School of computer science and engineering

Nameofthefaculty:Dr.Baljit Singh Saini

Programiningassignment

CourseTitle:OperatingSystem

Coursecode:CSE316

NAME: K . Sai Vivek Reddy

REGISTRATIONNUMBER:11805765

SECTION:K18VQ

CA_3

MAXMARKS:30

Q5)Sudesh Sharmaisa linuxexpert who wants to have an online system where he can handle student quaries.Since there can be multiple request at anytime he wishes to dedicate a fixed amout of time to every requests othat everyone gets a fairs hare of time.He will loginto system from 10 am to 12pm only.He wants to have a separate request squeues for students and faculty.Implement a strategy for the same.The summary at the end of the session should include the totaltime he spent on handling queries and average querytime.

Code:

```
#include<stdio.h>
#include<string.h>

struct process_Struct {
   char p_n[20];
   int a_t, b_t, c_t, r;
}temp_Struct;



void faculty_Queue(int n_o_p) {

   int c, arrival_Time, burst_Time, q_t;
   struct process_Struct faculty_Process_info[n_o_p];

   for(c = 0; c < n_o_p; c++) {
      printf("Enter the details of Process[%d]", c+1);
      puts("");
      printf("Process Name : ");ss
```

```
    scanf("%s", faculty_Process_info[c].p_n);

    printf("Arrival Time : ");
    scanf("%d", &faculty_Process_info[c].a_t);

    printf("Burst Time : ");
    scanf("%d", &faculty_Process_info[c].b_t);
    puts("");
}
printf("Now, enter the quantum time for FACULTY q : ");
scanf("%d", &q_t);



for(c = 0; c < n_o_p; c++) {
    for(int x = c +1; x < c; x++){
        if(faculty_Process_info[c].a_t > faculty_Process_info[x].a_t) {
            temp_Struct = faculty_Process_info[c];
            faculty_Process_info[c] = faculty_Process_info[x];
            faculty_Process_info[x] = temp_Struct;
        }
    }
}


for(c = 0; c < n_o_p; c++) {
    faculty_Process_info[c].r = faculty_Process_info[c].b_t;
    faculty_Process_info[c].c_t = 0;
}
```

```
int t_t, q, round_robin[20];
t_t = 0;
q = 0;
round_robin[q] = 0;


int flag, x, n, z, waiting_time = 0;
do {
   for(c = 0; c < n_o_p; c++){
      if(t_t >= faculty_Process_info[c].a_t){
         z = 0;
         for(x = 0; x <= q; x++) {
            if(round_robin[x] == c) {
               z++;
            }
         }
         if(z == 0) {
            q++;
            round_robin[q] == c;
         }
      }
   }

   if(q == 0) {
      n = 0;
   }
   if(faculty_Process_info[n].r == 0) {
      n++ ;
```

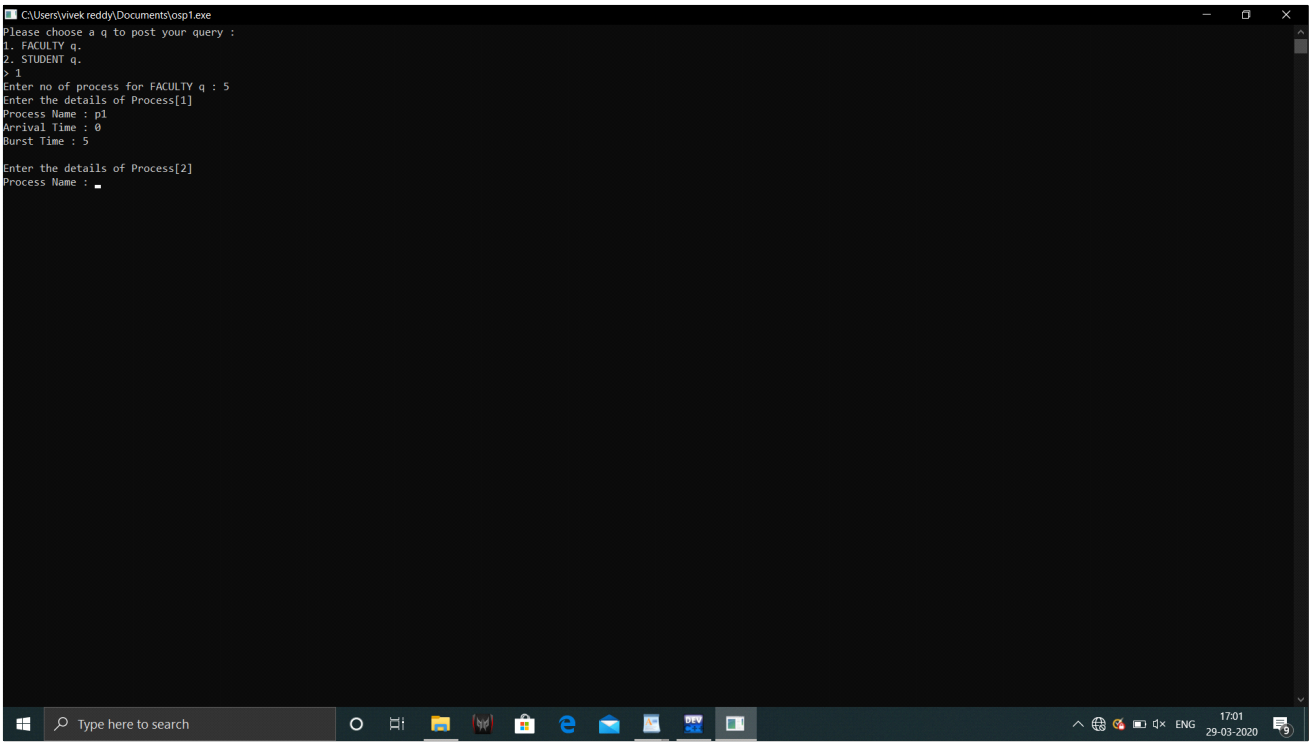
```
    }
    if(n > q) {
        n = (n - 1) % q;
    }
    if(n <= q) {
        if(faculty_Process_info[n].r > 0) {
            if(faculty_Process_info[n].r < q_t){
                t_t += faculty_Process_info[n].r;
                faculty_Process_info[n].r = 0;
            }else {
                t_t += q_t;
                faculty_Process_info[n].r -= q_t;
            }
            faculty_Process_info[n].c_t = t_t;
        }
        n++;
    }
    flag = 0;

    for(c = 0; c < n_o_p; c++) {
        if(faculty_Process_info[c].r > 0) {
            flag++;
        }
    }
}while(flag != 0);



puts("\n\t\t\t*****************************************");
puts("\t\t\t*****   ROUND ROBIN ALGORITHM OUTPUT   *****");
```

```
    puts("\t\t\t*****************************************\n");
    printf("\n|\tProcess Name\t  |\tArrival Time\t  |\tBurst Time\t |\tCompletion Time \t|\n");

    for(c = 0; c < n_o_p; c++){
        waiting_time = faculty_Process_info[c].c_t - faculty_Process_info[c].b_t - faculty_Process_info[c].a_t;

        printf("\n|\t %s\t   |\t %d\t   |\t %d\t   |\t %d\t   |\n", faculty_Process_info[c].p_n, faculty_Process_info[c].a_t, faculty_Process_info[c].b_t, faculty_Process_info[c].c_t);
    }

}

void student_Queue(int n_o_p) {

    int c, arrival_Time, burst_Time, q_t;
    struct process_Struct student_Process_info[n_o_p];

    for(c = 0; c < n_o_p; c++) {
        printf("Enter the details of Process[%d]", c+1);
        puts("");
        printf("Process Name : ");
        scanf("%s", student_Process_info[c].p_n);

        printf("Arrival Time : ");
        scanf("%d", &student_Process_info[c].a_t);

        printf("Burst Time : ");
```

```
        scanf("%d", &student_Process_info[c].b_t);
    }
    printf("Now, enter the quantum time for STUDENT q : ");
    scanf("%d", &q_t);



    for(c = 0; c < n_o_p; c++) {
        for(int x = c +1; x < c; x++){
            if(student_Process_info[c].a_t > student_Process_info[x].a_t) {
                temp_Struct = student_Process_info[c];
                student_Process_info[c] = student_Process_info[x];
                student_Process_info[x] = temp_Struct;
            }
        }
    }



    for(c = 0; c < n_o_p; c++) {
        student_Process_info[c].r = student_Process_info[c].b_t;
        student_Process_info[c].c_t = 0;
    }

    int t_t, q, round_robin[20];
    t_t = 0;
    q = 0;
    round_robin[q] = 0;
}
```

```
int main(int argc, char const *argv[]) {
    int select_q, n_o_p;

    puts("Please choose a q to post your query : ");
    puts("1. FACULTY q.");
    puts("2. STUDENT q.");
    printf("> ");
    scanf("%d", &select_q);

    switch(select_q) {
        case 1 :
            printf("Enter no of process for FACULTY q : ");
            scanf("%d", &n_o_p);

            faculty_Queue(n_o_p);

            break;

        case 2 :
            printf("Enter no of process for STUDENT q : ");
            scanf("%d", &n_o_p);

            student_Queue(n_o_p);

            break;

        default :
            printf("Please selet the correct option by running the program again.");
```

```
    }


    return 0;
}
```



report:

The given problem is based upon solving queries of persons of different classes i.e. Faculty and Students. Thus, these queries can be compared to different processes in terms of operating system where each process has its demands and needs resources and time for its execution. And this demands of processes are handled by the CPU. In the given scenario, Mr. Sudesh Sharma, Linux expert, can be considered as a CPU, who solves the queries of either Faculty or Student by allocating proper resources to their individual demands and processing them by allocating them time accordingly. Now, Mr. Sharma, wants to provide priority for each query based upon its class, as well as, he wants to dedicate a fixed amount of time to every request. Thus in Operating System, if we divide the requests into two separate queues i.e.

Faculty and Student such that the first queue contains faculty queries has higher priority and the second contains student queries which has lower priority, then we can resolve the problem, by allocating them required resources based upon their priorities as done in the scheduling algorithm in operating systems.

THE END