ACEDMIC TASK-3

OPERATING SYSTEM (CSE-316)

STUDENT NAME: Smithin reddy

STUDENT ID: 11807366

EMAIL ADDRESSES: [smithinreddy4704@gmail.com](mailto:smithinreddy4704@gmail.com)

Q1.EXPLANATION

This question is about the fibonacci sequence of arrangement of sequence numbers here fibonacci series means a series of numbers in which each number is the sum of the two preceding numbers.the use of fibonacci series are computer algorithms such as fibonacci heap data structure and graphs called fibonacci cubes in this question the series numbers are 0,1,1,2,3,4,7,9….formed it expresses as f ib0=0 f ib1=1 f ibn=f ibn-1+f ibn-2 the following program works as the multithreaded program and it works as on the command line if the user will enter the number of fibonacci numbers the program is generated the program will create a thread that generates the fibonacci numbers placed in the sequence data like array when it finishes the thread the parent thread will execute the output which child thread is generated we need to prevent the parent process from completing before the child process because child process will become orphan.there fore to prevent this we need to use the sys6call wait()the child process complete before the parent process

CODE:-

include<stdio.h>

#include<iostream>

using namepsace std;

int main(){

#include<iostream>

#include<pthread.h>

#include <unistd.h>

#include<stdlib.h>

#include<sys/wait.h>

using namespace std;

int size,first=0,second=1,third,n;

int a[1000];

void \* cal(void \* arg)

{

for(int i=0;i<n;i++)

{

first=a[i];

third=first+second;

first=second;

second=third;

}

}

int main(){ cout<<"enter number of elements"<<endl;

cin>>n;

pthread\_t fd;

{

pthread\_create(&fd,NULL,cal,NULL);

{

pthread\_join(fd,NULL);

wait(1);

for(int i=0;i<n;i++)

{

Printf(“parent is wating on chid process…\n");

Wait();

Printf(“parent process is done.\n");

}

return 0;

}

ALGORITHM:

1.the program takes an input from the user .

2.in the next step programm creates a child process.And the value returned from. The fork is compared with 0 to check if child process is created or not

3.if the child process is created then the value is used inside a whole loop and then the sequence is printed.

4.while printing the sequence the parent process waits until child process completes.

Q2.EXPLANATION:-

We know how to find an effective access time (EAT) for a given page-fault rate (p).Here, we have to find ‘p' for given ‘EAT' so we set up the following equation and solve for ‘p':

(Note: 1 millisecond = 1,000,000 nanoseconds = 1e6 nanoseconds)

Time taken to service page Fault for empty pages or unmodified page = 8ms.

Time taken to service page Fault for modified pages = 20ms

Memory access time = 100ns

Effective Access time = 200ns

EAT = (1-p)\*(100) + (p)\*(100 + (1-.7)\*(8msec) + (.7)\*(20msec))

= 100 - 100p + 100p +(2.4e6)\*p + (14e6)\*p

=100 + (16.4e6)\*p

200 = 100 + (16.4e6)\*p

P = 100/16.4e6 = 6.0975609756097560975609756097561e-6~6.01e-6

P-- > page Fault Rate

CODE :

#include <stdio.h>

#include <stdlib.h>

double page\_fault\_rate();

void userInput(void);

double service\_page\_fault\_empty;

double service\_page\_fault\_modified;

double mem\_access\_time;

double times\_page\_modified;

double effective\_access\_time;

double pageFaultRate;

double service\_page\_fault\_empty\_ns;

double service\_page\_fault\_modified\_ns;

double times\_page\_modified\_per;

void main(){

int swtch;

do{

printf("Select the required option \n");

printf("1.Find the PageFault Rate\n");

printf("2.Exit");

scanf("%d",&swtch);

switch(swtch){

case 1:userInput();break;

case 2:exit(0);

}

printf("\n\n");

}while(swtch<3);

}

void userInput(){

printf("\nEnter service Page Fault [Empty|Page is not Modified][in milliseconds]");

scanf("%lf",&service\_page\_fault\_empty);

printf("Enter Service Page Fault [Modified Page][in milliseconds]");

scanf("%lf",&service\_page\_fault\_modified);

printf("Enter Memory Access Time[in nanoseconds]");

scanf("%lf",&mem\_access\_time);

printf("Enter Percentage of time the page to be replaced is modified[0-100]");

scanf("%lf",&times\_page\_modified);

printf("Enter Effective Access time[in nanoseconds]");

scanf("%lf",&effective\_access\_time);

service\_page\_fault\_empty\_ns = (service\_page\_fault\_empty\*1000000);

service\_page\_fault\_modified\_ns = (service\_page\_fault\_modified\*1000000);

times\_page\_modified\_per = (times\_page\_modified/100);

printf("\nPage Fault rate calculated For:\n");

printf("Service Page Fault[Empty|Page Not Modified]=%lf \n",service\_page\_fault\_empty\_ns);

printf("Service Page Fault [Modified Page][in nanoseconds] %lf \n",service\_page\_fault\_modified\_ns);

printf("Memory Access Time[in nanoseconds]%lf\n",mem\_access\_time);

printf("Effective Access Time %lf\n",effective\_access\_time);

pageFaultRate = page\_fault\_rate(service\_page\_fault\_empty\_ns,service\_page\_fault\_modified\_ns,mem\_access\_time,times\_page\_modified\_per,effective\_access\_time);

printf("\nMaximum Acceptable Page Fault rate = %.2e[exponential notation]",pageFaultRate);

}

double page\_fault\_rate(double servicePageFaultEmpty,double servicePageFaultMod,double memAccess,double timesPages,double effAccess){

double assume,serve;

double numErator,denOminator;

double pageFault;

assume = (1- timesPages)\*servicePageFaultEmpty;

serve = timesPages\*servicePageFaultMod;

numErator = effAccess - memAccess;

denOminator = (assume+serve);

pageFault = numErator/denOminator;

return pageFault;

}

ALGORITHM

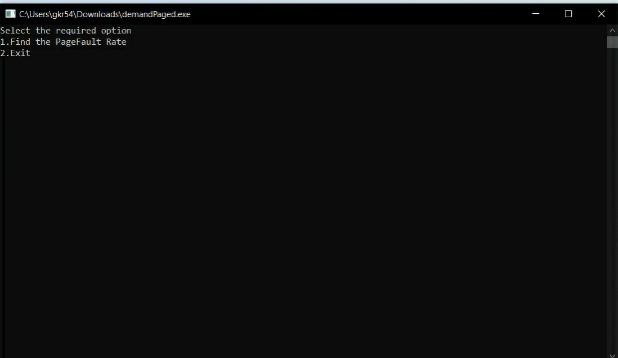
1.the page generate 70 percentage of the time while compared to the previous memory

2.here the programm declares an array of file

3.then the programm creates Fault process and check for the return value. If the process is created then the content of the source file is copied

4.then the value between millisecond and percentage are been done.

OUTPUT:



4