Name: Trần Quang Khải

ID:21520955

Class: IT007.N11.KHCL

OPERATING SYSTEM
LAB 4 Prepare

SUMMARY

Task		Status	Page
4.5	4	Done	2,3,4

Self-scrores: 95

Note: Export file to **PDF and name the file by following format:*
LAB X – <Student ID>.pdf

4.5.3

Mô phỏng giải thuật Round Robin

Code: https://github.com/TQKhai3002/Lab4

```c
void RR(struct process P[]){
    //Quantumtime
    float Quantum_time;
    printf("Enter Quantum time: ");
    scanf("%f",&Quantum_time);

    //Create a Queue
    pQueue Q = createQueue();
    //Time elapsed
    float time_elapsed = 0;
    //Check if done yet
    int done[num_process];
    for (int i = 0; i < num_process; i++) done[i] = 0;
    //QueueIndex
    int numP = 0;
    //Number of remaining processes
    int cntP = num_process;
    //While there are remaining processes
    while (cntP > 0){
        //While index < number of preocesses
        while (numP < num_process){
            //If process is not done yet
            if (done[numP] == 0){
                //Push into Queue
                push(Q,P[numP]);
                //Set time elapsed , add QueueIndex by 1 , move to next process
                time_elapsed = P[numP++].arr;
                break;
            }
            //If done , move to next process
            numP++;
        }
        //While there are processes in Queue
        while (!isEmpty(Q)){
            //Set index = process id
```

```c
            int idx = ((Q->front)->key).id;
            pop(Q);
            //If process is running first time
            if (P[idx].remain_time == P[idx].brust){
                //Calculate response time
                P[idx].rp_time = time_elapsed - P[idx].arr;
            }
            //Running time
            int rtime = P[idx].remain_time;
            if (rtime > Quantum_time) rtime = Quantum_time;
            //Update time elapsed
            time_elapsed += rtime;
            //Update remain time
            P[idx].remain_time -= rtime;
            //While index < number of processes and process has arrived
            while (numP < num_process && P[numP].arr <= time_elapsed) {
                //Push process from Queue
                push(Q,P[numP]);
                //Increse QuqueIndex
                numP++;
            }
            //If finish running
            if (P[idx].remain_time == 0) {
                //Set finish time, turnaround time, waiting time
                P[idx].finish = time_elapsed;
                P[idx].turnround_time = P[idx].finish - P[idx].arr;
                P[idx].wait_time = P[idx].turnround_time - P[idx].brust;
                done[idx] = 1;
                //Decrease No Reamaning Processes
                cntP--;
            }
            //If not finish remove from Queue
            else push(Q,P[idx]);

        }
```

```
    }
    show(P);


}



void reset(struct process P[]){
    for (int i = 0; i < num_process; i++){
        P[i].finish = P[i].turnround_time = P[i].wait_time = P[i].rp_time = 0;
        P[i].remain_time = P[i].brust;
    }
}

void sort(struct process P[]){
    for (int i = 0; i < num_process - 1; i++)
            for (int j = i + 1; j < num_process; j++)
            if (P[j].arr < P[i].arr)swap(&P[i],&P[j]);
            else if (P[i].arr == P[j].arr && P[j].id < P[i].id) swap(&P[i],&P[j]);
}

void Process_Sceduling_Algo(){
    struct process P[N];
    printf("Enter number of process: ");
    scanf("%d",&num_process);

    for (int i = 0; i < num_process; i++){
        printf("Enter Process Name: "); scanf("%s", P[i].name);
        printf("Enter Arrival Time: "); scanf("%f", &P[i].arr);
        printf("Enter Burst Time  : "); scanf("%f", &P[i].brust);
        P[i].finish = P[i].turnround_time = P[i].wait_time = P[i].rp_time = 0;
        P[i].remain_time = P[i].brust;
        P[i].id = i;
        printf("\n");
    }
```

```
    }
        reset(P);
        sort(P);
        RR(P);
}



int main(){
    Process_Sceduling_Algo();
    return 0;
}
```

Input:

Process	Arrival Time	Burst Time
P1	0	12
P2	2	7
P3	5	8
P4	9	3
P5	12	6

Output:

```
| Process Name | Arrival Time | Burst Time | Completion Time | Turnaround Time | Waiting Time | Response Time |
|P1            |0.000000      |12.000000   |30.000000        |30.000000        |18.000000     |0.000000       |
|P2            |2.000000      |7.000000    |19.000000        |17.000000        |10.000000     |2.000000       |
|P3            |5.000000      |8.000000    |34.000000        |29.000000        |21.000000     |7.000000       |
|P4            |9.000000      |3.000000    |22.000000        |13.000000        |10.000000     |10.000000      |
|P5            |12.000000     |6.000000    |36.000000        |24.000000        |18.000000     |10.000000      |
Average waiting time : 15.40
Average turn around time: 22.60
Average respone time: 5.80
khai21520955@ubuntu:~$
```