



OPERATING SYSTEM LAB ASSIGNMENTS

Name: SHRUTI PATHAK

Roll no: 002210503021

Dept: Computer Science and Engineering

Class: MCA 1st year 2nd sem

Session: 2022-2024

ASSIGNMENT 1:

PROBLEM - 1:

Write a shell script which accepts length and breadth of a rectangle and calculates the area and perimeter of the rectangle.

SOURCE CODE:

```
echo "Enter the length of the rectangle:"
read length
echo $length

echo -e "Enter the breadth of the rectangle:"
read breadth
echo $breadth

area=`expr $length \* $breadth`
echo -n "\narea of the rectangle: $area"
temp=`expr $length + $breadth`
perimeter=`expr 2 \* $temp`
echo -n "\nperimeter of the rectangle: $perimeter"
```

output:

```
user1@sumit-HP-Pro-3330-MT:~/MCA_Shruti_21/ASS1$ sh ass1_01.sh
Enter the length of the rectangle:
4
4
Enter the breadth of the rectangle:
5
5

area of the rectangle: 20

perimeter of the rectangle: 18
```

PROBLEM - 2:

Write a shell script which accepts basic salary of an employee and calculates net salary and displays the salary slip.

SOURCE CODE:

```
echo "Enter the basic salary:"
read basic

da=`expr $basic \* 30 / 100`
```

```
echo "\nDearness allowance: $da"

hra=`expr $basic \* 25 / 100`
echo "\nHouse rent allowance: $hra"

net=`expr $basic + $da + $hra`
echo "\nNet payment: $net"
```

output:

```
user1@sumit-HP-Pro-3330-MT:~/MCA_Shruti_21/ASS1$ sh ass1_02.sh
Enter the basic salary:
10000

Dearness allowance: 3000

House rent allowance: 2500

Net payment: 15500
```

PROBLEM - 3:

Write a shell script which accepts a five digit number and prints sum of its digits.

SOURCE CODE:

```
echo "Enter the five digit number: "
read number

temp=$number
sum=0

while [ $number -ne 0 ]
do

    sum=`expr $sum + \( $number % 10 \)`
    number=`expr $number / 10`
done
echo "\n Sum of the digits of $temp: $sum"
```

output:

```
user1@sumit-HP-Pro-3330-MT:~/MCA_Shruti_21/ASS1$ sh ass1_03.sh
Enter the five digit number:
12345

Sum of the digits of 12345: 15
```

PROBLEM - 4:

Write a shell script which accepts a five digit number and prints the reverse number.

SOURCE CODE:

```
echo "Enter the five digit number: "
read number

temp=$number
rev=0

while [ $number -ne 0 ]
do
    rev=`expr \($rev \* 10\) + \($number % 10\)`
    number=`expr $number / 10`
done
echo "\nReverse of the digits of $temp : $rev"
```

output:

```
user1@sumit-HP-Pro-3330-MT:~/MCA_Shruti_21/ASS1$ sh ass1_04.sh
Enter the five digit number:
12345
```

Reverse of the digits of 12345 : 54321

PROBLEM - 5:

The /etc/passwd file stores user account information. It contains one entry per line for each user (user account) of the system. Each line contains seven fields which are separated by a colon (:) symbol. The fields are:

- (i) Username
- (ii) Password
- (iii) User Id
- (iv) Group Id
- (v) User Id Info
- (vi) Home Directory
- (vii) Login Shell

Write a shell script which accepts a user login name and displays detail information about the users as available from the file /etc/passwd.

SOURCE CODE:

```
while true
do
    echo -n "Enter username: "
```

```

read user
i=`grep -w "$user" /etc/passwd`
if [ -z "$i" ]
then
    echo -n "Not found."
else
    break
fi
done
echo -n "Username: "
grep $user /etc/passwd | cut -d ":" -f1
echo -n "Password: "
grep $user /etc/passwd | cut -d ":" -f2
echo -n "User_Id: "
grep $user /etc/passwd | cut -d ":" -f3
echo -n "Group_Id: "
grep $user /etc/passwd | cut -d ":" -f4
echo -n "User_Id Info: "
grep $user /etc/passwd | cut -d ":" -f5
echo -n "Home Directory: "
grep $user /etc/passwd | cut -d ":" -f6
echo -n "Login Shell: "
grep $user /etc/passwd | cut -d ":" -f7

```

output:

```

user1@sumit-HP-Pro-3330-MT:~/MCA_Shruti_21/ASS1$ sh ass1_05.sh
Enter username: user1
Username: user1
Password: x
User_Id: 1001
Group_Id: 1001
User_Id Info: USER1,,,
Home Directory: /home/user1
Login Shell: /bin/bash
user1@sumit-HP-Pro-3330-MT:~/MCA_Shruti_21/ASS1$ sh ass1_05.sh
Enter username: user2
Not found.Enter username:

```

ASSIGNMENT 2:

PROBLEM - 1:

Write a shell script which, for all files in present directory displays whether it is a regular file or a directory.

SOURCE CODE:

```
for file in *
do
    if [ -f "$file" ]
    then
        echo "$file is a regular file."
    elif [ -d "$file" ]
    then
        echo "$file is a directory."
    fi
done
```

output:

```
user1@sumit-HP-Pro-3330-MT:~/MCA_Shruti_21/ASS2$ sh ass2_01.sh
abc is a directory.
ass2_01.sh is a regular file.
ass2_02.sh is a regular file.
ass2_03.sh is a regular file.
ass2_04.sh is a regular file.
ass2_05.sh is a regular file.
```

PROBLEM - 2:

The PATH variable is an environment variable that contains an ordered list of paths that Linux will search for executables when running a command. Write a shell script to display all the directories in the PATH variable in a simple way, i.e., one line per directory. In addition, display information about each directory, such as the permissions and the modification times.

SOURCE CODE:

```
IFS=:
for dir in $PATH
do
    if [ -d $dir ]
    then
        echo "$dir"
        echo "permission : $(ls -ld $dir | awk '{print $1}')"
        echo "modification time : $(date -r $dir)"
    else

```

```
    echo "$dir does not exit"
fi
done
```

output:

```
user1@sumit-HP-Pro-3330-MT:~/MCA_Shruti_21/ASS2$ sh ass2_02.sh
/home/user1/bin does not exit
/home/user1/.local/bin does not exit
/usr/local/sbin
permission : drwxr-xr-x
modification time : Fri Aug 7 04:07:49 IST 2020
/usr/local/bin
permission : drwxr-xr-x
modification time : Fri Aug 7 04:07:49 IST 2020
/usr/sbin
permission : drwxr-xr-x
modification time : Thu Dec 8 11:19:35 IST 2022
/usr/bin
permission : drwxr-xr-x
modification time : Thu Dec 8 11:20:19 IST 2022
/sbin
permission : drwxr-xr-x
modification time : Thu Dec 8 11:13:49 IST 2022
/bin
permission : drwxr-xr-x
modification time : Thu Dec 8 11:11:09 IST 2022
/usr/games
permission : drwxr-xr-x
modification time : Fri Aug 7 04:15:08 IST 2020
/usr/local/games
permission : drwxr-xr-x
modification time : Fri Aug 7 04:07:49 IST 2020
/snap/bin does not exit
```

PROBLEM - 3:

Write a shell script which displays vendor id, model name, cpu MHz,cache size information about the processor present in your computer. Hint: most of this information can be obtained by reading the file/proc/cpuinfo.

SOURCE CODE:

```
#!/bin/bash

cpuinfo_file="/proc/cpuinfo"

vendor=$(grep -m 1 "vendor_id" $cpuinfo_file | awk '{print $3}')
model=$(grep -m 1 "model name" $cpuinfo_file | awk -F ':' '{print $2}')
```

```
cpumhz=$(grep -m 1 "cpu MHz" $cpuinfo_file | awk '{print $4}')
cache=$(grep -m 1 "cache size" $cpuinfo_file | awk '{print $4}')

echo "Vendor ID: $vendor"
echo "Model Name: $model"
echo "CPU MHz: $cpumhz"
echo "Cache Size: $cache"
```

output:

```
user1@sumit-HP-Pro-3330-MT:~/MCA_Shruti_21/ASS2$ sh ass2_03.sh
Vendor ID: GenuineIntel
Model Name: Intel(R) Core(TM) i3-3220 CPU @ 3.30GHz
CPU MHz: 1636.783
Cache Size: 3072
```

PROBLEM - 4:

Write a shell script to show your home directory, Operating System type, version, release number, kernel version and current path setting. Hint: use uname command or use content of /proc/sys/kernel/osrelease file.

SOURCE CODE:

```
echo "Home directory : $HOME"

os=$(uname -o)
echo "Operating System type is - $os"

version=$(uname -v | cut -d ' ' -f 1 | cut -c 6-)
echo "Operating System version is - $version"

release=$(uname -v)
echo "Operating System release is - $release"

kernel=$(uname -r)
echo "kernel version - $kernel"

echo "Current path setting $PATH"
```

output:

```
user1@sumit-HP-Pro-3330-MT:~/MCA_Shruti_21/ASS2$ sh ass2_04.sh
Home directory : /home/user1
Operating System type is - GNU/Linux
Operating System version is - 16.04.1-Ubuntu
```


Operating System release is - #146~16.04.1-Ubuntu SMP Tue Apr 13 09:27:15 UTC 2021

kernel version - 4.15.0-142-generic

Current path setting

/home/user1/bin:/home/user1/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin

PROBLEM - 5:

Write a shell script to display a summary of the disk space usage foreach directory argument (and any subdirectories), both in terms of bytes, and kilobytes or megabytes (whichever is appropriate). [du -b]

SOURCE CODE:

```
if [ $# -eq 0 ]; then
    echo "No argument given"
    exit 1
fi
for dir in "$@"; do
    if [ -d "$dir" ]; then
        echo "$dir"
        echo "Disk space usage in bytes:"
        du -b "$dir"
        echo "Disk space usage in kilobytes or megabytes:"
        du -h "$dir"
    else
        echo "$dir is not a valid directory"
    fi
    echo "Total size of the directory is "
    echo "$(du -sh "$dir")\n"
done
```

output:

user1@sumit-HP-Pro-3330-MT:~/MCA_Shruti_21/ASS2\$ sh ass2_05.sh

No argument given

user1@sumit-HP-Pro-3330-MT:~/MCA_Shruti_21/ASS2\$ sh ass2_05.sh abc

abc

Disk space usage in bytes:

4096 abc

Disk space usage in kilobytes or megabytes:

4.0K abc

Total size of the directory is

4.0K abc

ASSIGNMENT 3:

PROBLEM - 1:

Write a shell script which reads a input file that contains three integers in each line. The script should display the sum of all integers in each line.

SOURCE CODE:

```
files="num.txt"
while read -r line
do
    sum=0
    for num in $line
    do
        sum=$(( sum + num ))
    done
    echo "Sum is $sum"
done <$files
```

output:

contents of the "num.txt" file:

12 13 15

5 7 9

10 20 30

user1@sumit-HP-Pro-3330-MT:~/MCA_Shruti_21/ASS3\$ sh ass3_01.sh

Sum is 40

Sum is 21

Sum is 60

PROBLEM - 2:

Write a shell script to find out how many file and directory are there in the current directory. Also list the file and directory names separately.

SOURCE CODE:

```
count=0
for file in *
do
    count=$(( count + 1 ))
done
echo "Total count of files and directory is $count"
for file in *
do
    if [ -f $file ]
    then
        echo "$file is a regular file"
    elif [ -d $file ]
    then
        echo "$file is a directory"
```

fi
done

output:

```
user1@sumit-HP-Pro-3330-MT:~/MCA_Shruti_21/ASS3$ sh ass3_02.sh
Total count of files and directory is 7
ass3_01.sh is a regular file
ass3_02.sh is a regular file
ass3_03.sh is a regular file
ass3_04.sh is a regular file
ass3_05.sh is a regular file
ass3_06.sh is a regular file
num.txt is a regular file
```

PROBLEM - 3:

Write a script that adds up the sizes reported by the ls command for the files in the current directory. The script should print out only the total number of bytes used.

SOURCE CODE:

```
#!/bin/bash

echo "Total bytes used: "$(ls -lA | awk '{ total += $5 } END { print total}')
```

output:

```
user1@sumit-HP-Pro-3330-MT:~/MCA_Shruti_21/ASS3$ sh ass3_03.sh
Total bytes used: 1052
```

PROBLEM - 4:

Write a shell script that delete all temporary files (end with ~) in current directory.

SOURCE CODE:

```
#!/bin/bash

echo "Files before delete :"
ls -p | grep -v /

find . -name "*~" -type f -delete
echo "Deleted files ending with ~"

echo "Files after delete :"
ls -p | grep -v /
```

output:

```
user1@sumit-HP-Pro-3330-MT:~/MCA_Shruti_21/ASS3$ sh ass3_04.sh
```

Files before delete :

abc.txt~

ass3_01.sh

ass3_02.sh

ass3_03.sh

ass3_04.sh

ass3_05.sh

ass3_06.sh

num.txt

Deleted files ending with ~

Files after delete :

ass3_01.sh

ass3_02.sh

ass3_03.sh

ass3_04.sh

ass3_05.sh

ass3_06.sh

num.txt

PROBLEM - 5:

Write a shell script to rename file having extension .sh to .exe.

SOURCE CODE:

```
#!/bin/bash
```

```
for file in *.sh; do
    mv -- "$file" "${file%.sh}.exe"
done
```

```
# Print a message indicating how many files were renamed
num_renamed=$(find . -name "*.exe" -type f | wc -l)
echo "Renamed $num_renamed files from .sh to .exe"
```

output:

```
user1@sumit-HP-Pro-3330-MT:~/MCA_Shruti_21/ASS3$ sh ass3_05.sh
```

Renamed 6 files from .sh to .exe

PROBLEM - 6:

Write a shell script to count number of shell scripts (with .sh extension) present in the current directory.

SOURCE CODE:

```
#!/bin/bash
```

```
num_scripts=$(find . -maxdepth 1 -type f -name "*.sh" | wc -l)
```

```
echo "There are $num_scripts shell scripts in the current directory"
```

output:

user1@sumit-HP-Pro-3330-MT:~/MCA_Shruti_21/ASS3\$ ass3_06.sh
There are 6 shell scripts in the current directory

MENU DRIVEN PROGRAM:

SOURCE CODE:

```
while [ true ]
do
    echo "\n0-> exit"
    echo "1-> Assignment 1"
    echo "2-> Assignment 2"
    echo "3-> Assignment 3"

    echo "\nEnter your choice : "
    read choice
    echo "You Choose $choice"
    case $choice in

        1) cd /home/user1/Desktop/SHRUTI_21/ASS1
            echo "You are in Assignment 1"
            echo -n "Enter program number to execute: "
            read num
            name=ass1_0$num.sh
            chmod +x $name
            sh $name
            ;;

        2) cd /home/user1/Desktop/SHRUTI_21/ASS2
            echo "You are in Assignment 2"
            echo -n "Enter program number to execute: "
            read num
            name=ass2_0$num.sh
            chmod +x $name
            sh $name
            ;;

        3) cd /home/user1/Desktop/SHRUTI_21/ASS3
            echo "You are in Assignment 3"
            echo -n "Enter program number to execute: "
            read num
            name=ass3_0$num.sh
            chmod +x $name
            sh $name
            ;;

        0) echo "Quitting...."
            exit 0 ;;
        *) echo "Invalid Choice....";;
    esac
done
```

ASSIGNMENT 4:

PROBLEM - 1:

Write a C program to create a child process. The parent process must wait until the child finishes. Both the processes must print their own pid and parent pid. Additionally the parent process should print the exit status of the child.

SOURCE CODE:

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/wait.h>
int main(){
    pid_t pid = fork();
    if(pid == -1){
        printf("Fork failed");
        exit(EXIT_FAILURE);
        0;
    }
    if(pid == 0){
        printf("Child process - PID: %d, parent PID: %d\n", getpid(), getppid());
        exit(EXIT_SUCCESS);
    }else{
        printf("Parent process - PID %d, Parent PID: %d\n", getpid(), getppid());
        int status;
        waitpid(pid, &status, 0);
        printf("Child process exited with status: %d\n", WEXITSTATUS(status));
    }

    return 0;
}
```

output:

```
user1@sumit-HP-Pro-3330-MT:~/MCA_Shruti_21/ASS4$ gcc ass4_01.c
user1@sumit-HP-Pro-3330-MT:~/MCA_Shruti_21/ASS4$ ./a.out
Parent process - PID 3229, Parent PID: 3209
Child process - PID: 3230, parent PID: 3229
Child process exited with status: 0
```

PROBLEM - 2:

Write a C program which prints prime numbers between the range 1 to 10,00,000 by creating ten child processes and subdividing the task equally among all child processes, i.e., the first child should print prime numbers in the range 1 to 1,00,000, the second child in the range 1,00,001 to 2,00,000, ... The child processes must run in parallel and the parent process must wait until all the child processes finish.

SOURCE CODE:

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/wait.h>
#include<stdlib.h>
//function to check whether a number is prime or not
int isPrime(int num){
    int i;
    if(num==1||num==0)
        return 0;
    if(num==2||num==3)
        return 1;
    if(num%2==0)
        return 0;
    //loop to check whether any odd integer divides num
    for(i=3;i*i<=num;i+=2){
        if(num%i==0)
            return 0;
    }
    return 1;
}
//function to print primes in a range
void print_prime(int start,int end){
    int i;
    printf("\nPrimes in the range %d to %d are:\n\n",start,end);
    printf("-----\n\n");
    for(i=start;i<=end;i++){
        if(isPrime(i)){
            printf("%d\t",i);
        }
    }
    printf("\n");
}
int main()
{
    int i,start,end,status;
    //loop to create 10 child process and distribute tasks
    for(i=0;i<10;i++){
        //allocate tasks to every child
        if(fork()==0)
        {
            start=100000*i+1;
            end=start+99999;
            print_prime(start,end);
            exit(0); //this exit suspend the child process,so that this cannot further iterate
in loop
        }
        sleep(1);
    }
    return 0;
}
```

output:

```
user1@sumit-HP-Pro-3330-MT:~/MCA_Shruti_21/ASS4$ gcc ass4_02.c
```

```
user1@sumit-HP-Pro-3330-MT:~/MCA_Shruti_21/ASS4$ ./a.out
```

Primes in the range 1 to 100000 are:

```
-----  
2    3    5    7    11    13    17    19    23    29    31    37    41    43  
    47    53    59    61    67    71    73    79    83    89    97    101   103  
    107   109   113   127   131   137   139   149   151   157   163   167   173  
    179   181   191   193   197   199   211   223   227   229   233 -----  
-----  
----- 999553 999563 999599 999611 999613 999623 999631 999653 999667  
999671 999683 999721 999727 999749 999763 999769 999773 999809 999853 999863 999883  
999907 999917 999931 999953 999959 999961 999979 999983
```

PROBLEM - 3:

Write a C program which creates a child process. The parent process sends a string (input by user) which the child process inspects and sends "YES" back to the parent if the string is a palindrome, otherwise it sends "NO". The IPC to be used is pipe. Both the processes terminate when the input string is "quit".

SOURCE CODE:

```
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <unistd.h>  
#include <sys/wait.h>  
  
#define BUFFER_SIZE 64  
int checkPalindrome(char str[]);  
  
int main() {  
  
    char buffer[BUFFER_SIZE];  
  
    int pipe1[2]; // child => parent communication  
    int pipe2[2]; // parent => child communication  
  
    if (pipe(pipe1) == -1) {return 1;}  
    if (pipe(pipe2) == -1) {return 2;}  
  
    pid_t pid = fork();  
  
    if(pid == -1) {return 3;};  
  
    if (pid == 0){  
        //child  
        close(pipe1[0]); // close read from pipe 1
```



```

close(pipe2[1]); // close write from pipe 2

while (1){
    read(pipe2[0], buffer, BUFFER_SIZE);

    if (strcmp(buffer, "quit") == 0) {
        break;
    }

    if(checkPalindrome(buffer)){
        write(pipe1[1], "Yes", 4);
    }else{
        write(pipe1[1], "No", 3);
    }

}
close(pipe1[1]);
close(pipe2[0]);
printf("\nChild processes terminated\n");
exit(20);
}else{
    //parent
    close(pipe1[1]); // close write from pipe 1
    close(pipe2[0]); // close read from pipe 2

    while (1){
        printf("\nEnter string ( for exit 'quit' ) :\n");
        fgets(buffer, BUFFER_SIZE, stdin);
        buffer[strcspn(buffer, "\n")] = '\0'; // make \n and make it null

        if (strcmp(buffer, "quit") == 0) {
            write(pipe2[1], buffer, BUFFER_SIZE);
            break;
        }

        write(pipe2[1], buffer, BUFFER_SIZE);

        read(pipe1[0], buffer, BUFFER_SIZE);
        printf("\nIs palindrome?: %s", buffer);
    }

    close(pipe1[0]);
    close(pipe2[1]);
    wait(NULL);
    printf("\nParent processes terminated\n");
    exit(19);
}

return 0;
}

int checkPalindrome(char str[])
{
    int i, len;
    len = strlen(str);

    for (i = 0; i < len / 2; i++) {
        if (str[i] != str[len - i - 1])

```

```

        return 0;
    }
    return 1;
}

```

output:

```

user1@sumit-HP-Pro-3330-MT:~/MCA_Shruti_21/ASS4$ gcc ass4_03.c
user1@sumit-HP-Pro-3330-MT:~/MCA_Shruti_21/ASS4$ ./a.out

```

Enter string (for exit 'quit') : Shruti

Is palindrome?: No

Enter string (for exit 'quit') :madam

Is palindrome?: Yes

Enter string (for exit 'quit') :quit

Child processes terminated

Parent processes terminated

PROBLEM - 4:

Write a C program which prints the following menu

- 1. ls**
- 2. pwd**
- 3. uname**
- 4. exit**

When, the user provides an input, the parent process creates a child process [if user's choice is between 1-3] and executes the corresponding command [use `execv()` system call]. The main process waits for the child to finish and displays the menu again. The parent process terminates if user's choice is 4.

SOURCE CODE:

```

#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
#include<sys/wait.h>
#include<stdlib.h>

int main(){
    int status,ch;
    do{

```

```

printf("\n1.ls\n2.pwd\n3.uname\n4.Exit\nEnter your choice:");
scanf("%d",&ch);
switch(ch){
    case 1:
        if(fork()==0){
            printf("\n-:ls command:-\n");
            char *str1[]={"/bin/ls","-l",NULL};
            execv("/bin/ls", str1);
        }
        else
            wait(&status);
        break;
    case 2:
        if(fork()==0){
            printf("\npwd command:=>\n");
            char *str2[]={"/bin/pwd",NULL};
            execv("/bin/pwd", str2);
        }
        else
            wait(&status);
        break;
    case 3:
        if(fork()==0){
            printf("\n-:uname command:-\n");
            char *str3[]={"/bin/uname",NULL};
            execv("/bin/uname", str3);
        }
        else
            wait(&status);
        break;
    case 4:
        printf("Exiting program...\n");
        break;
    default:
        printf("Wrong Choice!!!\n");
}
}while(ch!=4);
return 0;
}

```

output:

user1@sumit-HP-Pro-3330-MT:~/MCA_Shruti_21/ASS4\$ gcc ass4_04.c
user1@sumit-HP-Pro-3330-MT:~/MCA_Shruti_21/ASS4\$./a.out

1.ls
2.pwd
3.uname
4.Exit
Enter your choice:1

-.:ls command:-

total 28

```
-rwxrwxr-x 1 user1 user1 8928 Jun  8 16:26 a.out
-rw-r--r-- 1 user1 user1  639 May 22 12:33 ass4_01.c
-rw-r--r-- 1 user1 user1 1063 May 23 02:43 ass4_02.c
-rw-r--r-- 1 user1 user1 1555 May 23 02:43 ass4_03.c
-rw-r--r-- 1 user1 user1 2023 May 23 02:43 ass4_04.c
```

1.ls

2.pwd

3.uname

4.Exit

Enter your choice:2

pwd command:=>

/home/user1/MCA_Shruti_21/ASS4

1.ls

2.pwd

3.uname

4.Exit

Enter your choice:3

-.:uname command:-

Linux

1.ls

2.pwd

3.uname

4.Exit

Enter your choice:4

Exiting program...