**Q. You can see a file's owner and group using the 'ls -l' command. Take a look at your user and group IDs.**

id

**Q.** **Now take a look at the owner and group of your home directory.** **This information is in the third and fourth column of detailed listings produced by 'ls -ld'. Note that you are currently in your home directory**

ls -ld

**Q.** **If the permissions on a file were set so that everyone could read from or write to the file, how would that be represented symbolically (within the output of an 'ls -l' command)?**

Rw-rw-rw-

**Q.** **Write the symbolic representation of a permission setting where the user could read, write, or execute the file; group members could read or execute only; and others could only**

**execute the file.**

rwxr-x—x

**Q. What is the octal equivalent to this symbolic permission: rwxrw-r—**

764

**Q. Now try writing the symbolic representation for this octal permission: 774**

Rwxrwxr--

**Q. For example, to set file permission of 744 (rwxr--r--) on the file 'cows', you would use the command**

'chmod 744 cows'.

**Q. Now set permissions of 761 on 'file1'.**

Chmod 761 file1

**Q. Now check the permissions on 'file1'**

ls – l fiel1

**Q.. Now try setting the permissions on 'file1' to 640 using octal notation.**

Chmod 640 file1

**Q. For example, to add read permission for Other for the file 'file1', you would write:**

chmod o+r file1

**Q. To add read and write permission for everyone:**

chmod ugo+rw file1 or chmod a+rw file1

**Q, To remove read permission for the Group:**

chmod g-r file1

**Q. To set the User permission to rw:**

chmod u=rw file1

**Q. To add read permission for User and remove read for Group and Other**:

chmod u+r,go-r file1

**Q. To add read and execute permission for everyone and remove write permission for Other:**

chmod a+rx,o-w file1

**Q. Use chmod's symbolic representation to add read permission for everyone to 'file1'.**

chmod a+r file1

**Q. use chmod's symbolic representation to remove write permission for User from 'file1'.**

chmod u-w file1

**Q. Remember that 'x' means Passthrough permission on directories? If passthrough permission is turned off, then the contents of the directory and subdirectories is effectively cut off.**

**This provides a simple way to prevent other people from accessing the contents of your files: simply remove 'x' permission for Group and Other on your home directory (use '~' to represent your home directory).**

**Enter a command to do this using symbolic notation (it won't be run**).

Chmod go-x ~

**Q. For example, to prevent the system from setting write permission for group, and prevent it from setting ANY permission for other, you would use a umask of 027. Also note that**

**umask only affects NEW files created, not existing files**

umask 027

**Q. What is the symbolic representation, as displayed by the 'ls -l' command, for a permission setting for a file where the User can read, write, and execute; the Group can read; and the Other users can't do anything?**

rwxr-----

**Q. What is the octal permission number that represents 'rwxr-----'?**

740

**Q. What is the command to grant all permissions for everyone for the file 'foo' (using octal representation)?**

Chmod 777 foo

**Q. What is the command to remove read permission for Group and Other on the file 'green'?**

**(Hint: Since you don't want to change other permissions, use a symbolic permission instead of an octal permission).**

chmod go-r green

**Q. What is the command to add passthrough permission for everyone for your home directory (~)?(Hint: Since you don't want to change other permissions, use a symbolic permission again instead of an octal permission, Hint: Passthrough is execute).**

chmod a+x ~ chmod a=x ~ 는 안됐다.

**Redirection And Pipes - Completed**

**Q.** **Enter the command 'who am i' to find out the file name for your terminal (look in the second column of output)**

who am i

**'who am i' reported that your terminal device is pts/4 (second column of the output). That is a relative pathname based on the device directory /dev, so the full pathname for**

**your terminal is /dev/pts/4**

**Q. Check to make sure that file exists by getting a detailed listing of it.**

ls -l /dev/pts/4

**Q. Now use the 'file' command to see what kind of a file /dev/pts/4 really is.**

file /dev/pts/4 입력시 /dev/pts/4: character special 출력

As you saw, /dev/pts/4 is a 'character special' file, which means that it is a device which accepts one character at a time and can send one character at a time.

It is possible to use the shell to connect the standard file descriptors to other files or devices, or even to other commands. The next sections of this Assignment discuss how this is done.

**Q. Let's redirect the output of the 'date' command into the file 'today'.**

date > today

**Q. Next, try redirecting the output from 'cal' into a file named 'october'.**

cal > october

**Q. Let's redirect the output of the 'whoami' command into that same file ('october').**

Whoami > october

**Q. Try using '>>' to append the output from 'cal' to the file 'october' without overwriting what is already there**

Cal >> october

**Q. Let's put the current month's calendar into the file 'caltest'.**

Cal > caltest

**Q. Use 'tr' to translate the contents of 'caltest' to uppercase.**

tr "[a-z]" "[A-Z]" < caltest

The command to translate to uppercase is 'tr "[a-z]" "[A-Z]"' -- add '< caltest' to make the command take its input from the file 'caltest'

**Q. You should have seen this month's calendar appear on your screen in uppercase**

Cat < caltest

Some commands take their input from stdin if there is no file argument on the command-line. For example, you can give a filename argument to the 'cat' command; but if you do

not include a filename argument, then the input is taken from stdin (as you found out earlier).

This means that 'cat < caltest' and 'cat caltest' do the same thing. Enter 'cat < caltest' now to prove this.

**Q. Let's intentionally create an error while redirecting the output**

cal 1963 8 > caltest 입력 cal: illegal month value: use 1-12 라는 오류 뜸

Notice that the error message appeared on the screen even though the output of the command was redirected. This is why standard error is separate from standard output.

To redirect standard error we use the same symbols as we use for redirecting standard output -- '>' and '>>' -- except that we add the file descriptor number for stderr (2)

directly in front of the symbol, like this:

2> file (overwrites)

2>> file (appends)

**Q. This time, let's intentionally create an error while redirecting stderr into a file.**

cal 1963 8 2> caltest

cat caltest 입력시 cal: illegal month value: use 1-12 출력

**Q. For example, we can take the standard output of 'cal 2020' and connect it to the standard input of 'grep' so that we can specify which lines to output on the screen.**

cal 2020 | grep er 입력

July August September

October November December 라고 출력

**Q. Notice that the commands are connected from left-to-right. The standard output of 'cal' was connected to the standard input of 'grep'.**

**Let's try using 'head' to view the first 10 lines of the output from 'cal 2020'.**

cal 2020 | head

2020

January February March

Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa

1 2 3 4 1 1 2 3 4 5 6 7

5 6 7 8 9 10 11 2 3 4 5 6 7 8 8 9 10 11 12 13 14

12 13 14 15 16 17 18 9 10 11 12 13 14 15 15 16 17 18 19 20 21

19 20 21 22 23 24 25 16 17 18 19 20 21 22 22 23 24 25 26 27 28

26 27 28 29 30 31 23 24 25 26 27 28 29 29 30 31Q.

**Q. Now let's add 'tail -5' to get just the last 5 lines of those first 10 lines.**

cal 2020 | head | tail -5

**Q. Let's use 'ls -l' to get a detailed list of files in your home directory, and then use 'sort' to sort the list by file size.**

**For the 'sort' command, we'll need the arguments '-k 5' to sort on the 5th column (file size) and '-n' to sort numerically instead of alphabetically.**

ls -l | sort -nk 5

**Q. The files were sorted by size, with the largest at the bottom. But what if we only want to see the 2 largest files? Try the same command, but add 'tail -2' to see only the last two files listed.**

ls -l | sort -nk 5 | tail -2

**Inf.   
Remember that >, >>, < only connect commands to files.**

this will work:

cal -y > today

These will not work:

cal -y > sort -n

sort -n < cal -y

**the pipe symbol '|' can only be used to connect commands.**

this will work:

cal -y | sort -n

This will not work:

cal -y | outputfile

**Pipes are written left-to-right. In this example:**

a | b | c

The stdout of command 'a' is connected to the stdin of 'b', and the stdout of 'b' is connected to the stdin of 'c'

Data goes into the left side of a pipe and flows out the right side

Therefore you could do something like this:

a <inputfile | b | c

But not:

a | b | c <inputfile

The problem with the second example is that the command shows the stdin of 'c' connected to both 'inputfile' and the stdout of 'b'.

**Q. Redirect the output of 'cal -y' into a new file called 'review1'**

cal -y > review1

**Q. Redirect the output of 'date' so that it is added to the end of the file 'review1'.**

date >> review1

**Q. Display a listing of the directory '/etc' one screen at a time.   
(Remember to press <q> when you want to quit).**

ls -l /etc | more

Q. The file 'threelines' contains 3 lines of text. Display the middle line only on the screen. You are not allowed to use the more command.

head -2 threelines | tail -1

**Q. Display the current month using 'cal', translating to uppercase using 'tr'**

cal | tr "[a-z]" "[A-Z]"

**Q. Run the command 'mkdir old' and redirect any error messages into the file 'errors'.**

mkdir old 2> errors

**Q. Search the file 'data' for all of the lines that contain the pattern 'linux' and put those lines in the file 'matches'.**

grep linux data | mv matches

**Q. The file 'numbers' contains a list of numbers. Write a command to place the largest one of those numbers into the file 'largest' (there should be nothing else in that file). Do not use the 'head' command in your answer**.

sort -n numbers | tail -1 > largest

**Q. Create a link to the 'cars' file called 'cars2'**

ln cars cars2

**Q. Look at detailed information about the 'cars' and the 'cars2' files in the current directory:**

ls -l

**Q.** **Note that creating an additional link to a file, even if in a different directory, is not a method of backing up the file. If the hard drive fails, then the physical file can still be corrupted or destroyed regardless of the number of links.**

**Try the '-i' option of the ls command:**

ls -i

**The '-i' option of ls gives the 'inode' number of filenames. 'inode' stands for information node, and contains all the information about a file, including permissions, owner,**

**group, create date-time, last modified date-time, and other pieces of information**

**Q**. **Symbolic links are also called soft links** **In order to give a file a symbolic link, the 'ln' command is used with the '-s' option.**

**Give 'cars' the symbolic link 'cars3':**

ln -s cars cars3

**Q. Now take a look at the files:**

$ ls -l

total 8

-rw-rw-rw- 1 ssuh6 users 445 Oct 21 20:08 cars

lrwxrwxrwx 1 ssuh6 users 4 Oct 21 20:11 cars3 -> cars

-rw-rw-rw- 1 ssuh6 users 128 Oct 21 20:08 new

**Q. To demonstrate, start by displaying the contents of the directory '~uli101/2020c/sample\_dir3/linked\_directories/sample\_files':**

ls -l ~uli101/2020c/sample\_dir3/linked\_directories/sample\_files

**Q. It's very cumbersome to type a long pathname such as**

**~uli101/2020c/sample\_dir3/linked\_directories/sample\_files.**

**If we were often accessing the files in this directory, we could**

**create a link.**

**Let's create a symbolic link to the directory, and call it 'linkdir':**

ln -s ~uli101/2020c/sample\_dir3/linked\_directories/sample\_files linkdir

Q**. Now we can refer to 'linkdir' instead of**

**~uli101/2020c/sample\_dir3/linked\_directories/sample\_files.**

**Copy the file 'cars' from 'linkdir' to the current directory:**

cp linkdir/cars .

**Q. Create a subdirectory called autos within the current directory:**

mkdir autos

**Q. Create an additional name (hard link) to the file 'cars'. The link should be called 'cars.link' and should be in the directory 'autos':**

ln cars autos/cars.link

**Q. Create a link to the directory '~uli101/2020c/sample\_dir3/linked\_directories/sample\_files'. The link, called 'samples', should be in the current directory.**

ln -s ~uli101/2020c/sample\_dir3/linked\_directories/sample\_files samples

**Q. There is a command called 'showtree' in the directory you just linked to. Execute this command using the newly created 'samples' link**

samples/showtree