Linux

This document contains some brief notes on a small set of linux CLI commands.

Contents and Traversals

Current Location

```
$ pwd

# Example Output
$ pwd
/home/worker
```

> This command will print the path to the current working directory, its where you currently are in the filesystem

Directory Contents

```
# Current directory
$ ls

# Some other directory
$ ls <path/to/directory>
```

> This command will print the contents of the current directory or some other directory we specify the path to. It will list files, and directories within. We can utilize flags to change its behavior if we would like.

Directory Traversals

```
# Change Directory
$ cd <directory>

# Go to the home directory
$ cd ~

# Go to the parent directory
$ cd ..

# Go to a subdirectory
$ cd /etc/nginx/
```

> The cd (change directory) command will move us to the directory that we specify, there are some shortcuts described in the examples above.

```
# pushd
$ pushd .

# popd
$ popd
```

> Push and Pop Directory are a nice pair of tools that we can use to "Save a directory" using pushd and later go back to it (the last one "saved") using popd. These are not necessary for the tasks described.

Files and Directories

File Continents

```
# Print contents of a file
$ cat <file>

# Examine contents (Scroll)
$ more <file>

# Examine contents (Scroll)
$ less <file>
```

> These commands allow us to examine the contents of a file, but they do not allow us to modify them. We can use a text editor as described later down.

Directory Creation

```
# Create directory
$ mkdir -p <path/to/dir>

# Example
$ mkdir -p /etc/home/manager/proxy-conf/http
```

> This command (with the **-p** flag) will make a directory and all directories in the path if they are not already created.

File Creation

```
# Without a text editor
$ touch <file_name>
# Example
$ touch file.txt
```

> This command will create a file if it does not already exist, otherwise it will update the last modified time of the file.

Directory Removal

```
$ rm -r <directory>
```

File Removal

```
$ rm <file>
```

Copy a Directory

```
$ cp -r <directory> <path/to/new/location/new_name>

# Examples
$ cp -r dir /home/manager/new/dir_cpy
$ cp -r dir /home/manager/new/dir
$ cp -r dir dir_cpy
```

> We can copy a directory using the copy command with the **-r** flag, the new location can be to a different directory with the same name, a different directory with a different name, or the same directory with a different name. When copying the original directory will still exist.

Copy a file

```
$ cp <file>
# Examples
$ cp file /home/manager/new/file
$ cp file /home/manager/new_file
$ cp file new_file
```

> We can copy a file using the copy command, the new location can be to a different directory with the same name, a different directory with a different name, or the same directory with a different name. When copying the original file will still exist.

Move a Directory

```
$ mv -r dir <path/to/new/location/new_name>

# Examples
```

```
$ mv -r dir /home/manager/new/dir_cpy
$ mv -r dir /home/manager/new/dir
```

> We can move a directory to a new directory, with a name that is new or the same as the one we are copying **Move a File**

```
$ mv file  <path/to/new/location/new_name>
# Examples
$ mv file /home/manager/new/file_cpy
$ mv file /home/manager/new/file
```

> We can move a file to a new directory, with a name that is new or the same as the one we are copying

Rename a File/Directory

```
$ mv old_object new_name
```

> When using the mv command to the same directory, it will rename the file/directory.

Editing a File

```
# Nano
$ nano <file>
# Emacs
$ emacs <file>
# Vim
$ vim <file>
```

> We can use a text editor to modify the contents of a file, they each have their own following and related commands. The easiest to use would be Nano, so its use is suggested if you are unfamiliar with other text editors. (You will need to install emacs if you wish to use it)

Processes

```
ps
```

> List the processes that are currently running (created by the user)

System Processes

These commands require **sudo** privileges

Sudo

```
$ sudo <command>
```

Start a process

```
$ sudo systemctl start <service>

# Example
$ sudo systemctl start nginx
```

> This command will start the service, if the system shuts down or restarts it will not continue to work.

Enable a Process

```
$ sudo systemctl enable <service>

# Example
$ sudo systemctl enable nginx
```

> This will enable the service, it does not start it (unless a flag is used). This makes it so the service will be started when the system starts up.

Stop a process

```
$ sudo systemctl stop <service>

# Example
$ sudo systemctl stop nginx
```

> This command will stop the service, if the service is **enabled** as described above it will start on system startup - it would also need to be disabled.

Disable a process

```
$ sudo systemctl disable <service>

# Example
$ sudo systemctl disable nginx
```

> This will disable the service, it does not stop it (unless a flag is used). This makes it so the service will not be started when the system starts up.

Service Status

```
$ sudo systemctl status <service>

# Example
$ sudo systemctl status nginx
```

> This command will output the current status, along with some information as to why it may have failed - with a command you can run for more information (This will be some journalctl -u ... command)

Package Management

The system you are given is a Debian based system, so it uses the **apt** package manager

Install Package

```
$ sudo apt install <package> -y

# Example
$ sudo apt install emacs -y
```

> This will install the package and all of its dependencies the **-y** flag preempts the "Would you like to install this" prompt with a yes.

Remove Package

```
$ sudo apt remove <package> -y

# Example
$ sudo apt remove emacs -y
```

Switching Users

```
# Switch to the specified user
su <user>

# Switch to the root account
sudo su
```

> The su command allows us to switch users if we know their password. If we use the sudo command, we can switch to any user even the root account while only having to use our own password.

Remote Connections

```
$ ssh user@remote_ip
```

> There are additional flags that can be used, and possibly additional requirements.

Additional References

- Linux: https://cheatography.com/davechild/cheat-sheets/linux-command-line/
- Nano: https://www.nano-editor.org/dist/latest/cheatsheet.html
- Emacs: https://www.gnu.org/software/emacs/refcards/pdf/refcard.pdf
- Vim: https://vimsheet.com/