Operating System Assignments

Student Name:- Atul Shukla

Reg No:- 11804538

Section:- K18SP

Roll No:- 44

Email Address:- at887439@gmail.com

Github Link:- https://github.com/at887439/Os-assignment

Problem:-

Question No 21

Explain the problem in terms of operating system concept?

The process of loading the page into memory on demand is known as demand paging. If CPU try to refer a page that is currently not available in the main memory, it generates an interrupt indicating memory access fault. The OS puts the interrupted process in a blocking state.In this we have to find Effective Access Time (EAT) for a given page-fault rate(p).

Time taken to service page fault for empty page or unmodified page= 8ms

Time taken to service page fault for modified page= 20ms

Memory access time= 100ms

Effective Access time= 200ms

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 =

6.09756097561e-6 ~ 6.01e-6

Algorithm:-

Step 1) The execution begins with process P1, which has burst time 4. Here, every process executes for 2 seconds. P2 and P3 are still in the waiting queue.

Step 2) At time =2, P1 is added to the end of the Queue and P2 starts executing.

Step 3) At time=4 , P2 is preempted and add at the end of the queue. P3 starts executing.

Step 4) At time=6 , P3 is preempted and add at the end of the queue. P1 starts executing.

Step 5) At time=8 , P1 has a burst time of 4. It has completed execution. P2 starts execution

Step 6) P2 has a burst time of 3. It has already executed for 2 interval. At time=9, P2 completes execution. Then, P3 starts execution till it completes.

Functions:-

```
1. Start traversing the pages
   i)  If set holds less pages than capacity.
      a) Insert page into the set one by one until the size of set reaches
         capacity or all page requests are processed.
      b) Simultaneously maintain the pages in the queue to perform FIFO
      c) Increment page fault
   ii) Else

           If current page is present in set, do nothing.

           Else

   a)  Remove the first page from the queue as it was the first to be entered
       in the memory
   b)  Replace the first page in the queue with current page in the string
   c)  Store current page in the queue
   d)  Increment page faults
2. Return page faults
```

Coding :-

```
#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_
a ccess_time,times_page_modified_per,effective_access_time);     printf("\nMaximum
Acceptable Page Fault rate = %.2e[exponential notation]",pageFaultRate);
```

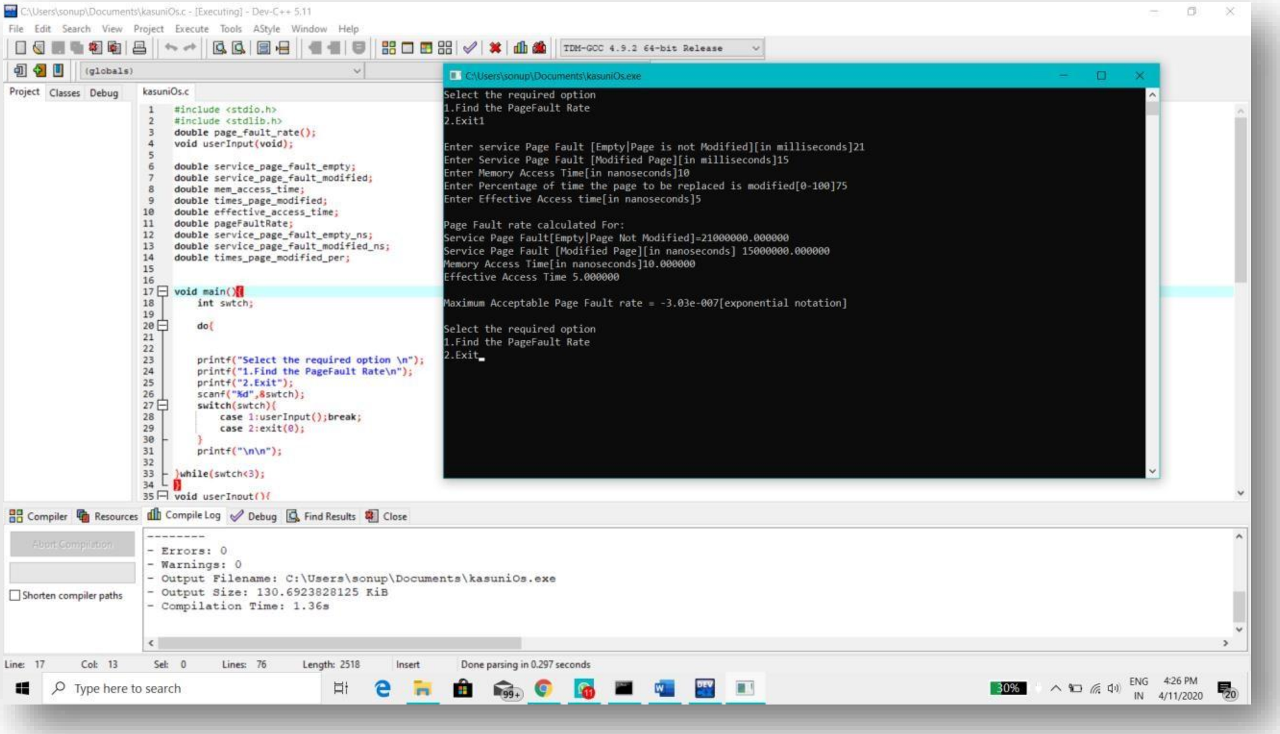
```
}

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;


}
```

Output:-



C:\Users\sonup\Documents\kasuniOs.c - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
kasuniOs.c
C:\Users\sonup\Documents\kasuniOs.exe
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Line: 17 Col: 13 Sel: 0 Lines: 76 Length: 2518 Insert Done parsing in 0.297 seconds
Type here to search
30%
ENG IN 4:26 PM 4/11/2020

```c
#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(){
```

```
Select the required option
1.Find the PageFault Rate
2.Exit1

Enter service Page Fault [Empty|Page is not Modified][in milliseconds]21
Enter Service Page Fault [Modified Page][in milliseconds]15
Enter Memory Access Time[in nanoseconds]10
Enter Percentage of time the page to be replaced is modified[0-100]75
Enter Effective Access time[in nanoseconds]5

Page Fault rate calculated For:
Service Page Fault[Empty|Page Not Modified]=21000000.000000
Service Page Fault [Modified Page][in nanoseconds] 15000000.000000
Memory Access Time[in nanoseconds]10.000000
Effective Access Time 5.000000

Maximum Acceptable Page Fault rate = -3.03e-007[exponential notation]

Select the required option
1.Find the PageFault Rate
2.Exit_
```

```
---------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\sonup\Documents\kasuniOs.exe
- Output Size: 130.6923828125 KiB
- Compilation Time: 1.36s
```

References:-

www.javatpoint.com

www.tutorialspoint.com

www.geeksforgeeks.com