SCI19 2243 Operating Systems (1/2568)

สาขาวิชาคณิตศาสตร์และภูมิสารสนเทศ สำนักวิชาวิทยาศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

รหัสนักศึกษา B6638146 ชื่อ-สกุล เจษฎา ธัญญะ

Assignment ครั้งที่ 5 – Basic CPU Scheduling

1. จงอธิบายการทำงานของ CPU scheduling แต่ละประเภท **สรุปใจความสำคัญ**

CPU scheduling	Summary
FCFS	เป็นอัลกอริธึมที่ให้โพรเซสที่มาถึงก่อนได้ทำงานก่อนจนเสร็จสิ้น ข้อเสียคืออาจทำให้โพรเซสสั้นๆ ต้องรอคิวนานหากมีโพรเซสยาวๆ มาก่อน
SJF	Non-Preemptive เลือกโพรเซสที่ใช้เวลาทำงานสั้นที่สุดทำก่อนจนเสร็จ โดยไม่สามารถถูกขัดจังหวะ Preemptive เลือกโพรเซสที่เหลือเวลาทำงานสั้นที่สุดทำก่อน และสามารถสลับไปทำงานที่สั้นกว่าได้ทันที

2. เขียนโปรแกรมภาษาซีเพื่อจำลองการทำงานของ CPU Scheduling แบบ FSFC
   - นักศึกษาต้องแก้ไข ปรับปรุงการทำงานของโปรแกรมจำลองนี้ให้สมบูรณ์เป็นไปตามลำดับการทำงานของ FCFS
   - วิธีการรันโค้ดผ่าน Terminal
     - Compile ด้วยคำสั่ง gcc **fcfs.c** -o **fcfs** (ตัวหนาคือส่วนที่สามารถเปลี่ยนชื่อได้)
     - Run ด้วยคำสั่ง ./**fcfs**

```
void input(int* total_process, float burst_time[]){
    int count;

    //How many processes execute
    printf("Enter The Number of Processes To Execute:\t");
    ……(1)….……..

    //How long each process will execute in CPU
    printf("\nEnter The Burst Time of Processes:\n\n");
    for(count = 0; count < *total_process; count++)
    {
        printf("Process [%d]:", count+1);
        scanf("%f", &burst_time[count]);
    }
}
void calculateWaitingTime(int total_process, float burst_time[],
```

```
float waiting_time[]){
    int count,j;
    waiting_time[0] = 0;

        //Calculate waiting time
        //Loop for all processes
        for(……(2)….……){
                //Initialize a waiting time array for each process
                ……(3)….…….
                // Loop to get previous processes' burst time
                for(……(4)……..……){
                // Calculate waiting time of the current process
                        ……(5)……..…….
                }
        }
}
```

```
Enter The Number of Processes To Execute:       3

Enter The Burst Time of Processes:

Process [1]:10
Process [2]:20
Process [3]:30

Process         Burst Time      Waiting Time    Turnaround Time

Process [1]             10.00           0.00            10.00
Process [2]             20.00           10.00           30.00
Process [3]             30.00           30.00           60.00

Average Waiting Time = 13.333333
Average Turnaround Time = 33.333332
```

```
Enter the Total Number of Processes:    4

Enter Details of 4 Processes

Enter Arrival Time:     0
Enter Burst Time:       8

Enter Arrival Time:     1
Enter Burst Time:       4

Enter Arrival Time:     2
Enter Burst Time:       9

Enter Arrival Time:     3
Enter Burst Time:       5


Average Waiting Time:   6.500000
Average Turnaround Time:        13.000000
```

3. เขียนโปรแกรมภาษาซีเพื่อจำลองการทำงานของ CPU Scheduling แบบ Preemptive SJF (สร้างไฟล์ชื่อว่า sjf.c)

```
 void input(int* total_process, int burst_time[], int arrival_time[], int
temp[]){
      int i;
      printf("\nEnter the Total Number of Processes:\t");
      ……(1)……
      printf("\nEnter Details of %d Processes\n", *total_process);
      for(i = 0; i < *total_process; i++)
      {
            printf("\nEnter Arrival Time:\t");
            scanf("%d", &arrival_time[i]);
            printf("Enter Burst Time:\t");
            scanf("%d", &burst_time[i]);
            temp[i] = burst_time[i]; //Keep actual burst time
      }
 }

  void calculateWaitingTime(int total_process, int burst_time[], int
arrival_time[], double *turnaround_time, double *wait_time, int temp[]){
     int i, smallest, count = 0, curr_time;
     double end;
     burst_time[MAX_PROCESS-1] = 9999;
```

```
      for(curr_time = 0; count != total_process; curr_time++)
      {
            smallest = MAX_PROCESS-1;
            for(i = 0; i < total_process; i++)
            {
 //Detect the process with the smallest burst time when its arrival time
does not exceed current time and its burst time is less than the smallest
process's burst time and is higher than zero
                  if(……(2)……)
                  {
      //Assign the current process as the smallest burst time process
                        ……(3)……
                  }
            }
             //The burst time of the process executed in CPU decreases
             ……(4)……
            if(burst_time[smallest] == 0)
            {
                  count++;
                  end = curr_time + 1; //Present time
              //Calculate waiting and turnaround time

                  *wait_time =  ……(5)……
                  *turnaround_time = *turnaround_time + ……(6)……
            }
      }
 }
```