Intro to Unix Command Line

You can also do the activities below via the clab app. Enter lab0. The home folder within the corresponding activity has been updated with an extra unix folder. Enter that folder. Most of the commands below assume you are inside that folder. But if the explanation below requires you to navigate to another folder, please do so.

In addition to the above, we provide a tar file containing the ‘Unix’ folder if you want to navigate it on your computer (Linux/Mac). But clab is highly recommended (as there is a possibility of inconsistency, especially with Mac and non-Debian Linux distros, so we stick to the GNU/Debian(Ubuntu) Linux for this course). Download the file and untar it (tar -xvf filename; replace filename with the name of whatever you downloaded). It should create a folder called ‘Unix.’ Enter that folder.

clear

One of the most basic commands. It just clears the terminal screen. Doesn't delete history or affect running programs. It only affects the visual display.

man

This command provides the documentation for almost all of the commands. They display the manual/help pages. They are the reference you would want to look at when you are unsure about something.

```
man <command name>
Eg:
        man ls
        man wc
        man man
```

pwd

pwd (present working directory) command tells your current working directory, in the terminal. (A directory is similar to a folder.) It displays the full path of the directory you are in.

```
$ pwd
/home/labdirectory/unix
```

ls

The ls command ‘lists’ the contents of a directory. You have various options for this command. Some notable ones. Try them as ls -`OPTIONS` <path, considers the pwd if empty>:
`l` - lists in a long format: gives additional information about the files and folders, such as owner, permissions, size, date modified, and other information.
`a`: display all files including the hidden files. The hidden files start with a dot in front of them
`R`: recursive listing of the subdirectories
`h`: Used with `l`, gives out the size in a human readable format (KB, MB)
`t`: Sorts the output by the time of last modification, with the newest files first
`S`: Sorts by size, largest first
`X`: Sorts by extension, with extensions sorted alphabetically

cd

‘Change Directory’: changes the working dir (pwd), no options. Supports absolute and relative paths.
cd <path, absolute/relative>
`cd`: Changes the directory to the home directory
`cd -`: Changes the directory to the previous directory (OLDPWD)

Paths:

- `.`: Current directory
- `..`: Parent directory
- `~`: Home directory
- `/`: Root directory

```
cd code
cd ../photos
cd -
cd ..
```

mkdir

Creates new directories. Created by using abs/rel paths.
Options:
`p`: creates the directory only if it doesn’t exist, makes parent directories as needed
`v`: The verbose output displays a message for each directory that is created

```
mkdir hello
mkdir -p nested/dir
mkdir -pv nest/folder
```

rmdir

Removes empty directories from the file system.
`p`: removes the specified directory and its parent directories if they are empty.

```
rmdir hello
rmdir -p nested/dir
rmdir -pv nest/folder
```

Use rmdir when you want to ensure that only empty directories are deleted.

cp

Copies files and directories from one location to another.
Syntax: cp [options] <source(s)> <destination>

When *multiple files or directories are given as a source*, the destination must be a directory.
If the destination file already exists, cp will overwrite it without warning.

`i`: Prompts before overwriting an existing file
`r`: recursive copy - an entire directory and its contents, including subdirectories
`v`: verbose mode - Displays the files being copied, useful for tracking the operation

mv

Moves/renames files and directories.

Syntax: mv [options] <source(s)> <destination>

If you have one item in sources and if the source and the destination's directory are the same with just a name change, it is equivalent to renaming.

`i` (Interactive): Prompts for confirmation before overwriting an existing file
`f` (Force): Forces the move operation without prompting for confirmation, even if it involves overwriting files
`u` (Update): Moves the source file only if it is newer than the destination file or if the destination file does not exist
`v` (Verbose): Provides detailed information about the files being moved, including the source and destination paths
`n` (No Clobber): Prevents overwriting of existing files. If a destination file exists, the move is not performed.

Note: If you want to move or remove something by mistake, do it carefully, as it is difficult to recover them. If you want to explore, copy the directory and then try.

rm

Removes files and directories. Be very careful with this command.
Syntax: rm [options] <items>
`f` (Force): Forces the removal of files without prompting for confirmation
`i` (Interactive): Prompts for confirmation before deleting each file
`r` or `R` (Recursive): Deletes directories and their contents recursively (deleting non-empty directories)
`v` (Verbose): Displays detailed information about the files being deleted
`d` (Directory): Removes empty directories.

`rmdir` and `rm -d` are more or less functionally similar

echo

The echo command displays a line of text or string to the standard output. The -n option omits the trailing newline. The -e option enables the interpretation of backslash-escaped characters such as \n for newlines and \t for tabs. The -E option disables this interpretation (this is the default behavior).

```
echo "Hello World"
echo -e "Hello\nWorld"
echo -e "Tab\tSeparated\tText"
echo -n "Hello, world!"
Echo -E "Hello\nWorld"
```

touch

The touch command is typically used to create an empty file or update file timestamps; the -a option updates only the access timestamp, and -m updates only the modification timestamp.

To follow this demo, please navigate to the 'code' directory.

```
stat Tree.cpp                    (stat is used to display the status of a file)
touch -a Tree.cpp
stat Tree.cpp
touch -m Tree.cpp
stat Tree.cpp
touch -t 202411070900.30 filename.txt
stat filename.txt
stat Tree.cpp
touch -t 202411070900.30 Tree.cpp
stat Tree.cpp
touch -c file                    (no file created)
```

cat

The cat command concatenates and displays file contents, with -v showing non-printable characters as escape sequences (excluding tabs and newlines), -n numbering all output lines, including blank ones, and -s suppressing repeated empty lines to output a single blank line for consecutive empty lines.

> *To follow this demo, please navigate to the 'unix' directory. Upon running the ls command, you should see files joke1, joke2, and joke3.*

```
cat joke1
cat joke1 joke2 joke3
cat joke*

cat -v example.txt
```

Numbering and Deleting extra lines

```
cat -n /etc/lsb-release
cat -s joke1
cat joke1
```

less

The less command lets you view file contents one screen at a time, with -i making searches case-insensitive unless uppercase letters are used in the search term and -N displaying line numbers in the left margin.

> *To follow along with this demo, please navigate to the 'file-analysis' directory.*

```
less bigfile            (then, search for the word 'wolf')
less -i bigfile         (then, search for the word 'wolf'. Is there a difference?)
less -N bigfile
```

more

The more command also allows viewing files page by page, with +[number] starting at a specified line number and -[number] displaying a set number of lines per page.

> *To follow along with this demo, please navigate to the 'file-analysis' directory.*

```
more bigfile
more +19 bigfile
more -5 bigfile
```

head

The head command displays the first few lines of a file, and the -n option specifies exactly how many lines to display from the beginning, defaulting to 10 if not specified.

To follow along with this demo, please navigate to the ‘file-analysis’ directory.

```
head bigfile
head -n 5 bigfile; head -5 bigfile
head -n 2 bigfile smallfile
```

tail

The tail command displays the last few lines of a file, and the -n option specifies exactly how many lines to display from the end, defaulting to 10 if not specified.

To follow along with this demo, please navigate to the ‘file-analysis’ directory.

```
tail bigfile
tail -n 5 bigfile; tail -5 bigfile
tail -n 2 bigfile smallfile
```