Basic Linux Operations

Concepts to be delivered during tutorial:

1. System, Process, Executable, Shell, Commands, Command-Line-Interface
2. Filesystem, directory tree, Filesystem Hierarchy Standard
3. Standard input/output, pipe

1. Navigation

```
azureuser@test-hpl:~$ pwd
/home/azureuser
azureuser@test-hpl:~$ ls
azureuser@test-hpl:~$ cd ..
azureuser@test-hpl:/home$ ls
azureuser  ubuntu
azureuser@test-hpl:/home$ cd ..
azureuser@test-hpl:/$ ls
bin   dev  home        lib    lost+found  mnt  proc  run   selinux  sys  usr  vmlinuz
boot  etc  initrd.img  lib64  media       opt  root  sbin  srv      tmp  var
azureuser@test-hpl:/$ pwd
/
```

NOTE : `$` is the command prompt, after which you can type a command. After entering a command, there will be some output. Rule of thumb: read output carefully, esp. during more complex operations later. Linux command outputs are usually self-documenting.

2. Seeking for help

```
azureuser@test-hpl:/$ ls --help
Usage: ls [OPTION]... [FILE]...
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.
  -a, --all                  do not ignore entries starting with .
...
```

NOTE : `...` in the last line denotes omitted console outputs. Find the complete version by enter the corresponding command.

```
azureuser@test-hpl:/$ man ls

LS(1)       User Commands       LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about the FILEs (the current directory by default).  Sort entries alphabetically
       if none of-cftuvSUX nor --sort is specified.
```

```
...

       --block-size=SIZE
            scale sizes by SIZE before printing them.  E.g., `--block-size=M' prints sizes in  units  of
            1,048,576 bytes.  See SIZE format below.
 Manual page ls(1) line 1 (press h for help or q to quit)
```

TIP : Use arrow keys or `j` / `k` to move around the manual page. Use `/` to search for a keyword. Press `q` to end.

What is the man command? Try `man man`

3. Basic file/dir operations

Make directories:

```
azureuser@test-hpl:~$ mkdir mydir
azureuser@test-hpl:~$ ls
mydir
azureuser@test-hpl:~$ cd mydir/
azureuser@test-hpl:~/mydir$ pwd
/home/azureuser/mydir
azureuser@test-hpl:~/mydir$ cd ..
azureuser@test-hpl:~$ ls
mydir
azureuser@test-hpl:~$ rmdir mydir
azureuser@test-hpl:~$ ls
azureuser@test-hpl:~$
```

Ceate and view text files:

```
azureuser@test-hpl:~$ echo "this is my first file" > myfile
azureuser@test-hpl:~$ ls
myfile
azureuser@test-hpl:~$ cat myfile
this is my first file
```

NOTE : `echo` prints the string to `STDOUT` . `>` redirects `STDOUT` to a file. More on IO redirection. What is `cat` then? Try `man` or `--help` .

Move/Copy/Remove file:

```
azureuser@test-hpl:~$ ls
myfile
azureuser@test-hpl:~$ cat myfile
this is my first file
azureuser@test-hpl:~$ cp myfile myfile2
azureuser@test-hpl:~$ ls
myfile  myfile2
azureuser@test-hpl:~$ cat myfile2
this is my first file
azureuser@test-hpl:~$ mv myfile myfile.moved
azureuser@test-hpl:~$ ls
myfile2  myfile.moved
azureuser@test-hpl:~$ rm myfile2
azureuser@test-hpl:~$ ls
myfile.moved
```

TIP : Do more experiments like this. Use `ls` (probably with options like `-a` , `-l`) to inspect a dir. Use `cat` or `less` /more to inspect the content of a text file.

About filename:

1. Basically a flat string. No concept of "extension name". Though, people may have naming conventions sometime.
2. Files start with `.` is "hidden". Use `ls -a` to see them.

4. File download from the Internet

```
azureuser@test-hpl:~$ mkdir try-curl
azureuser@test-hpl:~$ cd try-curl/
azureuser@test-hpl:~/try-curl$ curl -JLO 'https://github.com/hupili/agile-ir/raw/master/data/Shakespeare.tar.gz'
--2014-01-14 03:02:09--  https://github.com/hupili/agile-ir/raw/master/data/Shakespeare.tar.gz
Resolving github.com (github.com)... 192.30.252.131

...

azureuser@test-hpl:~/try-curl$ ls
Shakespeare.tar.gz
```

Now you have downloaded Shakespeare's works, all in one compressed `archive Shakespeare.tar.gz` . Following is a shortcut to uncompress it:

```
azureuser@test-hpl:~/try-curl$ tar -xzvf Shakespeare.tar.gz
data/
data/sonnet-59.txt
data/sonnet-139.txt
data/sonnet-88.txt
data/sonnet-123.txt
data/sonnet-137.txt
data/play-twogents.txt

...

data/sonnet-134.txt
data/sonnet-93.txt
data/sonnet-24.txt
data/sonnet-3.txt
data/play-juliuscaesar.txt
```

What's `-xzvf` ? Try `man` or `--help` .

NOTE : Some commands have shorthand notation for multiple options. In the above example, `tar -xzvf YOUR_FILE` is equivalent of `tar -x -z -v -f YOUR_FILE` . Try the latter one yourself.

EXERCISE : Navigate the `data` dir and operate on those files, e.g. `cp` , `mv` .

EXERCISE : Get familiar with `tar` , `zip` , `gzip` , `bzip2` . You are very likely to get others' data in those formats.

EXERCISE : Get familiar with `curl` options. A simple crawler can be obtained by `curl -JLO START_URL` .

EXERCISE : Try to use `wget` to download the same file. Most Linux distribution has `curl` and/or `wget` by default.

Suppose you have finished processing `data` . Cleanup as follows:

```
azureuser@test-hpl:~/try-curl$ ls
data  Shakespeare.tar.gz
azureuser@test-hpl:~/try-curl$ ls data/
play-12night.txt         play-titus.txt              sonnet-122.txt  sonnet-152.txt  sonnet-42.txt  sonnet-72.txt

...

play-tempest.txt         sonnet-120.txt              sonnet-150.txt  sonnet-40.txt   sonnet-70.txt
play-timonathens.txt     sonnet-121.txt              sonnet-151.txt  sonnet-41.txt   sonnet-71.txt
azureuser@test-hpl:~/try-curl$ rm -rf data/
azureuser@test-hpl:~/try-curl$ ls
Shakespeare.tar.gz
```

NOTE : `rm -rf` is a powerful command. Use with great care.

5. Execute an executable file

Write your first shell script

```
azureuser@test-hpl:~$ cat > hello.sh
echo "hello world. My first shell script!"
azureuser@test-hpl:~$ ls
hello.sh
azureuser@test-hpl:~$ cat hello.sh
echo "hello world. My first shell script!"
```

`cat >` reads STDIN and redirect all the content to `hello.sh` . The second line `echo "hello world. My first shell script!"` is typed by you. After that press `ctrl+d` to end typing.

EXERCISE : Try this way to create more files. This is the simplest way to write small text files without using a text-based editor.

Make it executable:

```
azureuser@test-hpl:~$ ls -l hello.sh
-rw-rw-r-- 1 azureuser azureuser 43 Jan 14 07:26 hello.sh
azureuser@test-hpl:~$ chmod a+x hello.sh
azureuser@test-hpl:~$ ls -l hello.sh
-rwxrwxr-x 1 azureuser azureuser 43 Jan 14 07:26 hello.sh
```

The `x` character indicates that the file is executable. Read more.

Execute it:

```
azureuser@test-hpl:~$ ./hello.sh
hello world. My first shell script!
azureuser@test-hpl:~$ /home/azureuser/hello.sh
hello world. My first shell script!
```

NOTE : One often ignored syntax: If the executable is under current working directory, prefix it with `./` . Or else, the system will try to locate that command in PATH .

6. About shell commands (optional)

The commands you use, e.g. `ls` , `cd` , `mkdir` , are just some pre-installed executables in the system. You can find their location and verify that they are executable:

```
azureuser@test-hpl:~$ which ls
/bin/ls
azureuser@test-hpl:~$ ls -l /bin/ls
-rwxr-xr-x 1 root root 105840 Nov 19  2012 /bin/ls
```

`which` itself is an executable file:

```
azureuser@test-hpl:~$ which which
/usr/bin/which
azureuser@test-hpl:~$ ls -l /usr/bin/which
lrwxrwxrwx 1 root root 10 Mar 29  2012 /usr/bin/which -> /bin/which
azureuser@test-hpl:~$ ls -l /bin/which
-rwxr-xr-x 1 root root 946 Mar 29  2012 /bin/which
```

7. Automate your work by shell

Create a script, `download.sh` , with the following content.

```
# Clean previously downloaded data
rm -f Shakespeare.tar.gz
rm -rf data/
# Download
curl -JLO 'https://github.com/hupili/agile-ir/raw/master/data/Shakespeare.tar.gz'
# Uncompress
tar -xzvf Shakespeare.tar.gz
# list files
ls data/
```

TIP : No need to type in. Use `cat >` and copy paste the content into your terminal. The paste operations are different across terminals.

Content after `#` is comment.

Now execute the script:

```
azureuser@test-hpl:~$ chmod a+x download.sh
azureuser@test-hpl:~$ ./download.sh

...
```

The result is same as that when you type those commands in shell one by one directly. By writing scripts, you can automate tedious daily jobs. You will see some of this in the later part of this course.

EXERCISE : Shell scripts also supports common programming constructs, e.g. condition, loop, etc. Try to self-learn them from the Internet. Google “bash script” or something similar.

8. Text editor – VIM and others (optional)

Write codes in your desktop locally with you favourite GUI editor.

VIM is a powerful text editor. There are many tutorials and guides online.

Emacs is also a widely available and highly customizable text editor. It’ s interesting to learn some Emacs basic operations and concepts.

Sometimes, nano will be fired up to input cerntain information.

Text editors are just tools. Pick one that is most convenient to you.