**CSc 3320: Systems Programming**

Spring 2021

Homework

# 3: Total points 100

Submission instructions:

1. Create a Google doc for each homework assignment submission.
2. Start your responses from page 2 of the document and copy these instructions on page 1.
3. Fill in your name, campus ID and panther # in the fields provided. If this information is missing in your document TWO POINTS WILL BE DEDUCTED per submission.
4. Keep this page 1 intact on all your submissions. If this *submissions instructions* page is missing in your submission TWO POINTS WILL BE DEDUCTED per submission.
5. Each homework will typically have 2-3 PARTS, where each PART focuses on specific topic(s).
6. Start your responses to each PART on a new page.
7. If you are being asked to write code copy the code into a separate txt file and submit that as well.
8. If you are being asked to test code or run specific commands or scripts, provide the evidence of your outputs through a screenshot and copy the same into the document.
9. Upon completion, download a .PDF version of the document and submit the same.

Full Name: Thach Pham

Campus ID: tpham84

Panther #: 002416419

**10 pts for the neatness factor of your presentation.**

**PART 1: 30pts**

1. For each command tryout at least one example provided in **Chapter 3** of the Unix textbook. Feel free to use your own example. Show the screenshot for each command’s output. Present your output in a tabular form with column 1 as index (1,2,3..), second column as the command, third as the usage, fourth as the screenshot of the output. You can just show a small snapshot for the output -- we do not need the entire screen’s image.

|  |  |  |  |
| --- | --- | --- | --- |
| Index | Command | Usage | Output |
| 1 | at | Scheduling commands to be execute at certain time.  For example: we use ‘at 9:23’ to schedule job at 9:23. After you enter the command, the terminal will prompt you what jobs to be done at the time. After you finish entering commands, hit Ctrl-D to end input. | at\_command.png |
| 2 | awk | Usually used for text processing and it has many ways to do it. For example, we can use awk to show lines contain certain word or string.  ‘awk ‘/bone/ {print}’ myexamfile.txt’ can be used to print out lines containing word ‘bone’ in myexamfile.txt | awk\_command.png |
| 3 | cmp | Used to compare two files byte by byte to see if two files are identical.  For example, ‘cmp myexamfile.txt phone\_book.txt” will compare these two files |  |
| 4 | compress/uncompress | command not found in bash shell | N/A |
| 5 | cpio | we use it to archive files into special \*.cpio files. For example, “ls \*txt | cpio -ov > backuptxt” can be used to archive all .txt files into backuptxt.cpio file. | cpio\_command.png |
| 6 | cron/crontab | Creating a table schedule to do certain command at certain time. We can use vi to create a crontab file. “vi hello.cron” consist command to print “Hello World!” every 1st and 15th day of the month, and also every Monday. To register the crontab hello.cron, we use “crontab hello.cron”. | hello.cron\_command.png  crontab\_command.png |
| 7 | crypt | crypt command not found in bash shell | N/A |
| 8 | diff | Compare the differences between two files and show what lines in which file to be deleted or replace.  For example, “diff -i myexamfile.txt phone\_book.txt” compare these two files and point out what needs to be done to make them similar. The -i option telling them to ignore case. | diff\_command.png |
| 9 | dump | No manual entry for dump | N/A |
| 10 | egrep | Search filter text in a file with extended regular expression. For example, we use “egrep -i “.\*ne” myexamfile.txt” to filter texts with the expression that has zero or more characters before “ne” in myexamfile.txt. The -i option makes it case insensitive. | egrep\_command.png |
| 11 | fgrep | Only search a fix string. For example, “fgrep “muscle fiber” myexamfile.txt” to find the string “muscle fiber” in file myexamfile.txt | fgrep\_command.png |
| 12 | find | Find things that match certain area. For example, “find . -name ‘\*.c’ -print” find files in current direction that end with .c and print those out. | find\_command.png |
| 13 | grep | Similar to egrep but it only use regular expression, not extended expression. For example, “grep “.ne” myexamfile.txt” searches for terms that start with any character and end with ne. | grep\_command.png |
| 14 | gzip/gunzip | gzip compress the file. “gzip -c myexamfile.txt > myexamfile.txt.gz”. The command compresses the myexamfile.txt to myexamfile.txt.gz with -c option directing the output. gunzip unzip files. “gunzip -v \*.gz”. The command unzips .gz files. | gzip\_gunzip\_command.png |
| 15 | ln | Creating hard link and soft link in directories. We first use “mkdir tmp” to make a tmp directory. We then use “ln \*.txt tmp” to link all .txt files to tmp directry. We can use “ls -l tmp” to check content of tmp. | ln\_command.png |
| 16 | mount/umount | Mount and unmount devices to terminal. | mount\_umount\_command |
| 17 | od | Use “od /bin/od” to display file od in octal | od\_command |
| 18 | perl | Rolling your own program | N/A |
| 19 | sed | Use ‘sed -n ‘s/bone/skeleton/p’ myexamfile.txt’ to substitute the word “bone” with “skeleton”. -n option print only the processed line. | sed\_command.png |
| 20 | sort | Use “ls | sort -n” to sort all files in directory in alphabetical order. | sort\_command.png |
| 21 | su | Use to subsitute user, but I don’t have other account. | N/A |
| 22 | tar | Archive the current directory “tar -cvf thachpham .” with -cvf option | tar\_command.png |
| 23 | tr | “tr a-z A\_Z < myexamfile.txt” translates all lower case character to upper case. | tr\_command.png |
| 24 | time | Time how long it take for a command. “time ls” to time the ls command | time\_command.png |
| 25 | ul | “man who | ul -tdumb > man.txt” to display underline text in man who correctly. | ul\_command.png |
| 26 | uniq | Filers out repeated adjacent line. “uniq myexamfile.txt” to test this | uniq\_command.png |
| 27 | whoami | Display the user | whoami\_command.png |

**Part II : 30pts**

1. For each command tryout at least one example provided in **Chapter 4** of the Unix textbook. Feel free to use your own example. Show the screenshot for each command’s output. Present your output in a tabular form with column 1 as index (1,2,3..), second column as the command, third as the usage, fourth as the screenshot of the output. You can just show a small snapshot for the output -- we do not need the entire screen’s image.

|  |  |  |  |
| --- | --- | --- | --- |
| Index | Command | Usage | Output |
| 1 | chsh | Change the working shell of the terminal | chsh\_command.png |
| 2 | echo | “echo $SHELL” to display the working shell | echo\_command.png |
| 3 | kill | Kill the process before it is completed.  “(sleep 10; echo done) &  ID  kill ID”  So we create a job in the background with id and use kill id to kill the process. | kill\_command.png |
| 4 | nohup | “nohup ping google.com &” to bring the command ping google.com to the background and not interrupting it. The output is stored in nohup.out. We can use “cat hohup.out” to see what’s inside. | nohup\_command.png |
| 5 | ps | “ps” to see the process status information. | ps\_command.png |
| 6 | sleep | “sleep 10; echo done” wait for 10 seconds then terminate echo done process. | sleep\_command.png |

**Part III : 30pts**

1. For each command tryout at least one example provided in **Chapter 5** of the Unix textbook. Feel free to use your own example. Show the screenshot for each command’s output. Present your output in a tabular form with column 1 as index (1,2,3..), second column as the command, third as the usage, fourth as the screenshot of the output. You can just show a small snapshot for the output -- we do not need the entire screen’s image.

|  |  |  |  |
| --- | --- | --- | --- |
| Index | Command | Usage | Output |
| 1 | expr | “$ x= 1  $x =`expr $x +1`  $echo $x”  expr says that this is a mathematical expression, so the value of x gets evaluate to 2. | expr\_command.png |
| 2 | test | “test 10 -gt 9 && echo “Yes” || echo “No” ”  this command will test if 10 is greater than 9 and print Yes if true or No if false. In this case it will print Yes. | test\_command.png |