ENGG1340 Computer Programming II

COMP2113 Programming Technologies

**Module 1 Checkpoint Exercise**

Name: Filbert David Tejalaksana

University ID: 3035945699

**Instructions:**

For each single question or each group of questions in the Checkpoint exercise, please type your answer right after the question in this Word document. Please refer to the example below.

Checkpoint 0:

What is the meaning of the command “date”?

Ans: The “date” command prints the current date of the current machine

**Checkpoint 1.1**

Now, let’s try to answer the following questions. Although you haven’t been taught the meaning of the following commands, you can display the manual page of these commands and learn their meanings by yourself

1. Why do we need to learn command line although we can use a GUI to control a computer?
2. What is the meaning of **ls -t**?
3. What is the meaning of the command **pwd**?
4. What is the meaning of the command **rm**?
5. What is the meaning of the command **mv**?
6. Suppose that the **fileA** does not exist in your present working directory, what is the meaning of the command **touch fileA**?
7. What is the meaning of the command **tar**?
8. What is the command for creating an archive **files.tar** from two files named **fileA** and **fileB**?

Ans:

1. There are several reasons to use command line instead of GUI. Other than operating faster than GUIs because they require less system resources to display fancy graphics, command line allows for ease of use if one masters it completely, eliminating the need to even let go of one's keyboard to use a mouse. Command line also increases the flexibility of use, allowing for a larger range of commands unavailable to the user without installing extra software, for example, mass renaming files with the same prefix.
2. **ls** will list out all the files in the current directory by default, if not given additional arguments. **-t** argument allows for **ls** to list out all the files in the targeted directory sorted by modification time, newest first.
3. **pwd** command will print out the full filename of the current directory. It takes in additional arguments to modify the command.
4. **rm** command removes files or directories. It takes in the path of the file or just the filename if it is in the working directory, as well as optional additional arguments.
5. **mv** command can move or rename files. It takes in the source and destination as well as optional additional arguments.
6. Creates an empty **fileA** in the current working directory.
7. **tar** command is used to archive files or to takeout files from archives. In the case of archiving, the command should look like **tar –c -f archive.tar foo bar**, where: -**c** means create, **-f** determines filename of archive, in this case **archive.rar, Archive.tar** is the filename of the created archive, and **foo** and **bar** are the files being archived into **archive.tar. -c** is an example of an argument for **tar**, but it does have a lot of other usages, such as extracting files from the archives, removing and appending files from and to the archive, as well as to list out files in an archive.
8. $ tar –c -f files.tar fileA fileB

**Checkpoint 1.2a**

Assume we have logged in Ubuntu and started a bash shell. The current directory is the home directory, i.e., ~ . We want to perform the following tasks sequentially. For each of the tasks below, please state the shell command(s) used to accomplish it.

1. Create a new subdirectory “***assignments***” under ~.
2. Create a new subdirectory “***assignment 1***” under “***assignments***”. (Note that we are creating one subdirectory “***assignment 1***” but not two subdirectories “assignment” and “1” )
3. Remove the directory “***assignments***” and all its subdirectories.

Ans:

1. **$ mkdir assignments**
2. **$ cd assignments**

**$ mkdir ”assignment 1”**

1. **$ cd**

**$ rm –r –f assignments**

**Checkpoint 1.2b**

**[Self-learning question]** - You need to search for the information on the Internet to answer this question.

There is another way to modify the permission, which is called the Absolute mode.

* 1. Explain the meaning of chmod 666 hello.txt
  2. Explain the meaning of chmod 700 hello.txt
  3. What is the chmod command, in absolute mode, to set the following permission for *hello.txt*?

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **User** **permissions** | | | **Group Permission** | | | **Other permission** | | |
| r | w | x | - | w | - | r | - | - |

* 1. The administrator says that “One does not simply 777 their entire server”, explain what the problem is if we chmod 777 for all the files.

Ans:

1. The number 6 indicates permission to read and write. Assigning hello.txt permission with chmod code 666 will give the user, group and others permission to only and read and write the file hello.txt.
2. The number 7 indicates permission to read, write and execute. The number 0 indicates full restriction. Assigning hello.txt permission with chmod code 700 will allow the user to read, write and execute the file hello.txt while restricting group and others from reading, writing or executing the same file.
3. **chmod 724 hello.txt**
4. Chmod code 777 will give access for anyone to view, modify or delete files in the server. Giving permission code 744 might be more appropriate in the case where the server is public to avoid and deletion or modification to files except if done by the user.

**Checkpoint 1.3**

Now you may have a doubt: I understand how **diff** works, but why is the output claimed to be the difference between the two files?

Consider the two files below:

|  |  |
| --- | --- |
| $cat question1A  Apple  Boy  Cat  Dog  Egg | $cat question1B  Boy  Cat  Egg |

Note that file **question1B** is created by removing “Apple” and “Dog” from the file **question1A**.

1. What will be the output if we execute the following command (Please try to think about the output before trying it in the shell)? Please explain your answer.

|  |
| --- |
| $ diff question1A question1B |

1. What will be the output if we execute the following command (Please try to think about the output before trying it in the shell)? Please explain your answer.

|  |
| --- |
| $ diff question1B question1A |

Ans:

1. **1d0**

**< Apple**

**4d2**

**< Dog**

The first line of file question1A will be deleted and the the file will be sync at like 0. The fourth line of the file will then be deleted and the file will be sync at line 2.

1. **0a1**

**> Apple**

**2a4**

**> Dog**

A line will be added to file question1B to line 0 from line 1 of file question1A. Line 4 from file question1A will then be added to file question1B at line 2.

**Checkpoint 1.4**

This is a challenging exercise! You need to understand the shell commands and the techniques introduced in the previous sections to work on this task.

The following C++ program *gen4.cpp* reads in a 4-character string from the input and generates all possible permutations from the 4 characters.

|  |
| --- |
| //gen4.cpp  #include <iostream>  #include <string>  int main() {  std::string s;  std::cin >> s;  for (int i = 0; i < s.length(); i++) {  for (int j = 0; j < s.length(); j++) {  for (int k = 0; k < s.length(); k++) {  for (int l = 0; l < s.length(); l++) {  if (i != j && i != k && i != l && j != l && j != k && k != l) {  std::cout << s[i] << s[j] << s[k] << s[l] << std::endl;  }  }  }  }  }  return 0;  } |

To compile *gen4.cpp*

|  |
| --- |
| $ g++ gen4.cpp -o gen4 |

The input of the program should be stored in the file *gen4\_input.txt* with the following content.

lopo

***gen4\_input.txt***

1. Give **ONE** command (one line of command(s)) to run the *gen4* with *gen4\_input.txt* as input and redirect the result to a file named *gen4\_output.txt*.

Hints:

|  |
| --- |
| $ *[your\_command]*  $ cat gen4\_output.txt  lopo  loop  lpoo  lpoo  …  $ wc gen4\_output.txt  24 24 120 gen4\_output.txt |

\*The output file should contain all permutations of the letters ‘l’, ‘o’, ’p’, and ‘o’. There should be 24 permutations in total.

Ans: **./gen4 < gen4\_input.txt > gen4\_output.txt**

1. Give **ONE** command to sort the words in *gen4\_output.txt* in alphabetical order, and then also remove the adjacent duplicate lines and finally store the result in a file named *sort\_uniq.txt*.

Hints: Consider the command **uniq**

|  |
| --- |
| $ [your command]  $ cat sort\_uniq.txt.  loop  lopo  lpoo  olop  olpo  oolp  oopl  oplo  opol  ploo  polo  pool  $ wc sort\_uniq.txt  12 12 60 sort\_uniq.txt |

\*There should be 12 unique words total.

Ans: **sort < gen4\_output.txt | uniq > sort\_uniq.txt**

1. Give **ONE** command to check the spelling in *sort\_uniq.txt* and store the misspelled words into another file named *misspell.txt*.

Ans: **spell sort\_uniq.txt > misspell.txt**

1. Now *sort\_uniq.txt* contains all distinct generated words, and *misspell.txt* contains all misspelled words. The differences between the two files are the meaningful 4-character words. Give **ONE** command to return the correctly spelled words as shown below:

|  |
| --- |
| $ [your command]  < loop  < polo  < pool |

Hints: Consider the command **diff** and **grep**.

Ans: **diff sort\_uniq.txt misspell.txt | grep ”<”**

Checkpoint 1.5

Consider the file *question1.txt*.

2011111111,John,M,98

2011222222,Marry,F,85

2011333333,Sally,F,85

2012111111,Kit,M,86

2012222222,Ben,M,97

2012333333,Smitty,F,92

2012444444,Jolly,F,93

2012555555,Ken,M,100

Figure 1 question1.txt

1. Give ONE command to return the lines that contain the record of Kit

Hints:

|  |
| --- |
| $ [Your command]  2012111111,Kit,M,86 |

Ans: **grep ”2012111111“ question1.txt**

1. Give ONE command to find the students with UID begin with “**2012**” (i.e., To find the lines that begin with 2012)

Hints:

|  |
| --- |
| $ [Your command]  2012111111,Kit,M,86  2012222222,Ben,M,97  2012333333,Smitty,F,92  2012444444,Jolly,F,93  2012555555,Ken,M,100 |

Ans: **grep ”2012“ question1.txt**

1. Give ONE command to return the lines that contain the record of the students who are both:

* UID start at **2012**, and
* Name starts with the characters **J** or **S**

Hints:

|  |
| --- |
| $ [Your command]  2012333333,Smitty,F,92  2012444444,Jolly,F,93 |

Ans: **grep –E ”2012[0-9]{6},[JS]“ question1.txt**