# CIS 106 – Final Exam Study Notes

## 1. How to Clone a GitHub Repository

**Definition:**
Cloning a GitHub repository means creating a local copy of a remote repository so you can work on the files from your own computer.

To clone a repository, you use the `git clone` command followed by the repository URL.

**Examples:**

```
git clone https://github.com/username/project.git
git clone https://github.com/classroom/final-exam.git
```

## 2. How to Use Git Commands

**Definition:**
Git is a version control system used to track changes, save progress, and collaborate with others.

**Common Git Commands:**

- `git status` – shows the current state of the repository
- `git add` – stages files for commit
- `git commit` – saves changes locally
- `git push` – uploads changes to GitHub
- `git pull` – downloads changes from GitHub

**Examples:**

```
git status
git add finalNotes.md
git commit -m "Update final exam notes"
git push
```

## 3. How to Write a Markdown File (Images & Formatting)

**Definition:**
Markdown is a simple text formatting language used to create structured documents such as README files.

Markdown supports headings, bold text, lists, and images.

**Examples:**

```
# Main Title
## Section Title

**Bold text**
*Italic text*

- Item one
- Item two

![Server Diagram](diagram.png)
```

---

## 4. How to Convert a Markdown File to PDF

**Definition:**
Converting a Markdown file to PDF allows you to submit or share the document in a fixed format.

This is commonly done using the `pandoc` tool.

**Examples:**

```
pandoc finalNotes.md -o finalNotes.pdf
pandoc README.md -o README.pdf
```

---

## 5. How to Compress (Zip) a Directory in Debian

**Definition:**
Compressing a directory means packaging multiple files into a single archive file.

**Examples:**

```
zip -r FinalExamStudyNotes.zip FinalExamStudyNotes/
zip -r backup.zip Documents/
```

---

## 6. Absolute Paths and Relative Paths

An **absolute path** is the complete path to a file or directory, starting from the root /. It always points to the same location, no matter where you are currently in the file system.

- **Example:**
  - /home/usr/Documents/file.txt

A **relative path** shows how to reach a file or directory from your current location. It does not start with the /, and changes meaning depending on where you are.

- **Example:**
  - If you are in current directory is /home/usr, and you want to access Documents/file.txt, the relative path would be:
    - Documents/file.txt

## 7. How to Work with Manual Pages (man Command)

**Definition:**
Manual pages provide official documentation for Linux commands.

**Examples:**

```
man ls
man grep
```

Inside a manual page:

- Type `/word` to search
- Press `n` for next match
- Press `q` to quit

---

## 8. How to Search for Specific Words in a Manual Page

**Definition:**
You can search inside a manual page to quickly find options or explanations.

**Examples:**

```
man cp
```

Then type:

```
/recursive
```

---

## 9. How to Redirect Output (>, >>, |)

**Definition:**
Output redirection allows you to control where command output goes.

- `>` sends output to a file (overwrites)
- `>>` appends output to a file
- `|` sends output to another command

**Examples:**

```
ls > files.txt
ls >> files.txt
ls | less
```

---

## 10. How to Append the Output of a Command to a File

Append means to add more to a file instead of overwriting its content. When we use > on a file that already exist and contains data, we overwrite whatever is already inside the file. For example:

- `ls -la > allmyfiles.lst`

In this example, if the file allmyfiles.lst had any data prior executing the command, that data will be overwritten by the output of ls-la. But, what happens when we don't want to keep the data? Then we use:

- `ls -la >> allmyfiles.lst`

Will add the output of ls-la to the end of the file `allmyfiles.lst`

## 11. Redirecting Output from One Command to Another (Pipes)

*The **pipe ( | )** allows you to redirect the standard output of a command to the standard input of another.

- Usage:
  - `command_1 | command_2 | command_3 | .... | command_N`
- Basic examples:
  - Use the grep to look for a string in a particular man page
    - `man ls | grep "human-readable"`
  - Display only the options of the of any command from its man page
    - `man ls | grep "[[:space:]]*[[:punct:]]"`

Other examples of |:

- Display only the ip addresses from the output of the ip command:
  - `ip addr | grep -Eo '[[:digit:]]{1,3}\.[[:digit:]]{1,3}\.[[:digit:]]{1,3}\.[[:digit:]]{1,3}'`
- Display only the 2nd line in a file:
  - `head -2 file.lst | tail -1`
- Parse a file with grep and replace a string in the output:
  - `grep -i "honda" cars.csv | sed 's/honda/tesla/g'`

## 12. Using echo and Output Redirection to Create a File

**Definition:**
The `echo` command prints text, which can be redirected into a file.

**Examples:**

```
echo "Hello World" > hello.txt
echo "My notes for CIS 106" > notes.txt
```

---

# 13. Using Wildcards

**Definition:**
Wildcards allow you to work with multiple files at once.

- `*` matches any number of characters
- `?` matches one character

## What is a Wildcard?

A wildcard is a symbol used to replace or represent one or more characters in a file name.

There a are 3 wildcards: *, ? and [set].

## * wildcard

The asterisk (*) matches **zero or more characters** in a filename.

- Examples:
  - List all the files in a given directory:
    - `ls Downloads/*`
  - List all text files in a given directory:
    - `ls Downloads/*.txt`
  - List all the text files in a given directory that start with the letter f
    - `ls Downloads/f*.txt`

The **? wildcard** metacharacter matches precisely one character. This wildcard is very useful when working with hidden files (also called dot files).

- Examples:
  - List all the hidden files in the current working directory
    - `ls ./.??*`
  - List all hidden files in the parent directory
    - `ls ../. ??*`
  - List all the files that have 2 characters in the file name between letters b and k
    - `ls b??k*`
  - List all the files that have a single character between letters f and l.
    - `ls f?l*`
  - List all the files with a 2 letter file extension
    - `ls *.??`

- [ ] wildcard

The brackets wildcard matches a single character in a range. The brackets wildcard use the exclamation mark to reverse the match.

- Examples:
  - To match all files that have a vowel after letter f:
    - `ls f[aeiou]`
  - To match all files that do not have a vowel after letter f:
    - `ls f[!aeiou]*`
  - To match all files that have a range of letters after f:
    - `ls f[a-z]`

## 14. Using Brace Expansion

**Brace expansion** is a feature of the bash shell that generates argument strings. Those strings can be used by commands to operate on files. This feature does not make calls to the operating system like wildcards, they just generate file names based on a given pattern.

- **How to use**:
  - Start with an open brace
  - With no spaces, type your string separating entries by command
  - Close the brace
    - Example:
      - `mkdir -pv example_site{assets/large,docs/share,scripts/js}`
- More examples:
  - Create 3 different files with the same name but different file extensions
    - `touch file.{md,txt,rtf}`
  - Create 10 files in a range from 0 to 9:
    - `touch file {0..9}.txt`
  - Remove specific files that start with a given keyword
    - `rm image_*{01..08}*camera.{png,jpg}`
  - Create an entire directory tree in a single command (1 level deep)
    - `mkdir -pv project_saturn/{code,source,database}/new`
  - Create an entire directory tree in a single command (2 levels deep)
    - mkdir -pv
      - project_jupiter/site{old,new}/{code/{scripts/markup},assets/{imgs,mp3,mp4}}

## 15. Creating a Simple Hello World Shell Script

**Definition:**
A shell script is a file that contains a sequence of Linux commands.

**Examples:**

```
#!/bin/bash
echo "Hello World"
```

Make it executable:

```
chmod +x hello.sh
```

---

## 16. Using Variables in a Shell Script

**Definition:**
Variables store values that can be reused inside a script.

**Examples:**

```
name="Student"
echo "Hello $name"
```

---

# 17. Commands

---

## awk

*The **awk** command is a scripting language used for processing and displaying text. Awk can work with a text file from standard output. There are several implementations of Awk, like: nawk, mawk, gawk, and busybox. It performs operations line by line.

- Usage:
    - `awk + options + {awk command} + file + file to save (optional)`
- Basic example:
    - Print the first column of every line of a file:
        - `awk '{print $1} ~/Documents/Csv/cars.csv'`
- Awk Variables:

 AwkVar

- Examples on awk:
    - Print first field of /etc/passwd file
        - `awk -F: '{print $1}' /etc/passwd`
    - Print the last field of the /etc/passwd file
        - `awk -F: '{print $NF}' /etc/passwd`
    - Print the first and last field of the /etc/passwd
        - `awk -F: '{print $1, " = ", $NF}' /etc/passwd`
    - Print the first 3 and 3 field with line numbers

        - `awl -F: '{print NR,$1,$3}' /etc/passwd`
    - Print the first and 4th field with a different field separator
        - `awk -F: '{OFS="="}{print $1,$4}' /etc/passwd`
    - Start printing a file from a given line (exclude the first 2 lines)
        - `awk 'NR > 3 { print }' /etc/passwd`

## cat

The **cat** command is used to display the content of a file. Cat is short for concatenate which is the command's intended use.

- Usage:
    - `cat + option + file(s) to display`
- Examples:
    - Display the content of a file located in ~/Documents/sample_files/:
        - `cat ~/Documents/sample_files/Code/helloworld.py`
    - Display the content of a file with line numbers:
        - `cat -n ~/Documents/sample_files/Code/helloworld.py`
    - Display the content of a file including non printing characters and line endings:
        - `cat -A ~/Documents/sample_files/Code/helloworld.py`

## cp

The **cp** command copies files/directories from a source to a destination. It uses the same structure as the mv command: - Formula: - `cp + files to copy + destination` - To copy directories you use the `-r` option: - `cp -r + directory to copy + destination` - Examples: - To copy a file: - `cp Downloads/games.zip Rockstar/` - To copy a directory with absolute path: - `cp -r ~/Downloads/games ~/Rockstar/` - To copy the content of a directory to another directory: - `cp Downloads/games/* ~/Rockstar`

## cut

The **cut** command is used to extract a specific section of each line of a file and display it to the screen.

- Usage:
    - `cut + option + file(s)`
- Example:
    - Display a list of all the users in your system
        - `cut -d ':' -f1 /etc/passwd`
    - Display a list of all the users in your system with their login shell
        - `cut -d ':' -f1,7 /etc/passwd`

## grep

The **grep** command is used to search text in a given file. Grep works by line basis (it matches the search criteria in a line by line basis).

- Usage:
    - `grep + option + search criteria + file(s)`
- Basic Example:
    - Search any line that contains the word 'Game' in the given file:
        - `grep 'Game' ~/Documents/GameofThrones.txt`
    - Search any line that contains the word 'thrones' regardless of the case.
        - `grep -i 'Thrones' ~/Documents/Books/GameofThrones.txt`
    - Display how many lines contain the matched string:
        - `grep -c 'Game' ~/Documents/Books/GameofThrones.txt`
- More examples of `grep`
    - Search any line that contains the word dracula regardless of case and with number line:
        - `grep -in 'dracula' ~/Documents/Books/dracula.txt`
    - Search for all the lines that do not contain the word 'war':
        - `grep -v 'war' ~/Documents/Books/war-and-peace.txt`
    - Search and display only matched string (pattern)
        - `grep -o 'pride ~/Documents/Books/Pride&Prejudice'`
    - Display a list of users with the /bin/bash login shell:
        - `grep -i "/bin/bash" /etc/passwd`
    - Display our user's information as stored in the /etc/passwd
        - `grep -i $USER /etc/passwd`

## head

he **head** command displays the top N number of lines of a given line. By default, it prints the first 10 lines. If more than one file name is provided then data from each file is preceded by its file name.

- Usage:
    - `head + option + file(s)`
- Examples:
    - Display the first 10 lines of a file:
        - `head ~/Documents/Book/AsongOficeandFire.txt`
    - Display the first 5 lines of a file:
        - `head -5 ~/Documents/Book/TheHobbit.txt`
- More examples:
    - Display the first 5 lines of multiple files:
        - `head -5 ~/Documents/Book/{TheHoobit,GameofThrones}.txt`
    - Display the first line of multiple files using wildcards
        - `head -n 1 Csv/*.csv Code/*.py`

- Display the name of a file in the output
  - `head -v -n 7 Json/joke.json`
- Display a given number of lines of the output of a given command
  - `ls -l ~/cis106/ | head -n 2`
- Display a given number of bytes instead of lines:
  - `head -c 50 Txt/TheHobbit.txt`

## ls

The **ls** command is used for listing the content of a given directory of the file/directory itself. It has a lot of characteristics:

- ls can be used with or without arguments.
- Using ls without a file or directory lists the content of the present working directory.
- ls has a lot of options. Which can be seen with man ls

***Formula*:**

- ls + option + directory to list

## man

**Definition:** Displays manual documentation.

**Examples:**

```
man ls
man awk
```

---

## mkdir

- The **mkdir** command is a used to create single or multiple directories. Directories can be made in the present working directory or in a different directory by using either absolute or relative path. In case you create an existing directory an error notifying its existence will pop up.
  - Formula:
    - `mkdir + name of directory`
  - Examples:
    - Creating in present directory:
      - `mkdir breathingStyles`
    - Creating in a different directory using relative path:
      - `mkdir BreathingStyles/WaterBreathing`
    - Creating in a different directory using absolute path:
      - `mkdir ~/BreathingStyles/FlameBreathing`

## mv

- The **mv command** moves and renames directories. The source of the formula can use both absolute or relative path.
  - The basic Formula:
    - `mv + source + destination`
    - In the basic formula source is the file or directory that you want to move and destination is where the directory or file is going.
    - For renaming files/directories the formula stays the same, only the value of the destination changes and it is now the brand new name of the file.
  - Examples:
    - To move a file from a directory to another using relative path:
      - `mv Downloads/text.pdf Documents/`
    - To move a directory from one directory to another using absolute path:
      - `sudo mv ~/Downloads/topic /usr/share/themes`
    - To move a file from one directory to another combining absolute path and relative path.
      - `mv Downloads/ironman.png /media/movies/flashdrive`
    - To move multiple directories/files to a different directory
      - `mv food/ cars/ podcasts/ /restaurants/dreams/audio/`
  - To **rename files**
    - Rename a file
      - `mv games.txt rockstargames.txt`
    - To rename a file using absolute path
      - `mv ~/Downloads/games.txt ~/Downloads/rockstargames.txt`
    - To move and rename a file in the same command:
      - `mv Downloads/rockstargames.txt Documents/rdrockstargames.txt`

## tac

The **tac** command is used for displaying the content of a file in reverse order. Just like cat, tac concatenates files and displays the output of the concatenation.

- Usage:
  - `tac + option + file(s) to display`
- Examples:
  - Display the content of a file located in ~/Documents/sample_files/ in reverse order
    - `tac ~/Documents/sample_files/Code/helloworld.py`
  - Display the content of multiple files in reverse order
    - `tac ~/Documents/sample_files/Code/helloworld.py tac ~/Documents/sample_files/Code/helloworld.py`

## tail

*The **tail** command displays the last N number of lines of a given file. By default, it prints the last 10 lines. If more than one file is provided then data from each file is preceded by its file name.

- Usage:
    - `tail + option + file(s)`
- Examples:
    - Display the last 10 lines of a file
        - `tail ~/Documents/sample_files/`
    - Display the last five lines of a file
        - `tail -5 ~/Documents/sample_files/`
- More examples:
    - In this section the examples are the same as head, just change the command.

## touch

- The **touch** command is used for creating files. However, the touch's command designed purpose is not to create files, but to update any given file's timestamp. However, if the file does not exist, it creates it.
    - Formula:
        - `touch + name of file`
    - Examples:
        - To create a single file:
            - `touch food`
        - To create many files:
            - `Groceries.txt notes.csv instructions.py`
        - To create files using relative path:
            - `touch Documents/explanations.txt`
        - To create files using absolute path:
            - `touch ~/Documents/explanations2.txt`

## tr

**Definition:** Translates or removes characters.

**Examples:**

```
tr a-z A-Z < notes.txt
tr -d '0-9' < notes.txt
```

---

## tree

**Definition:** Displays directory structure in tree format.

**Examples:**

```
tree
tree Project
```

---