UNIST STEM Camp (Jan. 06, 2020)   - Mr.O
Practice: UNIX File Systems and Advanced Commands / Basic of Python

Overview

The purpose of this lesson is to understand UNIX file systems, including how to store the files in hierarchical directory structure, followed by exercises for advanced UNIX commands to handle directories and files. Additionally, you will practice Basic commands for using Python. As always, becoming familiar to UNIX requires a lot of efforts. Practice as much as possible to make it your own.

Make sure:

1. Get help from TA if any of the above things is not ready.

[man]

In these labs, you will be learning a couple of UNIX commands. Commands tell UNIX to run a program by that name. You can obtain information about commands on-line on the computer. System documentation is stored on-line and is accessed by issuing the command:

    man <name> <cr>

```
$ man passwd ↵
```

⇨ ‘q’ will get you out of the manual

```
$ man -k touch ↵
```

⇨ Info is another command that shows on-line manual

[clear] Clear the screen

```
$ clear ↵
```

[touch]

This command is used to create empty text files easily.

```
$ touch test1 test2 ↵
```

[ls]

When you first logon on to your account, you will see a `` `[s201xxxxx@unimaster1 ~]$` `` (call it `prompt’) on the left hand side of your screen. At the prompt, issue the command:

```
$ ls -l ↵
```

If you have completed your labs successfully, you may see several files you created in the lab exercises. The option ‘-l’ lists the detailed information of the files in the working directory. Then, try the following:

```
$ ls -lt ↵
```

The default listing of ‘ls’ is an alphabetical order. By giving ‘-t’ option, the files are sorted by modification time. Now try the following again and write the output:

```
$ ls -ltr ↵
```

Then, change the order of the options and write the output:

```
$ ls -trl ↵
```

Generally, the order of options doesn’t make difference in UNIX.

```
$ ls -al ↵
$ ll ↵
```

[pwd / mkdir / cd]

We can create an empty directory using ‘mkdir’ command. Type in the following commands:

```
$ mkdir 0107 ↵
$ ls ↵
```

Now change to the new directory by

```
$ cd 0107 ↵
[0107]$ pwd ↵
```

pwd : print working directory

Directories can be contained by other directories. Create two subdirectories under ‘0107’.

```
[0107]$ mkdir sub1 ↵
[0107]$ mkdir sub2 ↵
[0107]$ ls ↵
```

Run the following and print the result:

```
[0107]$ cd sub1 ↵
[sub1]$ pwd ↵
```

If you want to your home directory, just type ‘cd’ command.

```
[sub1]$ cd ↵          (go to 'home' directory)
[sub1]$ cd .↵         (go to the current directory)
[sub1]$ cd ..↵        (go to the upper directory)
```

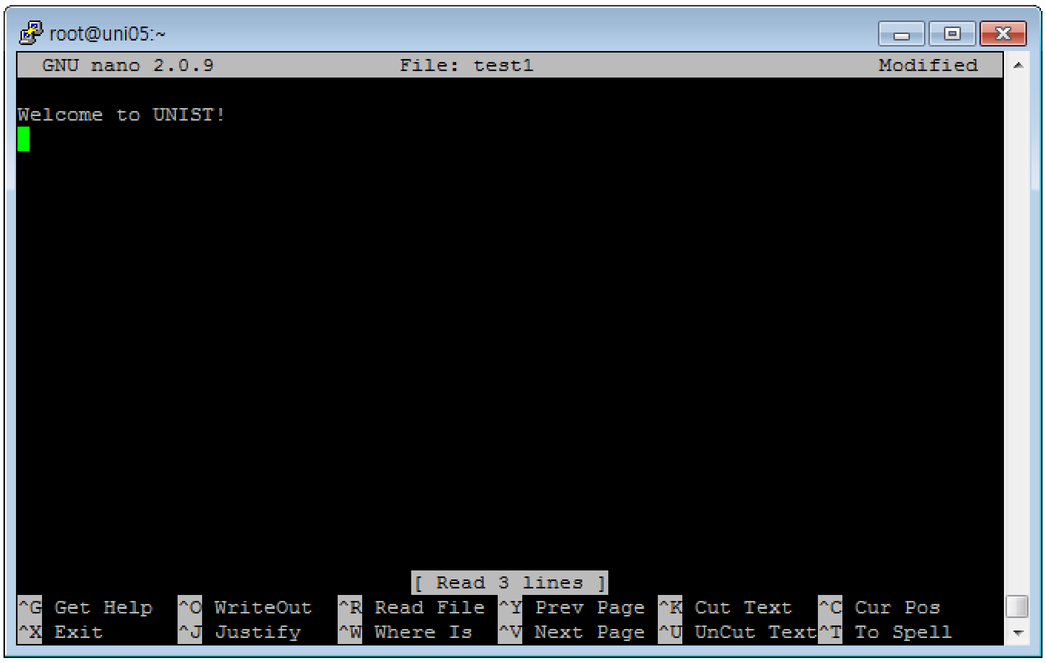
[nano / cat]

nano editor :

⇨ Control-key sequences are notated with a caret(^) symbol and can be entered either by using the Ctrl key or pressing the Esc key twice.

⇨ Escape-key sequences are notated with the Meta(M-) symbol and can be entered using either the Esc, or Alt key.

```
$ nano test1 ↲
```



```
root@uni05:~
  GNU nano 2.0.9              File: test1                          Modified

Welcome to UNIST!


                            [ Read 3 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

cat command : read a file and print out the file

```
$ cat test1 ↲      (display a file)
Welcome to UNIST
$ cat > test2 ↲    (edit the text )
Hello World
```

If you done editing, press Ctrl +'d' to save the file.

Run the following and print the result:

```
$ cat test2 ↲
Hello World
$ cat test1 test2 > test3↲    (combine files )
$ cat test1 >> test3↲         (append to a file)
$ cat -n test3 ↲              (show the line number )
```

[vi]

You will create a file, insert some text, make corrections and save the file. The vi editor is not as easy as a word processor and takes some time to learn. Keep hard work to make it your own.

```
$ vi practice.txt ↲
```

When you first get into the vi editor, all you will see is a screen full of tilde marks down the left side of your screen. At the left bottom, you will have the name of the file – in this case, "practice.txt"   [New file]. You will be typing in information on the lines that have the tildes. The vi editor has two different functional modes. They are:

⇨ Insert or append mode : When you are in the insert or append mode, you are typing in information into the file. This is where you input information into a file. You will also see the change, replace, and open modes used here. These are other ways to type in information into a file.

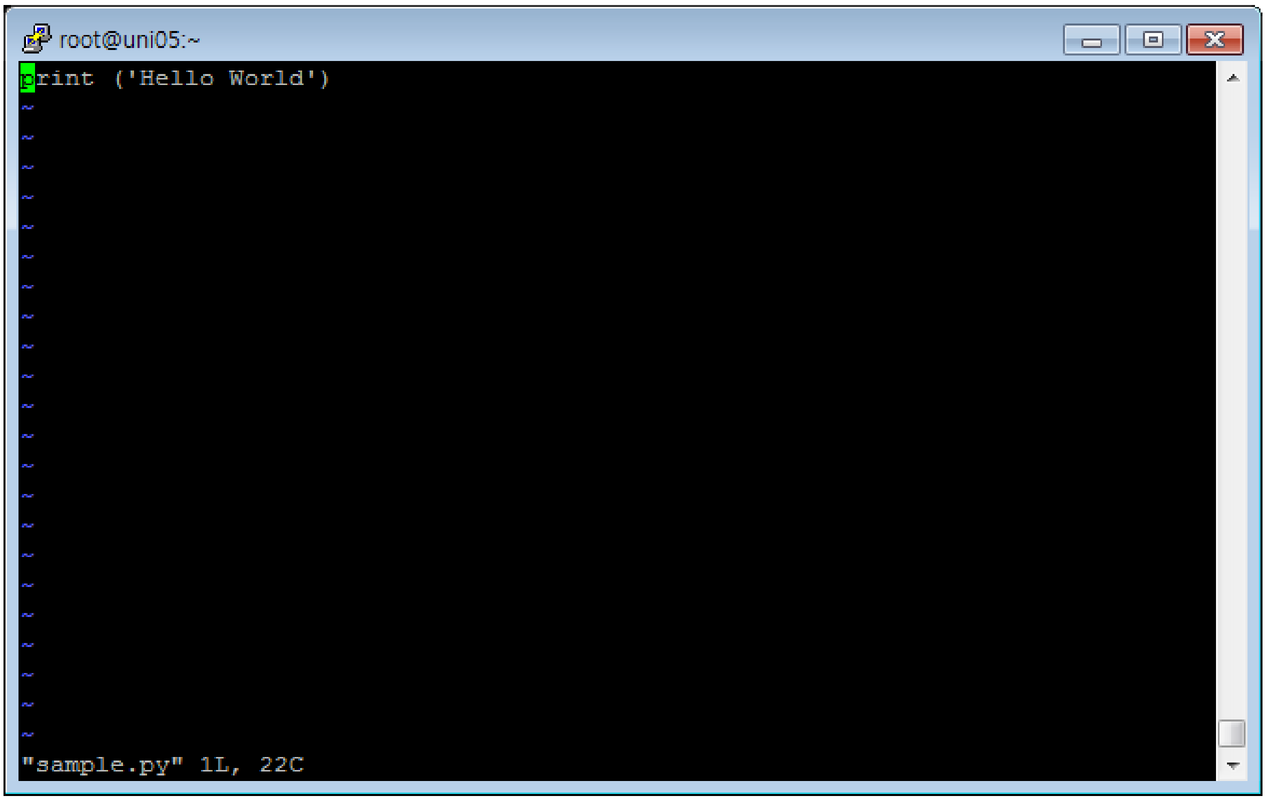
⇨ Command mode : This is where you want to make changes in the file that you have been typing. The

vi editor is a full screen editor. You can go over the entire document to make changes.
In order to enter the insert mode, type in the vi command : **i**
The insert mode allows you to enter text into the file you are creating. In order to make corrections after you have typed in the file, you must get out of the insert mode. You do this by pressing the **escape** key which is usually located at the top left of the keyboard, written **<Esc>**. When you finish with the insert mode, you will then press the <Esc> key and you will no longer have the message "—INSERT –" at the bottom of your screen.
Before you do anything else, you should save the document and then you can exit from this editor.
In order to save and exit, type in the command mode : **:wq or :wq!**

[cp / mv / rm]

Initially, when a directory is created, it is empty. Execute '**vi sample.py**', and just do '**:wq!**' to create an empty file. Then, do 'ls' to see the file has been created successfully.

```
$ vi sample.py ↲
```
(do ':wq! ↲' inside the editor)



```
print ('Hello World')
~
~
~
"sample.py" 1L, 22C
```

```
$ ls *.py  ↲
```

```
$ python sample.py ↲
Hello World
```

Copy the new file 'sample.py' to 'sample1.py', list the file names by 'ls' command, and print the result.

```
$ cp sample.py sample1.py ↲
$ ls *.py↲
```

Then, copy 'sample.py' to the sub directory, list the file names of the sub directory, and print the result.

```
$ mkdir sub1 ↵
$ cp sample.py ./sub1 ↵
$ cd sub1 ↵
[sub1]$ ls ↵
[sub1]$  ↵
```

Move to the parent directory, move ‘sample1.py’ to sub2, list the file name by ‘ls’, and print the result.

```
[sub1]$ cd .. ↵
$ mv sample1.py sub2 ↵
$ ls sub2 ↵
```

As a result, the file ‘sample1.py’ has been copied to ‘sub2’. We used ‘cp’ and ‘mv’ commands, but it seems inefficient. Let’s try better way. First, delete ‘sub2/sample1.py’.

```
$ rm sub2/sample1.py ↵
```

Go back to ‘sub1’ and complete the following single line command to copy ‘sample.py’ to ‘sub2’ by filling in the blank.

```
$ cd sub1 ↵
[sub1]$ cp sample.py ________ ↵
[sub1]$ cd .. ↵
$ ls sub2 ↵
```

What was printed on the screen? Do you see ‘sampe.py’?

____

[cp –r / rm –r]

The whole directory can be copied with its subdirectories including files by ‘cp –r’ command. Execute the following and print the results.

```
$ cp 0107 dummy ↵
```

____

The command ‘cp’ can only copy files. If you couldn’t create ‘dummy’, issue the following command and print the output:

```
$ cp –r 0107 dummy ↵
$ ls dummy ↵
```

____

If you have succeeded in copying the whole directory, try removing it. The command ‘rm’ can remove files. Give it a try and print the output:

```
$ rm dummy ↵
```

____

You must have failed and saw some error messages. Try the command ‘rmdir’ and print the output:

```
$ rmdir dummy ↵
```

____

The command ‘rmdir’ cannot remove non-empty folders. Like the relationship between ‘cp’ and ‘cp –r’, we can use ‘rm –r’ to delete non-empty directories recursively.

```
$ rm –rf dummy ↵
```

Were you successful? Otherwise, get help from TA.

____

By giving an option ‘-rf’, it doesn’t ask your confirmation. Think twice before using this command.

[rm / rmdir]

A directory including subdirectories can be removed by combination of ‘rm’ and ‘rmdir’.
Now go back to the parent directory, and remove the directory ‘sub*’ using ‘rmdir’ command.
Try the followings and print the result:

```
$ cd 0107 ↵
[0107]$ rmdir sub1 ↵
```

____

You must have seen some error message because ‘rmdir’ cannot remove non-empty directory. Without ‘rm –rf’ command, the files should be removed first. Get into ‘sub3’, delete the files, and print the results:

```
$ cd sub1 ↵
[sub1]$ rmdir sub3 ↵    (answer ‘y’ when asked)
[s20xxxx@uni01 sub1]$ cd ________ ↵    (fill in the blank to go to parent directory)
[s20xxxx@uni01 0107]$ rmdir sub1 ↵
[s20xxxx@uni01 0107]$ ls ↵
```

Fill the blank in the third line to move back to the parent directory. What was printed by the last ‘ls’?

____

If you still see the directory ‘sub1’, get help from TA.

[wget]

Wget is a utility for not-interactive download of files from the Web. It supports HTTP, HTTPS, and FTP protocols, as well as retrieval through HTTP proxies.

If you want to download ‘UNIST’ logo in the site, follow in the below.

⇨ address : http://www.unist.ac.kr/wp-content/uploads/2014/11/footer_logo.png

[End of lab practice 1]