Written by: Dr. Farkhana Muchtar

LAB 2: BASIC LINUX COMMANDS

Introduction of Linux OS

These exercises aim to acquaint students with the Linux command-line environment and introduce them to some essential Linux commands, with a focus on those specific to Ubuntu or Debian distributions.

1 Simple Linux Command

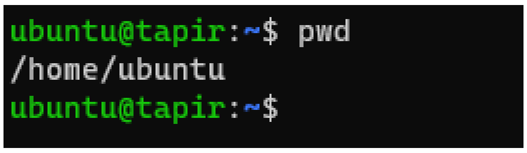
In this lab exercise, you will learn and practice essential Linux commands. Follow the step-by-step instructions to complete each task.

Task 1: Navigating directories

1. Access Linux instance by using multipass shell.

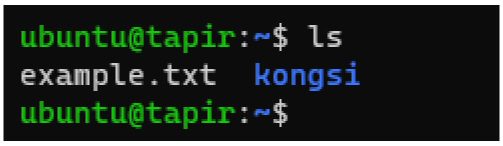
```
multipass shell <instance-name>
```

2. Use the `pwd` command to display the current working directory.



```
ubuntu@tapir:~$ pwd
/home/ubuntu
ubuntu@tapir:~$
```

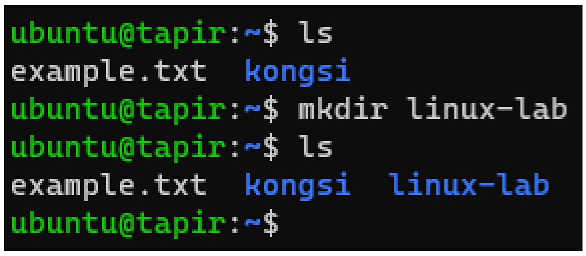
3. Use `ls` to list down contents of current directory. In your case it might be an empty folder.



```
ubuntu@tapir:~$ ls
example.txt  kongsi
ubuntu@tapir:~$
```

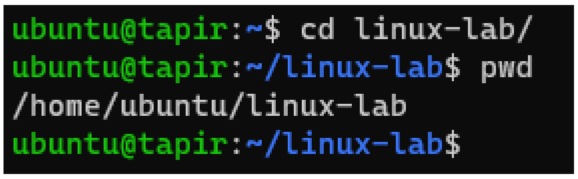
4. Create a new directory called `linux-lab` using the `mkdir` command:

```
mkdir linux-lab
```



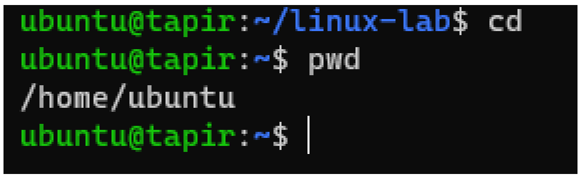
```
ubuntu@tapir:~$ ls
example.txt  kongsi
ubuntu@tapir:~$ mkdir linux-lab
ubuntu@tapir:~$ ls
example.txt  kongsi  linux-lab
ubuntu@tapir:~$
```

5. Change the current working directory to `linux-lab` using `cd linux-lab` command and using `pwd` to display current working directory.



```
ubuntu@tapir:~$ cd linux-lab/
ubuntu@tapir:~/linux-lab$ pwd
/home/ubuntu/linux-lab
ubuntu@tapir:~/linux-lab$
```

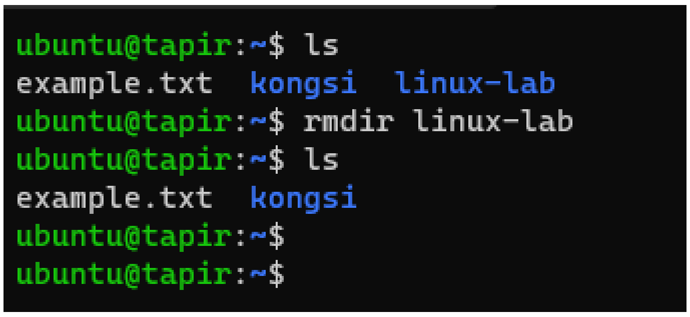
6. Use the `cd` command to navigate to your home directory.



```
ubuntu@tapir:~/linux-lab$ cd
ubuntu@tapir:~$ pwd
/home/ubuntu
ubuntu@tapir:~$
```

7. Run `rmdir <empty_directory_name>` to delete empty directory. For this tutorial, run `rmdir linux-lab`.

   * Remake `linux-lab` folder again by typing `mkdir linux-dir` for next exercise.



```
ubuntu@tapir:~$ ls
example.txt  kongsi  linux-lab
ubuntu@tapir:~$ rmdir linux-lab
ubuntu@tapir:~$ ls
example.txt  kongsi
ubuntu@tapir:~$
ubuntu@tapir:~$
```

Task 2: Working with files

1. Create an empty file called `file1.txt` using the `touch` command:

```
touch file1.txt
```

2. Use the `ls` command to list the contents of the current directory.
3. Rename `file1.txt` to `file2.txt` using the `mv` command:

```
mv file1.txt file2.txt
```

4. Use the `ls` command again to verify the file has been renamed.
5. Make a copy of `file2.txt` and name it `file3.txt` using the `cp` command:

```
cp file2.txt file3.txt
```

6. Use the `ls` command to verify the new file `file3.txt` has been created.

Task 3: Editing files with Nano

1. Open `file2.txt` in the Nano text editor:

   ```
   nano file2.txt
   ```

2. Write some text in the file, save your changes (`^O`, or Ctrl + O), and exit Nano (`^X`, or Ctrl + X).
3. Use the `cat` command to display the contents of `file2.txt`:

   ```
   cat file2.txt
   ```

Task 4: Working with file permissions

1. Use the `ls -l` command to display the file permissions of `file2.txt`.
2. Change the permissions of `file2.txt` to remove write access for the owner using the `chmod` command:

   ```
   chmod u-w file2.txt
   ```

3. Use the `ls -l` command to verify the permissions have been updated.

Task 5: Searching for files and content

1. Use the `grep` command to search for a specific word in `file2.txt`:

   ```
   grep 'word-to-search' file2.txt
   ```

Replace **'word-to-search'** with a word you wrote in `file2.txt` during Task 3.

2. Use the `find` command to search for a file named `file3.txt` in the current directory and its subdirectories:

   ```
   find . -name file3.txt
   ```

Task 6: Cleanup

1. Delete the `file3.txt` file using the `rm` command:

   ```
   rm file3.txt
   ```

2. Use the `ls` command to verify the file has been removed.
3. Navigate back to your home directory using the `cd` command.
4. Remove the `linux-lab` directory and its contents using the `rm` command with the `-r` option:

   ```
   rm -r linux-lab
   ```

5. Verify the directory has been removed using the `ls` command.

SECR2043 - OPERATING SYSTEMS (Lab Exercise)

Written by: Dr. Farkhana Muchtar

Upon completion of this lab exercise, you should have a better understanding of basic Linux commands and how to perform common tasks in the Linux command line environment.

2 Managing packages/software in Ubuntu

2.1 Upgrading existing packages

- Updating package/software status

  ```
  sudo apt update
  ```

- List down upgradable package/software

  ```
  apt list –upgradable
  ```

- Upgrade those packages/software

  ```
  sudo apt upgrade
  ```

2.2 Installing software package

We also can use the same command which is **apt** to install new application either from repositories or from downloaded files. The command is:

```
sudo apt install <package/software name>
```

The **build-essential** package in Ubuntu is a metapackage that contains essential tools and libraries required for compiling and building software from source code. Installing build-essential ensures that you have the necessary tools for basic software development and for compiling software on your system.

The package includes the following components:

- **`gcc`:** The GNU Compiler Collection, which contains compilers for C, C++, Objective-C, Fortran, Ada, and others.
- **`g++`:** The GNU C++ compiler, a part of the GNU Compiler Collection.
- **`make`:** A utility that automates the process of building executable programs and libraries from source code.
- **`libc6-dev`:** The C standard library, which provides essential functions for C programs.
- **`dpkg-dev`:** Development tools for creating, building, and managing Debian packages.

This build-essential metapackage is prerequisite for next exercise, which is writing, compiling and running C and C++ programs.

To install build-essential metapackage, run this command:

```
sudo apt install build-essential
```

3 Using Nano Text Editor

Nano is a simple, easy-to-use text editor for Unix-like operating systems, including Linux. It is designed to be a user-friendly alternative to other command-line text editors, such as Vi or Emacs. Nano is especially useful for beginners who are not yet familiar with the more complex features and commands of other text editors.

Now that you have a brief introduction to Nano, please refer to the tutorial provided in the previous response to learn how to use Nano for editing text files in the Linux terminal.

3.1 Launching Nano

To open a new or existing file with Nano, use the following command in the terminal:

```
nano filename.txt
```

Replace filename.txt with the name of the file you want to create or edit. If the file does not exist, Nano will create a new file with the specified name.

3.2 Basic Editing

Once you have opened a file, you can start editing it. Use the arrow keys to navigate through the text. You can insert, delete, and modify the text just like any other text editor.

3.3 Keyboard Shortcuts

Nano has several keyboard shortcuts for common tasks. Note that the ^ symbol represents the `Ctrl` key:

- `^X`: Exit Nano. If you have unsaved changes, Nano will prompt you to save them before exiting.
- `^O`: Save the current file. You will be prompted for the file name if it's a new file.
- `^W`: Search for a specific text within the file. Enter the text and press `Enter` to start the search.
- `^K`: Cut the current line. This is useful for moving or deleting lines of text.
- `^U`: Paste the previously cut text at the current cursor position.

- `^C`: Display the current line number and column position of the cursor.
- `^_`: Go to a specific line number. Enter the line number and press `Enter` to move the cursor to that line.

4 Writing and Compiling Simple C/C++ Code

4.1 Hello World Example in C

1. Create a new folder called **workspace** in your home `(/home/ubuntu)`

```
mkdir workspace
```

2. Go inside new workspace folder

```
cd workspace
```

3. Create a new file called hello.c and edit using nano editor

```
nano hello_world.c
```

4. In the Nano text editor, type the following C code:

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

5. Save the file and exit Nano

   To save your C program, press `^O` (Ctrl + O). Nano will prompt you to confirm the file name, which should be `hello_world.c`. Press `Enter` to save the file. Next, exit Nano by pressing `^X` (Ctrl + X).

6. Compile the C program

   Now that your C program is saved and GCC is installed, compile the program using the following command:

```
gcc hello_world.c -o hello_world
```

7. Run the compiled program

   Execute the compiled program by entering the following command:

```
./hello_world
```

   Your terminal should display the output:

```
Hello, World!
```

4.2 Hello World Example in C++

1. Make sure you are inside workspace folder. Use **pwd** command to check the current location. If not, just using **cd** command to go inside folder.

   ```
   cd workspace
   ```

2. Create a new file called hello.c and edit using nano editor

   ```
   nano hello_world.cpp
   ```

3. In the Nano text editor, type the following C code:

   ```
   #include <iostream>

   Using namespace std;

   int main() {

       cout << “Hello World!” << endl;

   }
   ```

4. Save the file and exit Nano

   To save your C++ program, press `^O` (Ctrl + O). Nano will prompt you to confirm the file name, which should be `hello_world.cpp`. Press `Enter` to save the file. Next, exit Nano by pressing `^X` (Ctrl + X).

5. Compile the C++ program

   Now that your C++ program is saved and GCC is installed, compile the program using the following command:

   ```
   g++ hello_world.cpp -o hello_world2
   ```

6. Run the compiled program

   Execute the compiled program by entering the following command:

   ```
   ./hello_world2
   ```

   Your terminal should display the output:

   ```
   Hello, World!
   ```

4.3 Linux Kernel Info in C

1. Make sure you are inside workspace folder. Use **pwd** command to check the current location. If not, just using **cd** command to go inside folder.

   ```
   cd workspace
   ```

2. Create a new file called hello.c and edit using nano editor

   ```
   nano kernel_info.c
   ```

3. In the Nano text editor, type the following C code:

```
#include <stdio.h>
#include <sys/utsname.h>
#include <stdlib.h>

int main() {
    struct utsname kernel_info;

    // Get kernel information
    if (uname(&kernel_info) == -1) {
        perror("uname");
        exit(EXIT_FAILURE);
    }

    // Display the kernel information
    printf("System name: %s\n", kernel_info.sysname);
    printf("Node name: %s\n", kernel_info.nodename);
    printf("Kernel release: %s\n", kernel_info.release);
    printf("Kernel version: %s\n", kernel_info.version);
    printf("Machine hardware: %s\n", kernel_info.machine);

    return 0;
}
```

4. Save the file and exit Nano

   To save your C program, press `^O` (Ctrl + O). Nano will prompt you to confirm the file name, which should be `kernel_info.c`. Press `Enter` to save the file. Next, exit Nano by pressing `^X` (Ctrl + X).

5. Compile the C program

   Now that your C program is saved and GCC is installed, compile the program using the following command:

   ```
   gcc kernel_info.c -o kernel_info
   ```

6. Run the compiled program

   Execute the compiled program by entering the following command:

   ```
   ./kernel_info
   ```

   Your terminal should display the output:

   ```
   System name: Linux
   Node name: os
   Kernel release: 5.15.0-67-generic
   Kernel version: #74-Ubuntu SMP Wed Feb 22 14:14:39 UTC 2023
   Machine hardware: x86_64
   ```