## 

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:

1. **Username**

## Password

1. **User Id**

## Group Id

1. **User Id Info**

## Home Directory

1. **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 "permision : $(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

permision : drwxr-xr-x

modification time : Fri Aug 7 04:07:49 IST 2020

/usr/local/bin

permision : drwxr-xr-x

modification time : Fri Aug 7 04:07:49 IST 2020

/usr/sbin

permision : drwxr-xr-x

modification time : Thu Dec 8 11:19:35 IST 2022

/usr/bin

permision : drwxr-xr-x

modification time : Thu Dec 8 11:20:19 IST 2022

/sbin

permision : drwxr-xr-x

modification time : Thu Dec 8 11:13:49 IST 2022

/bin

permision : drwxr-xr-x

modification time : Thu Dec 8 11:11:09 IST 2022

/usr/games

permision : drwxr-xr-x

modification time : Fri Aug 7 04:15:08 IST 2020

/usr/local/games

permision : 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 for each 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 scripts 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 furthur 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' ) :

level

Is palindrome?: Yes

Enter string ( for exit 'quit' ) :

abcd

Is palindrome?: No

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**

## pwd

1. **uname**

## 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...

## ASSIGNMENT 5:

**PROBLEM - 1:**

**Write a C program which creates a child process. The parent and child process communicate using a shared memory segment. The parent process generates 100 random integers and writes it into the shared memory segment. The child process then computes the maximum, minimum and average of all these 100 numbers and writes the result back into the shared memory segment, from where the parent process reads the result and displays it. Add appropriate code to synchronize the parent and child process. [Hint: It is an example of strict alteration where access to the shared memory segment alternates between the parent and child process]**

**SOURCE CODE:**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <sys/wait.h>

#define SHM\_SIZE 100\*sizeof(int)

typedef struct {

    int numbers[100];

    int min;

    int max;

    double avg;

} SharedData;

void parentProcess(int shmId) {

    SharedData\* sharedData = (SharedData\*) shmat(shmId, NULL, 0);

    // Generate random numbers and write them to shared memory

    for (int i = 0; i < 100; i++) {

        int num = rand() % 1000;

        sharedData->numbers[i] = num;

    }

    shmdt(sharedData);

    // Wait for the child process to finish

    wait(NULL);

    // Read the result from shared memory

    sharedData = (SharedData\*) shmat(shmId, NULL, 0);

    // Display the result

    printf("Maximum: %d\n", sharedData->max);

    printf("Minimum: %d\n", sharedData->min);

    printf("Average: %.2f\n", sharedData->avg);

    // Detach and remove shared memory

    shmdt(sharedData);

    shmctl(shmId, IPC\_RMID, NULL);

}

void childProcess(int shmId) {

    SharedData\* sharedData = (SharedData\*) shmat(shmId, NULL, 0);

    int min = sharedData->numbers[0];

    int max = sharedData->numbers[0];

    int sum = 0;

    // Compute the minimum, maximum, and average

    for (int i = 0; i < 100; i++) {

        int num = sharedData->numbers[i];

        if (num < min) {

            min = num;

        }

        if (num > max) {

            max = num;

        }

        sum += num;

    }

    sharedData->min = min;

    sharedData->max = max;

    sharedData->avg = (double) sum / 100;

    shmdt(sharedData);

}

int main() {

    key\_t key = ftok("shared\_memory", 1234);

    int shmId = shmget(key, SHM\_SIZE, IPC\_CREAT | 0666);

    pid\_t pid = fork();

    if (pid < 0) {

        fprintf(stderr, "Fork failed\n");

        return 1;

    } else if (pid == 0) {

        // Child process

        childProcess(shmId);

    } else {

        // Parent process

        parentProcess(shmId);

    }

    return 0;

}

## output:

user1@sumit-HP-Pro-3330-MT:~/MCA\_Shruti\_21/ASS5$ gcc ass5\_01.c

user1@sumit-HP-Pro-3330-MT:~/MCA\_Shruti\_21/ASS5$ ./a.out

Maximum: 996

Minimum: 11

Average: 476.84

**PROBLEM - 2:**

**P1, P2 and P3 are three processes executing their respective tasks. They should synchronize among themselves using semaphores such that the string ”ABCCAB” gets printed 10 times. Write codes for process P1, P2 and P3 to get the desired output. [Hint: Write code for the main process which creates and initializes necessary semaphores and then creates three child processes for executing tasks of process P1, P2 and P3 respectively.]**

**P1**

while ( t r u e ){

p r i n t ( ”A” ) ;

}

**P2**

while ( t r u e ){

p r i n t ( ”B ” ) ;

}

**P3**

while ( t r u e ){

p r i n t ( ”C” ) ;

}

**SOURCE CODE:**

#include <stdio.h>

#include <pthread.h>

#include <semaphore.h>

#include <time.h>

#include <unistd.h>

#include <stdlib.h>

#include <stdbool.h>

sem\_t semA,semB,semC;

void \*printA(void \*arr)

{

    int \*ar =(int \*)arr;

    while(true)

    {

        if(ar[1]==20)

        {

            exit(0);

        }

        sem\_wait(&semA);

        printf("A");

        ar[1]++;

        sem\_post(&semB);

    }

}

void \*printB(void \*arr)

{

    int \*ar =(int \*)arr;

    while(true)

    {

        sem\_wait(&semB);

        printf("B");

        if(ar[0]%2==1)

        {

            ar[0]=0;

            printf("\n");

            sem\_post(&semA);

        }

        else

        {

            sem\_post(&semC);

        }

    }

}

void \*printC(void \*arr)

{

    int \*ar =(int \*)arr;

    while(true)

    {

        sem\_wait(&semC);

        printf("C");

        if((ar[0]%2)==0)

        {

            (ar[0])++;

            sem\_post(&semC);

        }

        else if((ar[0]%2)==1)

            sem\_post(&semA);

    }

}

int main()

{

    pthread\_t TA,TB,TC;

    int arr[2];

    arr[0]=0;

    arr[1]=0;

    sem\_init(&semA,0,1);

    sem\_init(&semB,0,0);

    sem\_init(&semC,0,0);

    pthread\_create(&TA,NULL,printA,(void\*)arr);

    pthread\_create(&TC,NULL,printB,(void\*)arr);

    pthread\_create(&TC,NULL,printC,(void\*)arr);

    pthread\_join(TA,NULL);

    pthread\_join(TB,NULL);

    pthread\_join(TC,NULL);

    return 0;

}

## output:

user1@sumit-HP-Pro-3330-MT:~/MCA\_Shruti\_21/ASS5$ gcc ass5\_02.c

user1@sumit-HP-Pro-3330-MT:~/MCA\_Shruti\_21/ASS5$ ./a.out

ABCCAB

ABCCAB

ABCCAB

ABCCAB

ABCCAB

ABCCAB

ABCCAB

ABCCAB

ABCCAB

ABCCAB

**PROBLEM - 3:**

**Implement the solution to the producer-consumer problem using semaphores.**

**SOURCE CODE:**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>

#define MAX\_BUFFER\_SIZE 5

int buffer[MAX\_BUFFER\_SIZE];

int buffer\_index = 0;

sem\_t buffer\_lock;

sem\_t items\_available;

sem\_t spaces\_available;

void\* producer(void\* arg) {

    int item = 0;

    while (1) {

        // Simulate producing an item

        sleep(1);

        item = rand() % 100;  // Generate a random item

        sem\_wait(&spaces\_available);  // Wait for an available space in the buffer

        sem\_wait(&buffer\_lock);  // Acquire the buffer lock to modify the buffer

        buffer[buffer\_index] = item;  // Add the item to the buffer

        buffer\_index++;

        sem\_post(&buffer\_lock);  // Release the buffer lock

        sem\_post(&items\_available);  // Signal that there's an item available for consumption

        printf("Produced item: %d\n", item);

    }

}

void\* consumer(void\* arg) {

    int item;

    while (1) {

        sem\_wait(&items\_available);  // Wait for an available item in the buffer

        sem\_wait(&buffer\_lock);  // Acquire the buffer lock to modify the buffer

        item = buffer[buffer\_index - 1];  // Consume the last item from the buffer

        buffer\_index--;

        sem\_post(&buffer\_lock);  // Release the buffer lock

        sem\_post(&spaces\_available);  // Signal that there's an available space in the buffer

        printf("Consumed item: %d\n", item);

    }

}

int main() {

    srand(time(NULL));

    sem\_init(&buffer\_lock, 0, 1);

    sem\_init(&items\_available, 0, 0);

    sem\_init(&spaces\_available, 0, MAX\_BUFFER\_SIZE);

    pthread\_t producer\_thread, consumer\_thread;

    pthread\_create(&producer\_thread, NULL, producer, NULL);

    pthread\_create(&consumer\_thread, NULL, consumer, NULL);

    pthread\_join(producer\_thread, NULL);

    pthread\_join(consumer\_thread, NULL);

    sem\_destroy(&buffer\_lock);

    sem\_destroy(&items\_available);

    sem\_destroy(&spaces\_available);

    return 0;

}

## output:

user1@sumit-HP-Pro-3330-MT:~/MCA\_Shruti\_21/ASS5$ gcc ass5\_03.c

user1@sumit-HP-Pro-3330-MT:~/MCA\_Shruti\_21/ASS5$ ./a.out

Produced item: 84

Consumed item: 84

Produced item: 12

Consumed item: 12

Produced item: 28

Consumed item: 28

Produced item: 69

Consumed item: 69

Produced item: 16

Consumed item: 16

Produced item: 67

Consumed item: 67

Produced item: 86

Consumed item: 86

Produced item: 94

Consumed item: 94

Produced item: 76

Consumed item: 76

**PROBLEM - 4:**

**Implement the solution to the Reader-Writers problem using semaphores.**

**SOURCE CODE:**

#include<semaphore.h>

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<pthread.h>

sem\_t x,y;

pthread\_t tid;

pthread\_t writerthreads[100],readerthreads[100];

int readercount = 0;

void \*reader(void\* param)

{

    sem\_wait(&x);

    readercount++;

    if(readercount==1)

        sem\_wait(&y);

    sem\_post(&x);

    printf("%d reader is inside\n",readercount);

    usleep(3);

    sem\_wait(&x);

    readercount--;

    if(readercount==0)

    {

        sem\_post(&y);

    }

    sem\_post(&x);

    printf("%d Reader is leaving\n",readercount+1);

    return NULL;

}

void \*writer(void\* param)

{

    printf("Writer is trying to enter\n");

    sem\_wait(&y);

    printf("Writer has entered\n");

    sem\_post(&y);

    printf("Writer is leaving\n");

    return NULL;

}

int main()

{

    int n2,i;

    printf("Enter the number of readers:");

    scanf("%d",&n2);

    printf("\n");

    int n1[n2];

    sem\_init(&x,0,1);

    sem\_init(&y,0,1);

    for(i=0;i<n2;i++)

    {

        pthread\_create(&writerthreads[i],NULL,reader,NULL);

        pthread\_create(&readerthreads[i],NULL,writer,NULL);

    }

    for(i=0;i<n2;i++)

    {

        pthread\_join(writerthreads[i],NULL);

        pthread\_join(readerthreads[i],NULL);

    }

}

## output:

user1@sumit-HP-Pro-3330-MT:~/MCA\_Shruti\_21/ASS5$ gcc ass5\_04.c

user1@sumit-HP-Pro-3330-MT:~/MCA\_Shruti\_21/ASS5$ ./a.out

Enter the number of readers:5

1 reader is inside

Writer is trying to enter

Writer is trying to enter

2 reader is inside

2 Reader is leaving

Writer is trying to enter

2 reader is inside

2 Reader is leaving

2 reader is inside

Writer is trying to enter

2 Reader is leaving

2 reader is inside

2 Reader is leaving

Writer is trying to enter

1 Reader is leaving

Writer has entered

Writer is leaving

Writer has entered

Writer is leaving

Writer has entered

Writer is leaving

Writer has entered

Writer is leaving

Writer has entered

Writer is leaving