

COMP30640: Operating Systems – Finbar Allan

Assignment 3

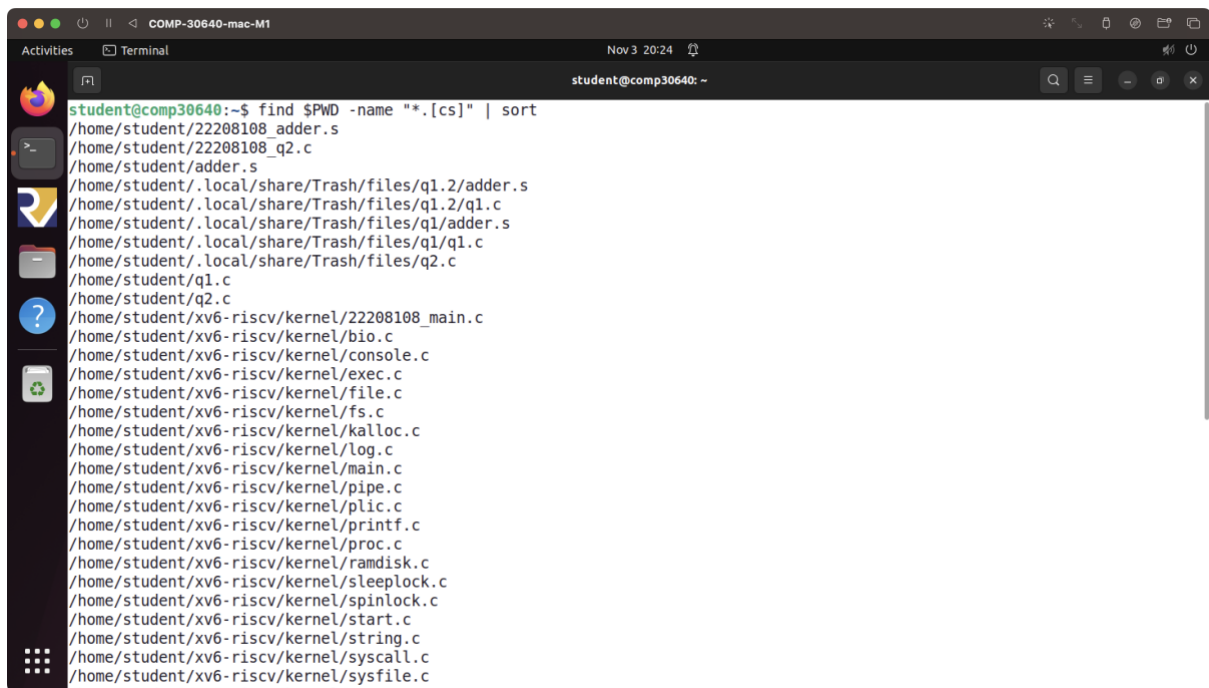
Question 1

The bash command string required to print an alphabetically sorted list of full path names for particular file types in specified directories is formed from several components beginning with the `find` command which locates files based on user-specified criteria. The `$PWD` shell variable gives the instruction to print the full name of the working directory and is given as the first argument to the `find` command.

The `-name` argument filters the files returned by the file command to only include those which conform to the pattern specified in the following expression. The file name for our example can include any string of characters which is represented with the `*` symbol, this string of characters must be followed with a period to denote the beginning of the file extension. After the period the argument requires an expression to denote the acceptance of both `.c` and `.s` file types; this is achieved by enclosing the correct file extensions in square brackets. The files are sorted alphabetically by default with the `sort` command which is pipelined to the `find` command and its arguments giving the following expression:

```
find $PWD -name "*.c" | sort
```

The bash command string given as input to the terminal will produce the output displayed in figure 1. It should be noted that this output does not include an exhaustive list of all files returned by the command, it instead shows only the amount of files which fit on the screen of the virtual machine.



```
student@comp30640:~$ find $PWD -name "*.c" | sort
/home/student/22208108_adder.s
/home/student/22208108_q2.c
/home/student/adder.s
/home/student/.local/share/Trash/files/q1.2/adder.s
/home/student/.local/share/Trash/files/q1.2/q1.c
/home/student/.local/share/Trash/files/q1/adder.s
/home/student/.local/share/Trash/files/q1/q1.c
/home/student/.local/share/Trash/files/q2.c
/home/student/q1.c
/home/student/q2.c
/home/student/xv6-riscv/kernel/22208108_main.c
/home/student/xv6-riscv/kernel/bio.c
/home/student/xv6-riscv/kernel/console.c
/home/student/xv6-riscv/kernel/exec.c
/home/student/xv6-riscv/kernel/file.c
/home/student/xv6-riscv/kernel/fs.c
/home/student/xv6-riscv/kernel/kalloc.c
/home/student/xv6-riscv/kernel/log.c
/home/student/xv6-riscv/kernel/main.c
/home/student/xv6-riscv/kernel/pipe.c
/home/student/xv6-riscv/kernel/plic.c
/home/student/xv6-riscv/kernel/printf.c
/home/student/xv6-riscv/kernel/proc.c
/home/student/xv6-riscv/kernel/ramdisk.c
/home/student/xv6-riscv/kernel/sleeplock.c
/home/student/xv6-riscv/kernel/spinlock.c
/home/student/xv6-riscv/kernel/start.c
/home/student/xv6-riscv/kernel/string.c
/home/student/xv6-riscv/kernel/syscall.c
/home/student/xv6-riscv/kernel/sysfile.c
```

Figure 1: Terminal input and output for the bash command string detailed for Question 1.

Question 2

The number of user accounts which exist on a Linux system corresponds to the number of users in the `/etc/passwd` file and as each users information is stored on a newline in this file the number of newlines will also correspond to the number of user accounts existing on the virtual machine. The bash command string required to count the number of lines in a file begins with the `cat` command which concatenates files and prints the results allowing us to extract data from the file's contents. The `wc` command takes `-l` as its argument giving the instruction to count the number of newline characters instead of the default standard input. Linking the `cat` command and `wc` command with a pipeline gives the following expression which will return the number of newline characters in the target file:

```
cat /etc/passwd | wc -l
```

Figure 2 displays the result of the above bash command string when entered in the terminal which corresponds to the number of newline characters in the `/etc/passwd` file. From this output it can be concluded that 51 user accounts exist on the virtual machines system.

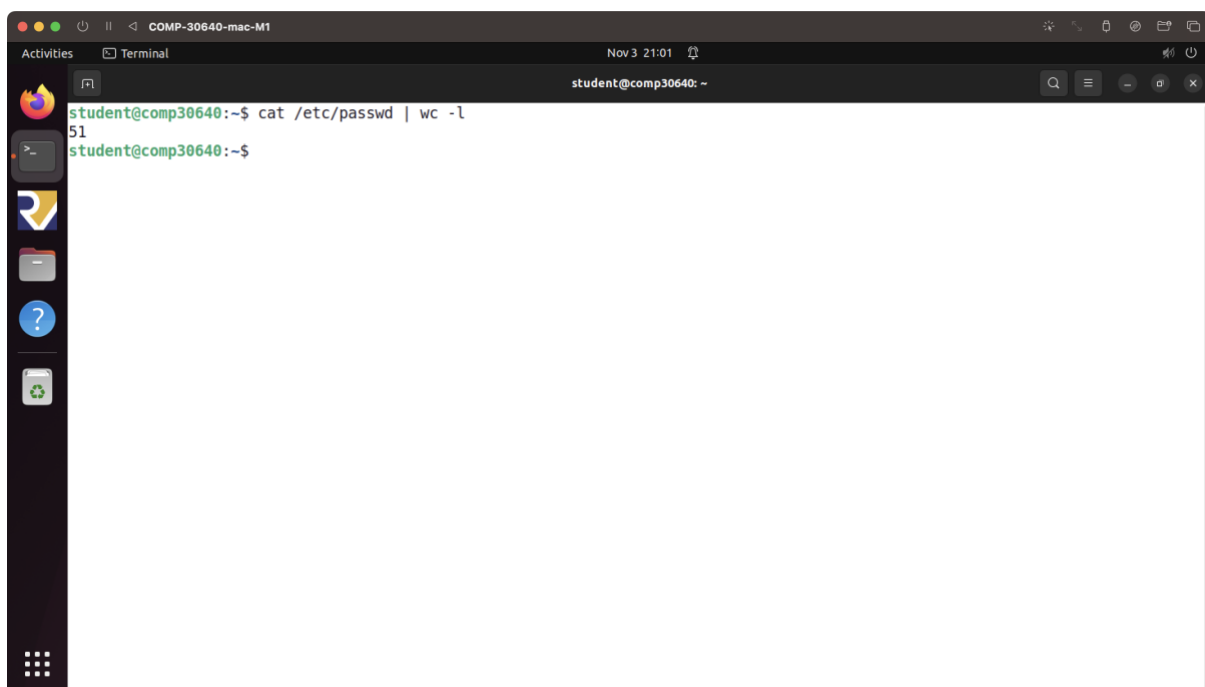
A screenshot of a terminal window on a macOS system. The window title is 'COMP-30640-mac-M1'. The terminal shows the command 'cat /etc/passwd | wc -l' being entered, followed by the output '51'. The prompt 'student@comp30640: ~\$' is visible. The terminal is part of an 'Activities' window, and the dock on the left shows various application icons including Firefox, VS Code, and Finder.

Figure 2: Terminal input and output for the bash command string detailed for Question 2.

Question 3

To write a bash command which prints information associated with the process consuming the most memory on our virtual machine a series of three commands are required to access the information and filter the results. The expression begins with the `ps` command which shows processes belonging to and associated with the same terminal as the invoker by default. This command is then manipulated with the `-e` and `-l` arguments which select all processes on the system and show additional process information in longform respectively. The argument expression can be simplified by denoting the `-e` and `-l` arguments in their shorthand form of `-el`.

Before filtering the results to return only the processes consuming the most memory on our virtual machine it is necessary to sort the results of the `ps` command to print the target process in a predictable location. The `--sort=%mem` expression is implemented as the key to sort the processes by memory consumption from highest to lowest. As the first row of the returned results will always be the heading of the `ps` command's system information table and the second row will contain the process with the highest memory consumption on the virtual machine we can begin filtering to print information as detailed in the assignment description. The `head` command is paired with the `-n` argument to return the first `n` rows of the printed results, similarly, the `tail` command is paired with the `-n` argument to return the last `n` rows of the printed results. The three commands described above can be pipelined together with the appropriate filtering arguments to give the following expression:

```
ps -el --sort=%mem | head -2 | tail -1
```

Information about the process consuming the most memory on our virtual machine is displayed as output in figure 3 from the result of our bash command input expression.



The screenshot shows a terminal window titled "COMP-30640-mac-M1" with the date and time "Nov 3 22:17". The prompt is "student@comp30640: ~". The command entered is `ps -el --sort=%mem | head -2 | tail -1`. The output is a single line of process information: `0 S 1000 1587 1470 1 80 0 - 1352355 futex_? 00:02:20 gnome-shell`. The terminal window has a dark background and a sidebar on the left with various application icons.

Figure 3: Terminal input and output for the bash command string detailed for Question 3.

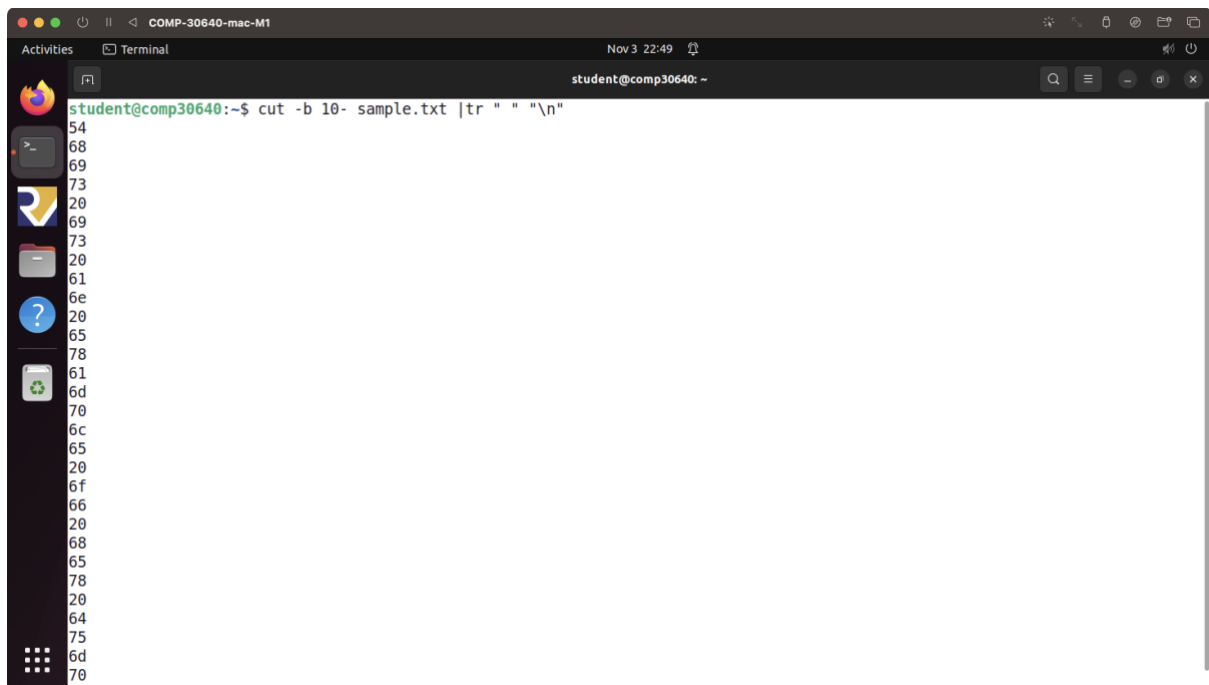
Question 4

Constructing a pipeline with the `cut` and `tr` bash commands to discard and reformat bytes from the `sample.txt` input file requires a single occurrence of each command with a series of arguments to manipulate the results to match the required output per the assignment instructions. The first argument of the `cut` command is `-b`, which allows us to specify the bytes to be printed; this argument is followed by `10-` which specifies the contents of the line from the tenth byte to the end of the line should be printed (denoted by the absence of a character after the dash). The `cut` command is completed by specifying the target file, `sample.txt`.

The `tr` command is implemented to translate all occurrences of the space character, specified as the first argument, with the newline character, specified as the second argument. The bash command string required to print the reformatted bytes in a single column is constructed by pipelining these commands to form the following expression:

```
cut -b 10- sample.txt |tr " " "\n"
```

The above expression produces the output displayed in figure 4 when given as input to the terminal. The hex dump of data which produced the single column of bytes has been recreated from the `sample.txt` file available on Brightspace as opposed to downloaded to the virtual machine directly. This is as a result of my virtual machine losing internet access during the completion of this assignment, hence, any errors in recreating the `sample.txt` file or any mismatches between the bash command input strings in this report and in the screenshots are most likely the result of a typo and information from the virtual machine should take priority.



```
student@comp30640:~$ cut -b 10- sample.txt |tr " " "\n"
54
68
69
73
20
69
73
20
61
6e
20
65
78
61
6d
70
6c
65
20
6f
66
20
68
65
78
20
64
75
6d
70
```

Figure 4: Terminal input and output for the bash command string detailed for Question 4.