**OS lab statements**

**1)Write a shell program to print given number in reverse order**

echo -n "Enter a number:"

read n

num=0

while [ $n -gt 0 ]

do

num=$(expr $num \\* 10)

k=$(expr $n % 10)

num=$(expr $num + $k)

n=$(expr $n / 10)

done

echo "The reversed number is " $num

**Write a shell program to perform arithmetic operations using case**

read -p "Enter a string: " str

length=${#str}

i=$((length-1))

while [ $i -ge 0 ]

do

revstr=$revstr${str:$i:1}

i=$((i-1))

done

echo "Reverse of $str is $revstr"

**Write a  shell script to check file type and permissions of a given input by user**

echo -n "Enter file name : "

read file

#Checking the filetypes

if [ -e $file ]

then

if [ -d $file ]

then

echo "The file is a directory."

else

ch=`ls -l $file | cut -c 1`

echo $ch

if [ $ch == '-' ]

then

echo "The file is a text file."

elif [ $ch == 'b' ]

then

echo "The file is a block file."

elif [ $ch == 'c' ]

then

echo "The file is a character file."

elif [ $ch == 'l' ]

then

echo "The file is a link."

fi

fi

fi

#Cheking for WRITE permission

[ -w $file ] && W="Write = yes" || W="Write = No"

#Checking for EXECUTE permission

[ -x $file ] && X="Execute = yes" || X="Execute = No"

#Checking for READ permission

[ -r $file ] && R="Read = yes" || R="Read = No"

echo "$file permissions"

echo "$W"

echo "$R"

echo "$X"

**Write a shell script to Find factorial of a given number with and without recursion**

#!/usr/bin/bash

# Recursive factorial function

factorial()

{

product=$1

# Defining a function to calculate factorial using recursion

if((product <= 2)); then

echo $product

else

f=$((product -1))

# Recursive call

f=$(factorial $f)

f=$((f\*product))

echo $f

#echo "The factorial of the" $num "is" $f

fi

}

# main program

# reading the input from user

echo "Enter the number:"

read num

# defining a special case for 0! = 1

if((num == 0)); then

echo 1

else

#calling the function

factorial $num

fi

**4)Write a program demonstrating use of different system calls.**

1. **process related system all:fork,wait**

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/wait.h>

int main()

{

pid\_t p1, p2;

p1=fork();

if (p1==0){

printf("PID of 1st child P1 is: %d\n",getpid());

printf("PID of type PARENT of P1 is %d\n",getppid());

}

else{

wait(NULL);

p2=fork();

if(p2==0){

printf("PID of 2nd child P2 is: %d\n",getpid());

printf("PID of type PARENT of P2 is %d\n",getppid());

}

else{

wait(NULL);

printf("PID of the PARENT process is %d\n",getpid());

}

}

}

1. **file related:open ,read,write,close**

//open.c

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<fcntl.h>

int main()

{

int fd1,fd2,n;

char buff[25];

fd1=open("test.txt",O\_RDONLY);

fd2=open("test2.txt",O\_WRONLY|O\_APPEND);

//printf("The file descriptor of the file is: \n"%d,fd);

n=read(fd1,buff,15);

write(fd2,buff,n);

//int close(int fd);

}

**5) Implement multithreading for Matrix Operations using Pthreads.**

**6)Implementation of Classical problems using Threads and Mutex.**

Reader-Writer Problem

#include<stdio.h>

#include<pthread.h>

#include<unistd.h>

#include<stdlib.h>

pthread\_mutex\_t wr,mutex;

int a = 10,readcount=0;

void \* reader(void \*arg){

long int num;//when we enter into thread routine we first convert void \* argument to integer argument

//why long int?-void \* = 8 bytes and int = 4 bytes

num=(long int) arg;

pthread\_mutex\_lock(&mutex);

readcount++;

pthread\_mutex\_unlock(&mutex);

if(readcount==1){

pthread\_mutex\_lock(&wr);

}

printf("\nReader %ld is in critical section",num);

printf("\nReader %ld is reading data %d",num,a);

//sleep(1);

pthread\_mutex\_lock(&mutex);

readcount--;

pthread\_mutex\_unlock(&mutex);

if(readcount==0){

pthread\_mutex\_unlock(&wr);

}

printf("\nReader %ld left the critical section",num);

}

void \* writer(void \*arg){

long int num;

num=(long int)arg;

//lock wr variable to enter critical section

pthread\_mutex\_lock(&wr);

printf("\nWriter %ld is in critical section",num);

printf("\n Writer %ld have written data as %d:",num,++a);

//sleep(1);

//writer releases a lock on wr

pthread\_mutex\_unlock(&wr);

printf("\nWriter %ld left the critical section",num);

}

int main()

{

pthread\_t r[10],w[10]; //array of variable reader and writer

long int i,j;

int no\_of\_reader,no\_of\_writer; //index variables required for joining threads

//initialize mutex variables

pthread\_mutex\_init(&wr,NULL);

pthread\_mutex\_init(&mutex,NULL);

//get number of reader and writer

printf("Enter number of readers:");

scanf("%d",&no\_of\_reader);

printf("Enter number of writers:");

scanf("%d",&no\_of\_writer);

//create reader and writer threads of given number

for (i=0;i<no\_of\_reader;i++){

pthread\_create(&r[i],NULL,reader,(void \* )i);

}

for (j=0;j<no\_of\_writer;j++){

pthread\_create(&w[j],NULL,writer,(void \* )j);

}

//Join the threads

for (i=0;i<no\_of\_reader;i++){

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

}

for (j=0;j<no\_of\_writer;j++){

pthread\_join(w[j],NULL);

}

return 0;

}

**7)Implementation of Classical problems using Threads and Semaphore**

#include<stdio.h>

#include<pthread.h>

#include<unistd.h>

#include<stdlib.h>

#include<semaphore.h>

sem\_t empty,full,mutex;

int buffer[5];

int count=0;

void \* producer(void \*arg){

long int num=(long int)arg;

//Producer is trying to produce the data

sem\_wait(&empty);

//Producer is allowed to produce data

//Producer is waiting for his turn

sem\_wait(&mutex);

//Producer is producing the data;

buffer[count] = rand()%10;

printf("\nProducer: %ld produced %d",num+1,buffer[count]);

count++;

sleep(1);

sem\_post(&mutex);//Producer has released lock on critical section

sem\_post(&full);//Producer is incrementing full value

}

void \* consumer(void \*arg){

long int num=(long int)arg;

//Consumer is trying to consume the data

sem\_wait(&full);

//Consumer is allowed to consume data

//Consumer is waiting for his turn

sem\_wait(&mutex);

//Consumer is consuming the data;

buffer[count] = rand()%10;

printf("\nConsumer: %ld consumed %d",num+1,buffer[count]);

count--;

sleep(1);

sem\_post(&mutex);//Consumer has released lock on critical section

sem\_post(&empty);//Consumer is incrementing empty value

}

int main(){

int no\_of\_prod,no\_of\_con;

pthread\_t p[10],c[10];

unsigned long int i,j,k,l;

//Number of producers and consumers

printf("Enter no.of producers:");

scanf("%d",&no\_of\_prod);

printf("Enter no.of consumers:");

scanf("%d",&no\_of\_con);

//initialize semaphore variables

sem\_init(&empty,0,5); //1 var=name of variable, 2 var= 0 means not shared, 3 var= initial value

sem\_init(&full,0,0);

sem\_init(&mutex,0,1);

//create threads of producer and consumer

for(i=0;i<no\_of\_prod;i++){

pthread\_create(&p[i],NULL, producer, (void \*)i);

}

for(j=0;j<no\_of\_con;j++){

pthread\_create(&c[j],NULL, consumer, (void \*)j);

}

//join threads of producer and consumer

for(k=0;k<no\_of\_prod;k++){

pthread\_join(p[k],NULL);

}

for(l=0;l<no\_of\_con;l++){

pthread\_join(c[l],NULL);

}}

**8)Write a program to check whether a given system is in safe state or not using Banker’s  Deadlock Avoidance algorithm.**

// C Program to Implement Safety Algorithm- (Banker's Algorithm- Deadlock Avoidance Algorithm)

//This algo Prints whether the given system state is in SAFE state or UNSAFE state. If safe, then prints the SAFE SEQUENCE

#include <stdio.h>

#include <stdbool.h>

struct process\_info

{

int max[10];

int allocated[10];

int need[10];

};

int no\_of\_process,no\_of\_resources;

//Take the input

void input(struct process\_info process[no\_of\_process],int available[no\_of\_resources])

{

//Fill array of Structure

for(int i=0;i<no\_of\_process;i++)

{

printf("Enter process[%d] info\n",i);

printf("Enter Maximum Need: ");

for(int j=0;j<no\_of\_resources;j++)

scanf("%d",&process[i].max[j]);

printf("Enter No. of Allocated Resources for this process: ");

for(int j=0;j<no\_of\_resources;j++)

{

scanf("%d",&process[i].allocated[j]);

//calculate need/future need

process[i].need[j]= process[i].max[j] - process[i].allocated[j];

}

}

printf("Enter Available Resources: ");

for(int i=0;i<no\_of\_resources;i++)

{

scanf("%d",&available[i]);

}

}

//Print the Info in Tabular Form

void showTheInfo(struct process\_info process[no\_of\_process])

{

printf("\nPID\tMaximum\t\tAllocated\tNeed\n");

for(int i=0;i<no\_of\_process;i++)

{

printf("P[%d]\t",i);

for(int j=0;j<no\_of\_resources;j++)

printf("%d ",process[i].max[j]);

printf("\t\t");

for(int j=0;j<no\_of\_resources;j++)

printf("%d ",process[i].allocated[j]);

printf("\t\t");

for(int j=0;j<no\_of\_resources;j++)

printf("%d ",process[i].need[j]);

printf("\n");

}

}

//Apply safety algo

bool applySafetyAlgo(struct process\_info process[no\_of\_process],int available[no\_of\_resources],int safeSequence[no\_of\_process])

{

bool finish[no\_of\_process];

int work[no\_of\_resources];

for(int i=0;i<no\_of\_resources;i++)

{

work[i]=available[i];

}

for(int i=0;i<no\_of\_process;i++)

finish[i]=false;

bool proceed=true;

int k=0;

while(proceed)

{

proceed=false;

for(int i=0;i<no\_of\_process;i++)

{

bool flag=true;

//Find Index i

if(finish[i]==false)

{

for(int j=0;j<no\_of\_resources;j++)

{

//if Need <= Work

if(process[i].need[j] <= work[j])

{

continue;

}

else

{

flag=false; // implies that the current process need > work

break;

}

}

if(flag==false)

continue; //check for next process

//If we get Index i(or process i), update work

for(int j=0;j<no\_of\_resources;j++)

work[j]=work[j]+ process[i].allocated[j];

finish[i]=true;

safeSequence[k++]=i;

proceed=true; // tells that we got atleast one process in safe state, we can proceed

}

}//end of outer for loop

} // end of while

//check finish array

int i;

for( i=0;i<no\_of\_process&&finish[i]==true;i++)

{

continue;

}

//If all processes are completed, then return true

if(i==no\_of\_process)

return true;

else

return false;

}

//Checks if we State is safe or not

bool isSafeState(struct process\_info process[no\_of\_process],int available[no\_of\_resources],int safeSequence[no\_of\_process])

{

if(applySafetyAlgo(process,available,safeSequence)==true)

return true;

return false;

}

int main()

{

printf("Enter No of Process\n");

scanf("%d",&no\_of\_process);

printf("Enter No of Resource Instances in system\n");

scanf("%d",&no\_of\_resources);

int available[no\_of\_resources];

int safeSequence[no\_of\_process];

//Create Array of Structure to store Processes's Informations

struct process\_info process[no\_of\_process];

printf("\*\*\*\*\*\*\*\*\*\*\*Enter details of processes\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

//Take the Input

input(process,available);

//Print the Info in Tabular Form

showTheInfo(process);

if(isSafeState(process,available,safeSequence))

{

printf("\nSystem is in SAFE State\n");

printf("Safe Sequence is: ");

for(int i=0;i<no\_of\_process;i++)

printf("P[%d] ",safeSequence[i]);

printf("1");

}

else

printf("0");

return 0;}

**OS tutorial statement**

1. **Shell program to check entered string is  palindrome or not**

echo "Enter a number:"

read num

# Storing the remainder

s=0

# Store number in reverse

# order

rev=""

# Store original number

# in another variable

temp=$num

while [ $num -gt 0 ]

do

# Get Remainder

s=$(( $num % 10 ))

# Get next digit

num=$(( $num / 10 ))

# Store previous number and

# current digit in reverse

rev=$( echo ${rev}${s} )

done

if [ $temp -eq $rev ];

then

echo "Number is palindrome"

else

echo "Number is NOT palindrome"

fi

1. **Shell program to  find sum of digits of a given number**

echo "Enter a number: "

read num

sum=0

while [ $num -gt 0 ]

do

mod=$((num % 10)) #It will split each digits

sum=$((sum + mod)) #Add each digit to sum

num=$((num / 10)) #divide num by 10.

done

echo "The sum of the numbers in the given number is: "$sum

1. **Shell program to check whether given string is present in another string or not**

#!/bin/bash

echo "Enter 1st string: "

read str

echo "Enter 2nd string: "

read sub

if [[ "$str" == \*"$sub"\* ]]; then

echo "Given string is present in another string"

else

echo "Given string is not present in another string"

fi

1. **C program  to demonstrate the use of communication related system calls**

**Pipe()**

#include<unistd.h>

#include<stdio.h>

#include<sys/types.h>

#include<sys/wait.h>

int main()

{

int fd[2], n;

char buffer[100];

pid\_t p;

pipe(fd);

p=fork();

if(p>0){

printf("Passing value to child\n");

write(fd[1], "hello\n", 6);

}

else{

printf("Child received data\n"); n-read(fd[0], buffer, 100);

write(1,buffer,n);

}

}

**Shmget()\_sender**

#include<stdlib.h>

#include <unistd.h>

#include<sys/shm.h>

#include<string.h>

#include<stdio.h>

int main()

{

void \*shared\_memory;

char buff[100];

int shmid;

shmid=shmget((key\_t)1122, 1024, 0666|IPC\_CREAT); //creates shared memory segment with key 2345, having

printf("Key of shared memory is %d\n", shmid);

shared\_memory=shmat(shmid, NULL, 0); //process attached to shared memory segment

printf("Process attached at %p\n", shared\_memory); //this prints the address where the segment is attack

printf("Enter some data to write to shared memory\n");

read(0,buff,100); //get some input from user

strcpy(shared\_memory, buff); //data written to shared memory

printf("You wrote: %s\n", (char \*)shared\_memory);

}

**Shmget()\_receiver**

#include<stdio.h>

#include<stdlib.h>

#include <unistd.h>

#include<sys/shm.h>

#include<string.h>

int main()

{

void \*shared\_memory;

char buff[100];

int shmid;

shmid=shmget((key\_t)1122, 1024, 0666);

printf("Key of shared memory is %d\n", shmid);

shared\_memory=shmat(shmid, NULL, 0); //process attached to shared memory segment

printf("Process attached at %p\n", shared\_memory);

printf("Data read from shared memory is: %s\n", (char \*)shared\_memory);

}

**Mmap()**

#include <stdio.h>

#include <unistd.h>

#include <sys/wait.h>

#include <sys/mman.h>

int main()

{

int N=5; // Number of elements for the array

int \*ptr = mmap(NULL,N\*sizeof(int),

PROT\_READ | PROT\_WRITE,

MAP\_SHARED | MAP\_ANONYMOUS,

0,0);

if(ptr == MAP\_FAILED){

printf("Mapping Failed\n");

return 1;

}

for(int i=0; i < N; i++){

ptr[i] = i + 1;

}

printf("Initial values of the array elements :\n");

for (int i = 0; i < N; i++ ){

printf(" %d", ptr[i] );

}

printf("\n");

pid\_t child\_pid = fork();

if ( child\_pid == 0 ){

//child

for (int i = 0; i < N; i++){

ptr[i] = ptr[i] \* 10;

}

}

else{

//parent

waitpid ( child\_pid, NULL, 0);

printf("\nParent:\n");

printf("Updated values of the array elements :\n");

for (int i = 0; i < N; i++ ){

printf(" %d", ptr[i] );

}

printf("\n");

}

int err = munmap(ptr, N\*sizeof(int));

if(err != 0){

printf("UnMapping Failed\n");

return 1;

}

return 0;

}

1. **C program to perform file related system call operations**

//open.c

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<fcntl.h>

int main()

{

int fd1,fd2,n;

char buff[25];

fd1=open("test.txt",O\_RDONLY);

fd2=open("test2.txt",O\_WRONLY|O\_APPEND);

//printf("The file descriptor of the file is: \n"%d,fd);

n=read(fd1,buff,15);

write(fd2,buff,n);

//int close(int fd);

}

1. **C program to demonstrate the arithmetic operation on any two numbers using multithreading**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h> //Header file for sleep(). man 3 sleep for details.

#include <pthread.h>

// A normal C function that is executed as a thread

// when its name is specified in pthread\_create()

void \*mythread(void \*vargp)

{

int a=10,b=20,c;

c= a+b;

printf("Addition is: %d\n",c);

return NULL;

}

int main()

{

pthread\_t thread\_id;

printf("Before Thread\n");

pthread\_create(&thread\_id, NULL, mythread, NULL);

pthread\_join(thread\_id, NULL);

printf("After Thread\n");

exit(0);

}

1. **Implementation of Reader writer using Threads and Semaphore.**

#include<stdio.h>

#include<pthread.h>

#include<semaphore.h>

#include<unistd.h>

sem\_t r,w;

int h=23,m=59,s=55;

void \*reader(),\*writer();

int main()

{

pthread\_t rth,wth;

void \*status;

sem\_init(&r,0,0);

sem\_init(&w,0,1);

pthread\_create(&rth,NULL,(void \*)&reader,NULL);

pthread\_create(&wth,NULL,(void \*)&writer,NULL);

pthread\_join(rth,status);

pthread\_join(wth,status);

sem\_destroy(&w);

sem\_destroy(&r);

}

void \*writer()

{

while(1)

{

sem\_wait(&w);

s=s+1;

if(s==60)

{

m++; s=0;

}

if(m==60)

{

h++; m=0;

}

if(h==24)

{

h=1;

}

//sleep(1);

sem\_post(&r);

}

}

void \*reader()

{

while(1)

{

sem\_wait(&r);

printf("\n Display:\t");

printf("%d:%d:%d",h,m,s);

sem\_post(&w);

}

}

1. **Implementation of Classical problems producer-consumer using Threads and Mutex.**

#include<stdio.h>

#include<pthread.h>

#include<string.h>

#include<semaphore.h>

char buffer[20];

void \*produce();

void \*consume();

pthread\_mutex\_t mut;

int main()

{

void \*status;

pthread\_t p\_thr,c\_thr;

pthread\_mutex\_init(&mut,0);

pthread\_create(&p\_thr,NULL,(void\*)&produce,NULL);

pthread\_create(&c\_thr,NULL,(void\*)&consume,NULL);

pthread\_join(p\_thr,&status);

pthread\_join(c\_thr,&status);

return 0;

}

void \*produce()

{

char str[20];

while(1)

{

pthread\_mutex\_lock(&mut);

printf("\nENTER A STRING:");

scanf("%s",str);

strcpy(buffer,str);

pthread\_mutex\_unlock(&mut);

sleep(1);

}

}

void \*consume()

{

char str1[20];

while(1)

{

pthread\_mutex\_lock(&mut);

strcpy(str1,buffer);

printf("\nTHE CONSUMED STRING IS :%s",str1);

pthread\_mutex\_unlock(&mut);

sleep(1);

}

}