CSc 3320: Systems Programming

Fall 2021

Midterm 1: Total points = 100

Submission instructions:

1. Create a Google doc for your 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 TWO POINTS WILL BE DEDUCTED.
4. Keep this page 1 intact. If this *submissions instructions* page is missing in your submission TWO POINTS WILL BE DEDUCTED.
5. Start your responses to each QUESTION on a new page.
6. If you are being asked to write code copy the code into a separate txt file and submit that as well. The code should be executable. E.g. if asked for a C program then provide myfile.c so that we can execute that script. In your answer to the specific question, provide the steps on how to execute your file (like a ReadMe).
7. If you are being asked to test code or run specific commands or scripts, provide the evidence of your outputs through a screenshot and/or screen video-recordings and copy the same into the document.
8. Upon completion, download a .PDF version of the google doc document and submit the same along with all the supplementary files (videos, pictures, scripts etc).
9. Scripts/Code without proper comments, indentation and titles (must have the name of the program, and name & email of the programmer on top the script).

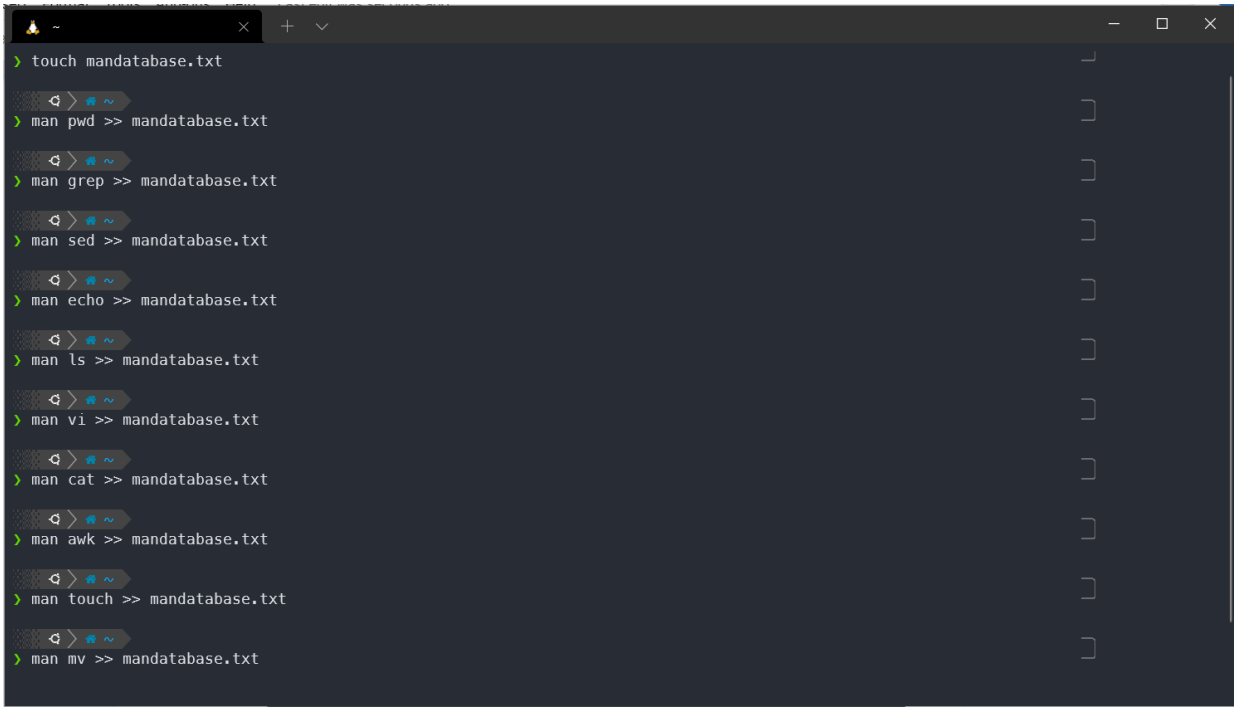
Full Name: Paul Ofremu Jr.

Campus ID: pofremu1

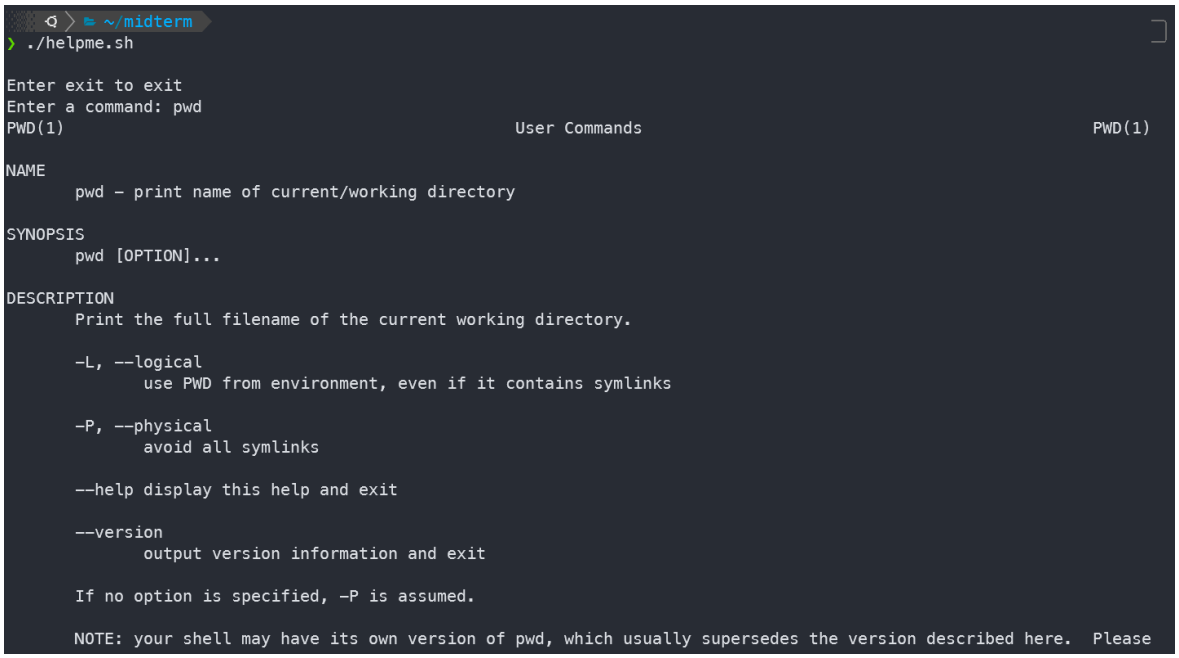
Panther #: 002513676

1. (20 pts) Pick any of your 10 favourite unix commands. For each command run the *man* command and copy the text that is printed into a mandatabase.txt. Write a shell script *helpme.sh* that will ask the user to type in a command and then print the manual's text associated with that corresponding command. If the command the user types is not in the database then the script must print

   sorry, I cannot help you



To run **helpme.sh** the user needs execution permission. You can do that with **chmod u+x helpme.sh**. The "mandatabase.txt" file must be created before running the file. Run the file with **./helpme.sh**.



```bash
#!/bin/bash
# Print manual for 10 favorite commands
# Paul Ofremu Jr., pofremu1@student.gsu.edu

mandatabase="mandatabase.txt"

while true; do
    echo ""
    # Get user input
    echo "Enter "exit" to exit"
    read -p "Enter a command: " command

    if [ $command = "exit" ]
    then
        exit 0
    fi

    # Make user input all uppercase
    command=$(echo "$command" | tr [a-z] [A-Z])

    # Check for the commandname(1) at the beginning and end of a line
    if [ ! -z "$(sed -n "/^$command(1)/,/$command(1)$/p" "$mandatabase")" ]
    then
        # Print everything in between first line and last line of command
        sed -n "/^$command(1)/,/$command(1)$/p" "$mandatabase"
    else
        # Command is not found
        echo "Sorry , I cannot help you"
    fi
```

2. (10pts each) On your computer open your favourite Wikipedia page. Copy the text from that page into a text file **myexamfile.txt** and then copy that file to a directory named **midterm** (use mkdir to create the directory if it doesn't exist) in your snowball server home directory (use any FTP tool such as Putty or Filezilla to copy the file from your computer to the remote snowball server machine: see Lab 6).

a. Write a shell script that will find the number of statements in the text. A statement is defined as the collection of text between two periods (full-stops).

```
#!/bin/bash
# Count # of Statements
# Paul Ofremu Jr., pofremu1@student.gsu.edu

myexamfile="myexamfile.txt"

count=$(grep -oc ".*\." "$myexamfile")
echo "There are $count statements in the text!"
echo ""
```

```
~/midterm
❯ ./getStatements.sh
There are 43 statements in the text!
```

b. Update the script to present a tabular list that shows the number of words and number of letters in each statement.

```
~/midterm
❯ ./getStatements.sh
There are 43 statements in the text!

Statement: 1 | # of words: 10 | # of letters: 60
 Statement: 2 | # of words: 7 | # of letters: 52
 Statement: 3 | # of words: 7 | # of letters: 34
 Statement: 4 | # of words: 13 | # of letters: 109
 Statement: 5 | # of words: 8 | # of letters: 39
 Statement: 6 | # of words: 55 | # of letters: 359
 Statement: 7 | # of words: 32 | # of letters: 182
 Statement: 8 | # of words: 42 | # of letters: 296
 Statement: 9 | # of words: 74 | # of letters: 501
 Statement: 10 | # of words: 80 | # of letters: 481
 Statement: 11 | # of words: 74 | # of letters: 486
 Statement: 12 | # of words: 6 | # of letters: 38
 Statement: 13 | # of words: 77 | # of letters: 455
 Statement: 14 | # of words: 82 | # of letters: 460
 Statement: 15 | # of words: 42 | # of letters: 251
 Statement: 16 | # of words: 64 | # of letters: 351
 Statement: 17 | # of words: 25 | # of letters: 176
 Statement: 18 | # of words: 56 | # of letters: 330
 Statement: 19 | # of words: 12 | # of letters: 77
 Statement: 20 | # of words: 78 | # of letters: 555
 Statement: 21 | # of words: 50 | # of letters: 360
 Statement: 22 | # of words: 46 | # of letters: 336
 Statement: 23 | # of words: 23 | # of letters: 163
 Statement: 24 | # of words: 24 | # of letters: 174
 Statement: 25 | # of words: 30 | # of letters: 221
 Statement: 26 | # of words: 9 | # of letters: 81
```

```
#!/bin/bash
# Count # of Statements
# Paul Ofremu Jr., pofremu1@student.gsu.edu

myexamfile="myexamfile.txt"

count=$(grep -oc ".*\." "$myexamfile")
echo "There are $count statements in the text!"
echo ""

statements=$(grep -o ".*\." "$myexamfile")
#echo $statements
echo $(grep -o ".*\." "$myexamfile" | awk '{print "Statement: " NR " | # of words: " NF " | # of letters: " length "
 \\n"}')
~
```

To run **getStatements.sh** the user needs execution permission. You can do that with **chmod u+x getStatements.sh**. The “myexamfile.txt” file must be created before running the file. Run the file with **./getStatements.sh**.

3. (20pts) Design a calculator using a shell script using regular expressions. The calculator, at the minimum, must be able to process addition, subtraction, multiplication, division and modulo operations. It must also have cancel and clear features.

```
#!/bin/bash
# Calculator
# Paul Ofremu Jr., pofremu1@student.gsu.edu

menu ()
{
echo "===========CALCULATOR==========="
echo "Enter expressions without spaces"
echo "Enter \"clear\" to clear screen or \"cancel\" to exit"
echo ""
}

menu
while true; do
    # Get user input
    echo "Enter an expression:"
    read expr


    case "$expr" in
        "cancel")
            exit
            ;;

        "clear")
            clear
            menu
            ;;

        *)
            # If input matches regex for valid expression evaluate with bc
            [[ $expr =~ ([0-9]+)([/%*+-])([0-9]+) ]] && echo "Answer: " && echo "$expr" | bc || echo "Invalid Expression."
            echo ""
            ;;
    esac
done
```

```
~/midterm
❯ ./calc.sh
===========CALCULATOR===========
Enter expressions without spaces
Enter "clear" to clear screen or "cancel" to exit

Enter an expression:
2+2
Answer:
4

Enter an expression:
10/2
Answer:
5

Enter an expression:
13%3
Answer:
1

Enter an expression:
20-6
Answer:
14

Enter an expression:
cancel

~/midterm
❯
```

4. (20pts) Build a phone-book utility that allows you to access and modify an alphabetical list of names, addresses and telephone numbers. Use utilities such as awk and sed, to maintain and edit the file of phone-book information. The user (in this case, you) must be able to read, edit, and delete the phone book contents. The permissions for the phone book database must be such that it is inaccessible to anybody other than you (the user).

```
3)   Delete Contact
4)   Edit Contact
5)   Find Contact
6)   Exit

Enter option >> 3

Enter exit to exit

Enter first name: Sarah
Enter last name: Johnson
Contact deleted

+=====================+
| PHONE-BOOK UTILITY  |
+=====================+

1)   Display Contacts
2)   Add Contact
3)   Delete Contact
4)   Edit Contact
5)   Find Contact
6)   Exit

Enter option >> 1

----------CONTACTS----------
John|Smith|1346366246|Boston
Paul|Ofremu|1123573421|Atlanta, GA
```

5. (4 pts each) Give brief answers with examples, wherever relevant

A. What is the use of a shell? A shell is an interface between a user and the operating system. A shell gives the user the ability to take advantage of the basic operations of the operating system such as file management, process management and batch processing.

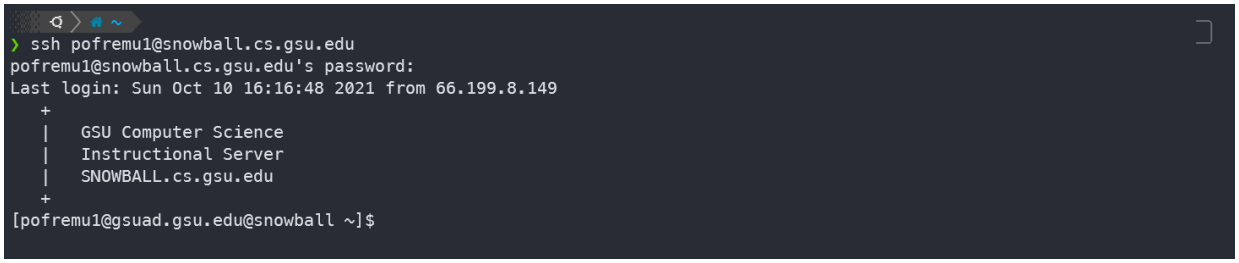
B. Is there any difference between the shell that you see on your PC versus that you see on the snowball server upon login. If yes, what are they? Provide screenshots for examples. The shell on my PC is a command-line shell while the shell on the snowball server is a bash shell.

C. What are the elements in a computer (software and hardware) that enable the understanding and interpretation of a C program? C is a compiled language so the source code is handed to a compiler which converts the source code into object code, which is machine language. This is then passed to linker to add any additional code needed for the code to execute. The output is code that the computer can understand and be executed by the CPU

D. The “printf()” C command is used for printing anything on the screen. In bash we use the command “echo ”. What is the difference (if any) in terms of how the computer interprets

and executes these commands? The “echo” command takes one argument and prints that argument to the screen followed by a new line. The “printf()” command in C takes multiple arguments, the first one is the string format to print to screen, and the remaining arguments are the arguments to replace the specifiers.

E. What do these shell commands do? “ssh”, “scp” and “wget”. Describe briefly using an example that you have executed using the snowball server. The “ssh” command allows you to securely connect to a remote computer or server and gain access to its files, the terminal and other applications. Ex: Connecting to the GSU Snowball server through ssh:



The “scp” command allows you to copy files and directories between servers or computers. (Local host to remote or remote to another remote computer). Ex: Copying myName directory to local machine:

The “wget” command is used to download files from the web. You can download files using the HTTP, HTTPS, and FTP protocols. Ex: Download Readme.md file from my github repo: