**OPERATING SYSTEMS – CSE316**

**Assignment based on Multithreading and Page Fault**

**Student Name:** Rohith Thokala

**Student Registration ID –** 11813646

**Email Id –**  rohithkumar.1601@gmail.com

**Github –** <https://github.com/Ricky-1999/rickassignment>

**Section –** K18GT

**Roll No – 56**

Code for Program Fibo Series

#include <stdio.h>

#include<unistd.h>

#include <pthread.h>

#include <stdlib.h>

pthread\_t tid;

pthread\_t tid2;

int fibo[1000];

int top=-1;

void \*printFibo(void \*vargp){

pthread\_join(tid2,NULL);

sleep(1);

for(int i=0;i<=\*(int\*)vargp;i++)

{

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

printf(" ");

}

}

void \*calcFibo(void \*vargp){

int t1 = 0, t2 = 1, nextTerm = 0;

fibo[0]=t1;

fibo[1]=t2;

top=1;

int i=0;

while(i <= \*(int \*)vargp)

{

nextTerm = t1 + t2;

top++;

fibo[top]=nextTerm;

t1 = t2;

t2 = nextTerm;

i++;

}

}

int main()

{

int n;

printf("Chose a number to get Fibo series");

scanf("%d",&n);

pthread\_create(&tid,NULL,printFibo,(void\*)&n);

pthread\_create(&tid2, NULL, calcFibo, (void\*)&n);

pthread\_join(tid, NULL);

exit(0);

}

Algorithm:

1.The algo collects the value from the user when entered .

2.In the next step the program creates a separate thread which generates Fibo series.

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

4- When the thread finishes execution,the parent thread will output the sequence series generated by the child thread.

Algo:

START

Procedure Fibonacci(n)

declare f0, f1, fib, loop

set f0 to 0

set f1 to 1

display f0, f1

for loop ← 1 to n

fib ← f0 + f1

f0 ← f1

f1 ← fib

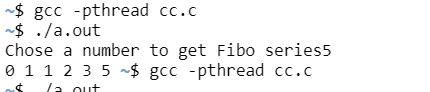
display fib

end for

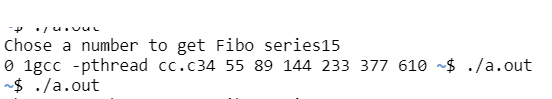
END

Test Cases:

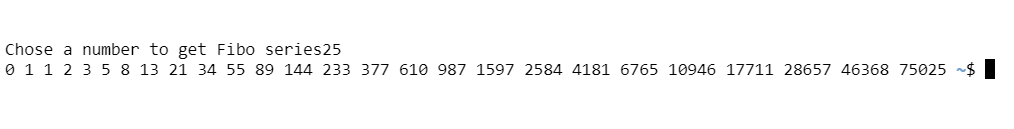
Test Case 1 :



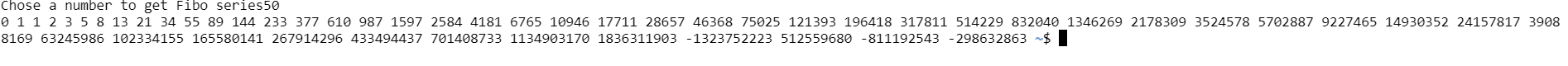
Test Case 2:



Test Case 3 :



Test Case 4 :



Tested with 4 different values.

1. Consider a scenario of demand paged memory. Page table is held in registers. It takes 8 milliseconds to service a page fault if an empty page is available or the replaced page is not modified and 20 milliseconds if the replaced page is modified. Memory access time is 100 nanoseconds. Assume that the page to be replaced is modified 70 percent of the time. Generate a solution to find maximum acceptable page-fault rate for access time that is not more than 200 nanoseconds.

Solution:

We have to generate Max acceptable Page-Fault for Access Time and it should not be more than 200 nano seconds

Time taken of empty Page of page fault is given = 8ms(8 milli seconds)

And the modified time taken is given as = 20ms( 20 milli seconds)

And We have a memory access time is 100 nanoseconds (20ns)

And Max Access Time should not go beyond 200ns , i.e., 200 nano seconds

E.A.T = 200ns

P= Page Fault (First we have to find page fault) with respect to E.A.T

Here is a point that 1 ms(milli second) = 1 million nanoseconds(1,00,000 nano seconds)

E.A.T = 200 nano seconds

(1-p)\*100 + p \*(100 + (1-70%) \* 8ms + 0.70 \* 20ms) = 200 nano seconds

(1-p)\*100 + p \*(100 + (1-0.70) \* 8ms +0.70 \* 20ms) = 200 nano seconds

100 -100p + 100p +2.4ms \* p + 1.4ms =200 nano seconds

100 +2.4ms \*p +1.4ms = 200 nano seconds

100 + (16.4 ns)\*p =200 ns

P = 100/16.4ns

P = 6.016 nanoseconds

The Page Fault Rate is received as 6.016 nano seconds

.Code for Page Fault and Effiecient Access Time:

#include <stdio.h>

#include <stdlib.h>

double page\_fault\_rate();

void userInput(void);

double sp\_fault\_empty;

double sp\_fault\_modified;

double mat;

double tp\_modified;

double eat;

double pageFaultRate;

double sp\_fault\_empty\_ns;

double sp\_fault\_modified\_ns;

double tp\_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 ms]");

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

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

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

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

scanf("%lf",&mat);

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

scanf("%lf",&tp\_modified);

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

scanf("%lf",&eat);

sp\_fault\_empty\_ns = (sp\_fault\_empty\*1000000);

sp\_fault\_modified\_ns = (sp\_fault\_modified\*1000000);

tp\_modified\_per = (tp\_modified/100);

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

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

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

printf("Memory Access Time[in ns]%lf\n",mat);

printf("Effective Access Time %lf\n",eat);

pageFaultRate = page\_fault\_rate(sp\_fault\_empty\_ns,sp\_fault\_modified\_ns,mat,tp\_modified\_per,eat);

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;

}

Working :

1.It takes the prior information from the user

2.It asks the user to enter the values in milliseconds and nano seconds

3.The output was try to calculate p value( Page Fault Rate ) in converted dorm to nanoseconds using EAT at the other end.

4.Then the runtime commutes to exit.

Output with Runtime Line:

