GL Server

Everyone has access to GL servers. This is a Linux machine that can be used for different purposes. Some courses use GL server for projects and homework assignments. The following are the settings for communicating with GL servers. We can connect to the servers remotely:

User name: your myUMBC user name (your regular logon username)
Host address: gl.umbc.edu
Port number: 22

To connect to the GL servers we use ssh (Secure Shell) applications. The command line version of ssh in Windows, Mac, and Linux platforms exists by default. It also is possible to install a third party ssh application with user interface, e.g. putty.

To run the terminal in Windows you can search for cmd or cmd.exe. In Mac machines it is called Terminal.

To connect we logon in a terminal. Please note, when entering password, you do not see any reaction (your password will not be printed in the terminal, but linux machine is reading it when you type it in)

yourusername@gl.umbc.edu
Password: your regular myUMBC password

There is a two-factor login system in effect, therefore, after typing your password, it asks for another passcode that can be sent to you by the Duo application at your request. You need to choose the appropriate option and enter the passcode.

If it is the first logon, the terminal may ask about the security key. You can accept the default message that you see in the terminal.

Note: if you frequently need to login into GL server, you can bypass the two-factor login by installing the UMBC GlobalProtect VPN. The following link provides the information about GlobalProtect:

https://umbc.atlassian.net/wiki/spaces/faq/pages/30741112/VPN+Virtual+Private+Network

Linux OS

After logging in we are on the Linux command line (Linux prompt) and we can run any Linux command. Lets try the following commands to see what OS and what version the current machine is running. The following command prints all information about the running OS to the terminal.

>> cat /etc/os-release

Since there is a lot of information we can filter what we are looking for using the grep command.

>> grep '^NAME' /etc/os-release

And we get the following answer:

NAME=Fedora

Linux servers are extremely powerful and flexible machines that allow for almost any type of processing. That is the reason according to available statistics the majority of servers in the world are running Linux.

In order to use Linux servers we need to be comfortable with some fundamental commands. In what follows we practice those commands.

Similar to any other OS in a Linux machine we need to work with directories (folders) and files. It is a common practice before doing anything we check where we are at the current moment, the following command shows the current folder:

>> pwd

In a linux system folders are separated by a slash character. The command pwd prints the absolute path for the current folder. Normally after logon we are in the user home directory. We can use the command cd to change the current directory to another location. And we use the mkdir command to create a new directory. The command ls prints the list of directories and files in the current directory.

>> ls -l

The command mkdir creates a new directory in the current directory. The following command creates a new directory called ceti.

>> mkdir ceti

To change the current location to the new directory we use the cd command.

```
>> cd ceti
```

Wherever we are we can go back to the home folder using the cd command with the shortcut character ~.

```
>> cd ~
```

There are other path shortcuts. A dot in the path alway refers to the current folder. A double-dot in the path always refers to the previous folder (the parent of the current folder).

```
>> ls -l ./
>> ls -l ../
```

We can create an empty file in the current directory using the touch command.

```
>> touch 1.txt
```

We can rename an existing file to another name using the mv command.

```
>> mv 1.txt 2.txt
```

We can copy an existing file over another file (or create a new copy).

```
>> cp 2.txt 3.txt
```

We can remove an existing file using the rm command. To complete the remove operation we answer y to the question asked by Linux.

```
>> rm 3.txt
rm: remove regular empty file '3.txt'? y
```

Every Linux command has multiple options. We can find the options in the man page of a command. The following shows the manual page for the ls command. To navigate in a man page we can use the keyboard buttons enter (line by line forward), space (page by page forward), up and down arrow keys (move up and down line by line), page up and page down (move up and down page by page) and q (exit the man page).

```
>> man ls
```

We can use wildcard character * to specify a pattern for commands. The following examples will list all files with .txt extension. It filters out all other files.

```
>> ls *.txt
```

We can pass the results of one command to another to perform more complex operations. The following prints the number of files in the current directory. The command ls generates a list of files, then the list is passed to the next command wc through the pipe character (vertical bar). The command wc with the option -l counts the number of lines of its input.

```
>> ls -l | wc -l
```

The clear command removes everything from the terminal and clears the terminal.

```
>> clear
```

In Linux we can define the operations that every user can perform on a file, a file can be read only or writable, and it can be executable or not. There are three types of users, that is the owner of the file, the group that owner belongs to, and everyone else. The following example shows a file with its modes. In this example username is the owner of the file and the first rwx tells us that the owner can read and write to the file and execute it. The second mode indicator r-x tells us that the file is readable by the group rpc and the members of the rpc group can execute the file too. But they do not have the right to write to the file. The third mode indicator r-x tells us that everyone can read the file and execute it. We can change the file modes using the chmod command.

```
>> ls -l
-rwxr-xr-x 1 username rpc 235 Jul 30 15:51 info
```

To make a file executable by everyone we run chmod with the option a+x. In this command a means all.

```
>> chmod a+x info
```

To remove the executable right for others (except the owner and the group) we run chmod with the option o-x. In this command o means others.

```
>> chmod o-x info
```

To give execution right to self (owner of the file) we run chmod with the option u+x. For example, when we create a shell file we need to change the mode to executable to be able to run the file. In this command u means user.

```
>> chmod u+x info
```

On a command line system often we need to search and find a specific file. We use the find command to perform the search operation. The find command is a very powerful and useful tool. In the following example we search for all files and directories that in their name there is txt.

The dot indicates that the search starts in the current directory. The pattern '*txt*' indicates that we are looking for all names with txt in the middle no matter what comes before or after txt. The pattern should be in single quotation marks.

```
>> find . -name  '*txt*'
```

The find command has multiple options. In the following example we search for only files with the specific pattern. We use the option -type f for this purpose. To search for directories instead we use the -type d option.

```
>> find . -type f -name  '*txt*'
```

Sometimes we need to search for a term in files. In such a case we can use the grep command. Grep searches the contents of the specified files for the search pattern. The following example looks for the search pattern “some text” in .txt files in the current directory. The option n asks for the line numbers that the search pattern occurs in. The option i asks the command to ignore the case.

```
>>  grep -ni  'some text' *.txt
```

In linux we can create links to files or directories. A link is a pointer to the real file or directory. There are two types of links, i.e. hard links and soft links. A hard link is a copy of the original file or directory. It is not very convenient. However, soft links are only pointers and are very helpful because it allows linking to a file on another hard drive or a file in another account. The following shows an example of defining a soft link. In this example, the file 10.txt is a real file in the directory testfolder. The command creates a soft link to 10.txt. The link is called 10link and it resides in the current folder.

```
>> ln -s testfolder/10.txt 10link
```

We can use the ls command to see the detailed information about the link. In the output the l character at the beginning of the line shows us this is a link.

```
>> ls -l
lrwxr-xr-x 1 username rpc   17 Aug 11 10:35 10link -> testfolder/10.txt
```

In many cases we need to create a backup of files or directories. We can use the tar command to create an archive (backup file). This command writes the complete structure of the directory to an archive file in a compressed format. The following command creates an archive of the directory ceti. The file archive.tar is the result, the output of tar command. The option c means create. The option v stands for verbose; it writes the operations to the standard output. The option f means the archive will be a single file.

>> tar -cvf archive.tar ceti

We can extract the archive, i.e. reconstruct the complete structure of the ceti directory by running the following command. The option x means extract.

>> tar -xvf archive.tar

On the shared servers every user has a specific quota in the file system. If the user space goes beyond the quota the user won't be able to run any command. If we feel there is such a problem we can check whether our quota is full. Using the following command we can check what percentage of quota is used.

>> quota

When the account goes over quota we can use the du command to find what files are consuming the quota. The du command stands for disk usage. The following example shows the usage in the current folder. The option h stands for human readable.

>> du -h *

The following command shows the disk usage for all .txt files in the directory testfolder. We run this command in the parent directory of testfolder.

>> du -h testfolder/*.txt

Tools

There are multiple tools in Linux that facilitate the work. For example, if we need to edit a file there are multiple text editors in Linux. One of the text editors is EMACS.

Another example of existing tools is the language compilers. In a Linux machine we can write programs, compile them, debug them, and run them. For example, C++ build tools are available. A Python interpreter is available.

EMACS

The text editor emacs can be used in a terminal. We create a file or open an existing file. The following opens a file named info. If the file does not exist emacs creates an empty file.

>> emacs info

Once the file opens we can write in the file. To save the file we use the combination of ctrl+x+s buttons (keep down ctrl and press x and press s). To exit the editor environment we use the key combination of ctrl+x+c (keep down ctrl and press x and press c). To delete a line we put the cursor at the beginning of the line and use the key combination ctrl+k.

To search for a string in a file we use ctrl+s to change to search mode, then type the search string, emacs finds the first occurrence. To continue searching for the same string we use ctrl+s.

To go to a specific line number in the file we can use the Meta commands. The meta key is either alt button or esc button. For example, the combination of M+x means first we press the meta key and then we press x key, the editor gives us a command prompt, on the command prompt we type goto-line and press enter, then the command line waits for the line number, we type the line number and press enter. It is possible to to go to a line using “M+g M+g #number” sequence too.

The combination m+> goes to the end of the file and the combination of M+< goes to the beginning of the file. Please note, > refers to the greater-than button and < refers to the less-than button.

Nano

Nano is another text editor that exists in Linux machines. It is simpler than emacs however it is less powerful than emacs. To save a file we use the command ctrl+o and then enter. To exit the editor we use ctrl+x. The following commands opens a file named info.

```
>> nano info
```

To search for a string in nano we use ctrl+w and type the string and enter, nano finds the first occurrence of the string. To continue searching for the same string we continue using ctrl+w and enter.

Vi (vim)

Vi is another text editor that exists in Linux machines. Once we open a file in vi we use the i key (i command) to enter the insert mode. After entering the insert mode we can start to edit the file. To exit the insert mode press esc button. To save the file we press : (colon) then type w and press enter. To exit the editor press : then type q and press enter. To exit the editor without saving press : then type q! And press enter.

To search for a string we type / followed by the string and enter, vi finds the first occurrence of the string. To continue searching for the same string we press the n key.

Shell Scripting

A shell is a command line interpreter that receives commands and executes them. There is a shell in every OS. Windows uses MS-DOS for its shell and it has a power shell for power users. Mac OS is using a Linux shell. There are multiple shells available for Linux machines. All shells support the fundamental required commands. Since there are differences between shells we need to know what shell we are using.

The shell allows us to write and run scripts. Scripts contain multiple OS commands and are very helpful in automating the tasks. The shell normally provides the possibilities of repeating commands and executing commands conditionally. This is very similar to programming. The following example loops through the current directory, finds all files with .txt extension and creates a copy of them with a .bak extension. This allows creating a backup of our files when we have many files with one command. At the beginning of the script we indicate this is a bash shell.

```
#!/bin/bash
for file in *.txt
do
   cp "$file" "$file.bak"
   echo "copied $file to $file.bak"
done
```

We write the script in a file. Then we make it an executable file and we run it in the current directory.

We can check the current shell using the following command.

```
>> echo $SHELL
/bin/tcsh
```

The following example is a tcsh shell. It lists the files in the current directory. The character \t means tab.

```
#!/bin/tcsh
foreach file (*)
   echo "\t$file"
end
```

If we write the script in the current shell we can run the script using source command without adding the first line in the file.

```
>> source scriptfile
```

Python

We can write and execute programs in Python. To check the python that is available we can use which command. The following example shows us that only Python 3 is available. Since there are differences between Python 2 and 3 this means we may not be able to run some Python 2 programs here.

```
>> which python
/usr/bin/python
>>ls -l /usr/bin/python
lrwxrwxrwx 1 root root 9 Aug 31  2021 /usr/bin/python -> ./python3
>> which python3
/usr/bin/python3
>> ls -l /usr/bin/python3
lrwxrwxrwx 1 root root 9 Aug 31  2021 /usr/bin/python3 -> python3.9
```

The output of the above commands shows us that by running the python command we run python 3.

Python is another type of scripting for automating the tasks. The following example lists all files in the current directory.

```
import os
files = [f for f in os.listdir('.') if os.path.isfile(f)]
for f in files:
   print("\t",f)
```

The following shows the execution of a python script named p.py. The extension .py is not necessary. It is for clarification.

```
>> python p.py
```

It also is possible to run the python file as executable by adding the following line at the beginning of the script and make the file executable using the chmod command.

```
#!/bin/python
```

C++ Build Toolset

We can run programs in many languages on Linux including C++ or Java. The following command shows compiling a C++ program. In this command g++ is the name of the compiler, the option -W (capital) tells the compiler to print all warning messages, and the option -g tells the compiler to generate debug information. The option -o indicates the name of the output executable file.

>> g++ -W -g source.cpp -o output.exe

Please note that ideally we should fix all warnings since they are sources of bugs.

In C++ programs it is the job of the programmer to manage the memory. Memory management includes allocating memory, deallocating memory, and preventing access violations. We can use the tool valgrind to find out about the memory leaks or memory errors in a C++ program. The following command checks whether the output.exe shows any problem in regard to memory.

>> valgrind output.exe

Resources

Website (Linux command line): https://lym.readthedocs.io/en/latest/index.html

Book: https://archive.org/details/linux-commands-handbook/mode/2up

Online Linux Terminal: https://bellard.org/jslinux/