in one line.
2. Print a message with a **tab** between two words.
3. Print three lines using one `echo` command.
4. Print your username using a variable.

Answers: Hello World Tasks

```
   echo "John Doe, $(date)"
   echo -e "Hello\tWorld"
   echo -e "Line 1\nLine 2\nLine
3"
   echo "Username: $USER"
```

Exploring

`pwd` prints the current directory `cd <directoryname>`

- moves to *directoryname*
- Path can be absolute or relative (be careful!!)
- `.` means the current directory
- `..` means the parent of the current directory
- Default directory is `HOME`
- `HOME` also represented by `!~`
- Can return `HOME` from anywhere by entering `cd`

```
cd /tmp
cd Documents
cd ..
cd
cd ~
cd ./Documents
cd ../parent_dir
```

Tasks

1. Go to `/tmp`, print path, return home.
2. Create a folder `practice` in your home, navigate into it.
3. From `practice`, go to parent, then back using `..` and `.`.

Answers: Exploring Tasks

1. ```
   cd /tmp
   pwd
   cd
   ```

2. ```
   mkdir ~/practice
   cd ~/practice
   ```

3. ```
   cd ..
   cd ./practice
   ```
   ```

Exploring

`ls [OPTION] [FILE]`

- Lists information about directory or file
- In Linux hidden files have names starting with `.` and they are not shown by `ls`
- `ls -a` lists the hidden files also
- `ls -l` lists in details
- `ls -R` recursively lists subdirectories
- `ls -S` sorts files by size

More Examples

```
ls -l
ls -a
ls -la
ls -lh # human-readable sizes
ls -lt # sort by time (newest
  first)
ls -lS # sort by size
ls *.txt # list only .txt files
```

```
ls -d */ # list directories only
```

Tasks

1. List all files (including hidden) in long format.
2. List only directories in `/$usr$`.
3. Show files in `/var/log` sorted by size.
4. Find all `.conf` files in `/etc`.

Answers: ls Tasks

1. `ls -la`
2. `ls -d /usr/*/`
3. `ls -lS /var/log`
4. `ls /etc/*.conf`

My friend man

`man command`

- Shows the manual for *command*
- You should frequently use it besides Google.
- Now you can see all the options associated with `ls` by

  `man ls`

- You can learn more from `man man`

Files and Directories

`mkdir [OPTION] <directory1> <directory2>`

- Makes each directory if it already doesn't exist
- `-p` Overwrite directory even if it exists.
- `-v` Shows a verbose description

```
mkdir newfolder
mkdir -p parent/child/grandchild
mkdir -v dir1 dir2 dir3
```

Tasks

1. Create: `projects/python/web` in one command.
2. Create three folders: `backup1`, `backup2`, `backup3` with verbose output.

Answers: mkdir Tasks

1. `mkdir -p ~/projects/python/web`
2. `mkdir -v backup1 backup2 backup3`

Files and Directories

cp

- Copy files and directories
- `cp SOURCE DEST` copy `SOURCE` to `DEST`
- `cp SOURCE DIRECTORY` copy multiple `SOURCE`s to `DIRECTORY`
- `-r` copy directories recursively
- `-i` interactive i.e. prompts before overwriting

```
cp file.txt backup/
cp -r folder/ backup_folder/
cp *.log logs/
cp -i important.txt safe/
cp -v file1 file2 file3
  destination/
```

Tasks

1. Copy all `.txt` files from `docs/` to `backup/`.
2. Copy `config/` directory recursively to `/tmp`.

3. Try overwriting a file with and without `-i`.

Answers: cp Tasks

1. `cp docs/*.txt backup/`
2. `cp -r config/ /tmp`
3. `cp file.txt backup/file.txt`
   `cp -i file.txt backup/file.txt`

Files and Directories

`rm [OPTION]... FILE...`

- Remove each of `FILE`s if exists
- `-f` ignores non-existent files, never prompts
- `-i` interactive i.e. prompts before deleting
- `-r` remove contents recursively
- `-v` shows verbose description

```
rm temp.txt
rm -f nonexistent.txt
rm -i important.txt
rm -r temp_dir/
rm -rv old_logs/
```

Warning: `rm -rf /` can destroy your system!

Tasks

1. Create a file, delete it interactively.
2. Make a directory `trash`, put 3 files, delete recursively with verbose.
3. Delete all `.tmp` files in `/tmp` (safely!).

Answers: rm Tasks

1. ```
   touch test.txt
   rm -i test.txt
   ```

2. ```
   mkdir trash
   touch trash/file1 trash/file2 trash
   /file3
   rm -rv trash
   ```

3. ```
   rm -i /tmp/*.tmp
   ```
   ```

Files and Directories

`mv`

- `mv SOURCE DEST` moves file from `SOURCE` to `DEST`
- `...mv SOURCE ODIRECTORY` moves files to `DIRECTOR`
- `-i` interactive i.e. prompts before overwriting

```
mv old.txt new.txt
mv file1 file2 /destination/
mv -i data.txt archive/
mv *.jpg photos/
```

Tasks

1. Rename `report.txt` → `final_report_v1.txt`.
2. Move all `.png` images to `images/` folder.
3. Try moving a file to existing name with/without `-i`.

Answers: mv Tasks

1. `mv report.txt final_report_v1.txt`

2. ```
   mkdir -p images
   mv *.png images/
   ```

3. ```
   mv file.txt backup/file.txt
   mv -i file.txt backup/file.txt
   ```

Tracking Location

Pushd Popd

```
pushd /var/log
pushd /etc
popd
popd
dirs -v # view stack
```

Tasks

1. Navigate: home → /usr → /bin → /tmp using pushd.
2. Use popd twice to return.
3. Print directory stack.

Answers: pushd/popd Tasks

1. ```
   cd ~
   pushd /usr
   pushd /bin
   pushd /tmp
   ```

2. ```
   popd
   popd
   ```

3. ```
   dirs -v
   ```
   ```

Files and Directories

`touch [OPTION] FILE...`

- Creates an empty file if it does not exist
- Updates access/modification timestamps if it does
- `-a` change only access time
- `-m` change only modification time
- `-t STAMP` use [[CC]YY]MMDDhhmm[.ss]

```
touch newfile.txt
touch -t 202501011200 oldstamp.txt
touch file1 file2 file3
```

Tasks

1. Create three empty files: `a.txt`, `b.txt`, `c.txt`.
2. Change only the access time of `a.txt` to now.
3. Create a file with timestamp 1 Jan 2025 12:00.

Answers: touch Tasks

1. `touch a.txt b.txt c.txt`
2. `touch -a a.txt`
3. `touch -t 202501011200 jan2025.txt`

Permissions

`chown [OPTION]... [OWNER][:GROUP] FILE ...`

- Change file owner and/or group
- `-R` recursive
- `--reference=RFILE` copy owner/group from RFILE

```
sudo chown alice report.txt
sudo chown -R bob:devs /projects/
  app
sudo chown --reference=/etc/passwd
   copy.txt
```

Tasks

1. Change owner of `log.txt` to `alice`.
2. Recursively change group of `web/` to `www-data`.

Answers: chown Tasks

1. `sudo chown alice log.txt`
2. `sudo chown -R :www-data web/`

File Viewing

`sort [OPTION]... [FILE]...`

- Sort lines of text files
- `-r` reverse order
- `-n` numeric sort
- `-k FIELD` sort by specific field
- `-u` unique lines

```
sort names.txt
sort -r numbers.txt
sort -n scores.txt
cat access.log | cut -d' ' -f1 |
  sort | uniq -c | sort -nr
```

Tasks

1. Sort `/etc/passwd` by UID (field 3).
2. Sort `scores.txt` numerically in descending order.
3. Remove duplicate lines from `list.txt`.

Answers: sort Tasks

1. `sort -t: -k3 -n /etc/passwd`
2. `sort -nr scores.txt`
3. `sort -u list.txt > unique.txt`

User Management

`su [OPTION] [USER]`

- Switch user (default root)
- `-` or `-l` login shell (load environment)

`sudo COMMAND`

- Execute command as superuser (or another user)

```
su - # become root with full
  environment
sudo apt update
sudo -u alice whoami
```

Tasks

1. Switch to root with login shell.
2. Install `htop` using `sudo`.
3. Run `whoami` as user `testuser`.

Answers: su / sudo Tasks

1. `su -`
2. `sudo apt install htop`
3. `sudo -u testuser whoami`

Process Management

`ps [OPTION]`

- Report a snapshot of current processes
- `aux` BSD style (all users)
- `-ef` System V style
- `-u USER` processes of USER

```
ps aux
ps -ef | grep nginx
ps -u $USER
```

Tasks

1. List all processes of current user.
2. Show full command line of PID 1234.
3. Count how many `bash` processes are running.

Answers: ps Tasks

1. `ps -u $USER`
2. `ps -p 1234 -o cmd=`
3. `ps aux | grep bash | wc -l`

Process Management

`kill [SIGNAL] PID...`

- Send signal to processes
- Common signals: `TERM` (default), `KILL`, `HUP`

`killall [OPTION] NAME...`

- Kill processes by name

```
kill 1234
kill -9 5678 # force kill
killall -HUP nginx
```

Tasks

1. Gracefully stop process with PID 4321.
2. Force-kill a stuck `python` script.
3. Restart all `apache2` processes.

Answers: kill / killall Tasks

1. `kill 4321`
2. `killall -9 python`
3. `killall -HUP apache2`

Text Editors

`vim [OPTION] FILE`

- Powerful modal editor
- Normal → Insert → Visual → Command
- `+<num>` start at line <num>

```
vim notes.txt
vim +42 largefile.c
```

- `i` insert, `:w` save, `:q` quit, `:wq` save+quit

Tasks

1. Open `/etc/hosts` in vim and go to line 10.
2. Create a new file `hello.c` and write a `main()` function.

Answers: vim Tasks

1. `vim +10 /etc/hosts`
2. `vim hello.c` → press `i`, type:

```
#include <stdio.h>
int main() {
    printf("Hello, World!\n");
    return 0;
}
```

→ press `Esc`, then `:wq`

Text Editors

`nano [OPTION] FILE`

- Simple, beginner-friendly editor
- On-screen shortcuts (Ctrl-O save, Ctrl-X exit)

```
nano todo.txt
nano -w largefile.txt # disable
  line-wrap
```

Tasks

1. Edit `/.bashrc!~ with nano and add an alias.`
2. Create a new file `notes.txt` and write three lines.

Answers: nano Tasks

1. `nano ~/.bashrc` → add:

```
alias ll='ls -la'
```

→ `Ctrl+O`, `Enter`, `Ctrl+X`

2. `nano notes.txt` → type:

```
Buy milk
Call mom
Finish lab
```

→ `Ctrl+O`, `Enter`, `Ctrl+X`

Network

`ping [OPTION] HOST`

- Send ICMP echo requests
- `-c COUNT` stop after COUNT packets
- `-i INTERVAL` wait interval seconds

```
ping -c 4 google.com
ping -i 0.5 192.168.1.1
```

Tasks

1. Ping `8.8.8.8` four times.
2. Ping your router with 0.2 s interval.

Answers: ping Tasks

1. `ping -c 4 8.8.8.8`
2. `ping -i 0.2 192.168.1.1`

Network

`traceroute [OPTION] HOST`

- Print the route packets take to host
- `-n` do not resolve IP addresses to hostnames
- `-m MAX` max hops

```
traceroute google.com
traceroute -n -m 15 1.1.1.1
```

Tasks

1. Trace route to `cloudflare.com` without DNS.
2. Limit hops to 10 for a local server.

Answers: traceroute Tasks

1. `traceroute -n cloudflare.com`
2. `traceroute -m 10 192.168.1.1`

Searching

`find [PATH] [EXPRESSION]`

- Search for files in a directory hierarchy
- `-name PATTERN` case-sensitive name
- `-iname PATTERN` case-insensitive
- `-type f/d` file or directory
- `-exec CMD ;` run command on matches

```
find . -name "*.txt"
find /var/log -type f -mtime -7
find . -name "*.conf" -exec cp {}
  /tmp \;
```

Tasks

1. Find all `.log` files modified in the last 2 days.
2. Locate every `Makefile` under `/usr/src`.
3. Delete all `*.tmp` files in `/tmp` (use `-delete`).

Answers: find Tasks

1. `find / -type f -name "*.log" -mtime -2 2>/dev/null`
2. `find /usr/src -name Makefile`
3. `find /tmp -name "*.tmp" -delete`

Terminal Control

`clear`

- Clear the terminal screen
- Shortcut: `Ctrl+L`

```
clear
```

Tasks

1. Clear the screen after a long `ls -R`.

Answers: clear Tasks

1. ```
   ls -R /etc
   clear
   ```
   ```

Terminal Control

`history [N]`

- Show command history list
- N! re-execute command number N
- !! last command

```
history 10
!235
!!
```

Tasks

1. Show the last 5 commands.
2. Re-run the command that created `myproject`.

Answers: history Tasks

1. `history 5`
2. mkdir! (or check history and use number)

Add and Remove User

```
sudo adduser username
sudo usermod -aG sudo username
sudo deluser username
sudo deluser --remove-home
  username
less /etc/passwd
```

Tasks

1. Add user `testuser` with home directory.
2. Add `testuser` to `sudo` group.
3. List all users.
4. Remove `testuser` and their home directory.

Answers: User Management Tasks

1. `sudo adduser testuser`
2. `sudo usermod -aG sudo testuser`
3. `cut -d: -f1 /etc/passwd`
4. `sudo deluser --remove-home testuser`

Permissions

- Linux is a **multiuser** system.
- A user may own files and directories.
- Users belong to groups.
- Access rights: read, write, execute.

`ls -l` output:

```
-rwxr-xr-- 1 alice devs 4096 Nov 7
  10:30 script.sh
```

Part	Meaning
`-rwxr-xr--`	File type + permissions
`1`	Number of hard links
`alice`	Owner
`devs`	Group
`4096`	Size
`Nov 7 10:30`	Modified time
`script.sh`	Name

Permissions

chmod – change the mode (permissions)

Octal Notation

```
chmod 644 file.txt # rw-r--r--
chmod 755 script.sh # rwxr-xr-x
chmod 700 private/ # rwx------
chmod 600 secrets.txt # rw-------
```

Symbolic Notation

```
chmod u+x script.sh
chmod g-w file.txt
chmod o+r shared.txt
chmod go-rwx secret/
chmod a+r public.txt
```

Tasks

1. Create `runme.sh`, make it executable for owner only.

2. Make `data.txt` readable/writable by owner, read-only for others.

3. Give group execute permission on a directory.

Answers: chmod Tasks

1. ```
   touch runme.sh
   chmod 700 runme.sh
   ```
2. ```
   chmod 644 data.txt
   ```
3. ```
   chmod g+x mydir/
   ```
   ```

Permissions

```
sudo chown alice report.txt
sudo chown -R bob:devs /projects/
  app
sudo chgrp admins config/
```

Tasks

1. Change owner of `log.txt` to `alice`.
2. Recursively change group of `web/` to `www-data`.

Answers: chown/chgrp Tasks

1. `sudo chown alice log.txt`
2. `sudo chgrp -R www-data web/`

File Viewing

Command	Use
`cat`	Concatenate & display
`less`	View with navigation
`more`	Simple pager
`head`	First N lines
`tail`	Last N lines
`wc`	Word/line/byte count
`grep`	Search pattern

```
cat /etc/passwd
less /var/log/syslog
head -5 access.log
tail -10 error.log
tail -f /var/log/auth.log
wc -l file.txt
grep "error" app.log
```

Tasks

1. Show first 3 lines of `/etc/hosts`.
2. Follow live changes in `auth.log`.

3. Count lines in `passwd`.
4. Find all lines containing `bash` in `/etc/passwd`.
5. Search recursively for `main()` in `.c` files.

Answers: File Viewing Tasks

1. `head -3 /etc/hosts`
2. `tail -f /var/log/auth.log`
3. `wc -l /etc/passwd`
4. `grep bash /etc/passwd`
5. `grep -r "main(" *.c`

I/O Redirection

- `>` Overwrite file
- `>>` Append to file
- `<` Input from file
- `2>` Redirect stderr
- `&>` Redirect stdout + stderr

```
echo "Hello" > greeting.txt
ls nonexistent >> output.log 2>&1
cat < input.txt
sort < names.txt > sorted.txt
find / -name "*.conf" 2>/dev/null
```

Tasks

1. Save `ls -l` output to `listing.txt`.
2. Append current date to `log.txt`.
3. Run a failing command → redirect error only.
4. Combine stdout and stderr into one file.

Answers: I/O Redirection Tasks

1. `ls -l > listing.txt`
2. `date >> log.txt`
3. `ls /fake 2> error.log`
4. `ls /fake &> combined.log`

Pipelines

```
ls -l /etc | grep conf | wc -l

ps aux | grep nginx | awk '{print
  $2}'

cat access.log | cut -d' ' -f1 |
  sort | uniq -c | sort -nr
```

Tasks

1. List all `.sh` scripts in `/bin`, count them.
2. Show top 5 largest files in current directory.
3. Extract unique IP addresses from `access.log`.
4. Find all users with `/bin/bash` shell from `/etc/passwd`.

Answers: Pipelines Tasks

1. `ls /bin/*.sh | wc -l`
2. `ls -lS | head -6`
3. `cut -d' ' -f1 access.log | sort | uniq`
4. `grep "/bin/bash" /etc/passwd | cut -d: -f1`

Final Challenge: Mini File Manager Script

Create `myfm.sh`:

1. Creates a directory `myproject`
2. Creates 3 files inside
3. Copies one to `backup/`
4. Lists contents with permissions
5. Changes permission of one file to `700`
6. Appends system info to `log.txt`
7. Displays log using `less`

```
chmod +x myfm.sh
./myfm.sh
```

Answer: Final Challenge – myfm.sh

```
#!/bin/bash
mkdir -p myproject backup
touch myproject/file1.txt
  myproject/file2.txt myproject/
  report.txt
cp myproject/report.txt backup/
ls -l myproject
chmod 700 myproject/file1.txt
echo "System: $(uname -a)" >> log.
  txt
echo "Uptime: $(uptime)" >> log.
  txt
less log.txt
```

- Save as `myfm.sh`, then: `chmod +x myfm.sh && ./myfm.sh`

Thank you

Now go open your terminal and try everything!