OS REPORT

Student Name : SNEHA DARANGULA

Student ID : 11805214

EMAIL Address : snehad086@gmail.com

GitHub Link :github.com/Sneha-112

CODE :

There are 3 student processes and 1 teacher process. Students are supposed to do their assignments and they need 3 things for that pen, paper and question paper. The teacher has an infinite supply of all the three things. One students has pen, another has paper and another has question paper. The teacher places two things on a shared table and the student having the third complementary thing makes the assignment and tells the teacher on completion. The teacher then places another two things out of the three and again the student having the third thing makes the assignment and tells the teacher on completion. This cycle continues. WAP to synchronize the teacher and the students.

1) the resources of student processes are declared using struct key.
2) Using the datatype int , the struct key declares resources using count which will count the pen,paper,question paper.
3) The resources are aloted with some values.
4) By printing the code the user can get the resources allotted to the student 0 are paper and the resources allotted to the student 1 are question paper the resources allotted to the student 2 are pen
5) The student 2 has completed his job
6) The teacher process allot the resources to student processes will prints three times.
7) The process will be exited after 0.03989 seconds with return value 0

CODE SNIPPET:

```
#include<stdio.h>

#include<unistd.h>

struct resource{


        int pen;

        int paper;

        int q_paper;

        int allot_id;


};

struct t_resource{


        int pen_count ;

        int paper_count ;

        int q_paper_count ;

};


struct t_resource t[3];


void allotment(struct resource a){

        if(a.allot_id == 0){

                a.pen=1;

        }

        else if(a.allot_id == 1){
                a.paper=1;
        }
        else if(a.allot_id == 2){
                a.q_paper=1;
        }

}



int main(){
        struct resource s[3];

        int i,j,count=0;
        for(i=0;i<3;i++){
                s[i].allot_id=i;
                allotment(s[i]);
        }

        printf("\nResources alloted are                    :");
        for(i=0;i<3;i++){
                if(s[i].pen == 1)
                printf("\nResources alloted to student %d are      : pen ",i+1);

                if(s[i].paper)
                printf("\nResources alloted to student %d are      : paper ",i);
                if(s[i].q_paper == 1)
                printf("\nResources alloted to student %d are       : question paper ",i+1);
        }

        while(count != 3){
                for(i=0;i<3;i++){
                        if(i=0){
                                t[i].paper_count=1;
                                t[i].pen_count=1;
                                for(j=0;j<3;j++){
                                        if(s[j].q_paper==1){
                                                printf("\nStudent %d has completed his job !",j+1);
                                                count++;
                                        }
                                }
                        }

                        if(i=1){
                                t[i].q_paper_count=1;
                                t[i].pen_count=1;
                                for(j=0;j<3;j++){
                                        if(s[j].paper==1){
                                                printf("\nStudent %d has completed his job !",j+1);
                                                count++;
                        }
                    }
                }

                if(i=2){
                    t[i].paper_count=1;
                    t[i].q_paper_count=1;
                    for(j=0;j<3;j++){
                        if(s[j].pen==1){
                            printf("\nStudent %d has completed his job !",j+1);
                            count++;
                        }
                    }
                }
            }

        }
        return 0;

        }
```

Problem 2:

Researchers designed one system that classified interactive and noninteractive processes automatically by looking at the amount of terminal I/O. If a process did not input or output to the terminal in a 1-second interval, the process was classified as noninteractive and was moved to a lower-priority queue. In response to this policy, one programmer modified his programs to write an arbitrary character to the terminal at regular intervals of less than 1 second. The system gave his programs a high priority, even though the terminal output was completely meaningless.

1) The interactive process is the simplest way to work on a system. You log in, run commands which execute immediately.
2) The non interactive process is you don't have to do anything, and nothing shows up on the monitor.
3) The resptime and the type variable is declared in the main function,then the user will print the number of process and the user have to give the data.
4) If the user enter the 3 process then immediately the compiler asks the three processers response time.
5) Lets have example number of process:2
6) The response time in milliseconds:1
7) The response time in milliseconds:3thats how the program ends

CODE SNIPPET:

```
#include<stdio.h>

int main()

{

        int i, type[20],n;

        int resptime[20];

        printf("Number of process: ");

        scanf("%d",&n);

        printf("Enter the data\n");
```

```
for(i=0;i<n;i++)
{
	printf("Response time of P%d (in milliseconds): ",i);
	scanf("%d",&resptime[i]);
	if(resptime[i]<1000)
	{
		type[i]=1;
	}
	else
	{
		type[i]=0;
	}
}
printf("Process Number\tResponse Time\tType\t\tPriority");
for(i=0;i<n;i++)
{
	printf("\nP%d\t\t%dms\t\t",i,resptime[i]);
	if(type[i]==1)
	{
		printf("Interactive\tHigh");
	}
	else
	{
		printf("Non-Interactive\tLow");
	}
```

```
          }

  }
```