آزمایشگاه سیستم عامل

گزارش آزمایش شماره ۸

نام استاد: مهندس حمیدرضا کیخا

تاریخ: ۳ / ۱۰ / ۱۴۰۰

نام و نام‌خانوادگی: نگار کرمی

شماره دانشجویی: ۹۷۲۲۰۳۹

نام و نام‌خانوادگی: مجید نامی

شماره دانشجویی: ۹۷۲۸۰۹۵

دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

بخش اول: **First Come First Serve**

خروجی:

```
first come first serve(FCFS)

Enter number of the process :
10
Enter Arrival time and Burst time of the process
Enter AT for process #0:7
Enter BT for process #0:3
Enter AT for process #1:1
Enter BT for process #1:5
Enter AT for process #2:6
Enter BT for process #2:2
Enter AT for process #3:3
Enter BT for process #3:6
Enter AT for process #4:4
Enter BT for process #4:3
Enter AT for process #5:9
Enter BT for process #5:1
Enter AT for process #6:5
Enter BT for process #6:3
Enter AT for process #7:6
Enter BT for process #7:1
Enter AT for process #8:7
Enter BT for process #8:2
Enter AT for process #9:8
Enter BT for process #9:5

Process | Waiting_time   | TurnA_time
  1     |          7     |         -4
  2     |          2     |          7
  3     |          2     |          4
  4     |          7     |         13
  5     |         12     |         15
  6     |         10     |         11
  7     |         15     |         18
  8     |         17     |         18
  9     |         17     |         19
 10     |         18     |         23

Average waiting time is : 10.700000
Average turn around time is : 12.400000
```

Processes ID	Burst Time	Arrive Time	Waiting Time	Turnaround Time
2	3	7	12	18
6	5	1	5	4
8	2	6	2	5
0	6	3	22	12
7	3	4	32	25
4	1	9	31	20
9	3	5	25	21
1	1	6	47	41
3	2	7	49	44
5	5	8	67	60

بخش دوم: Shortest Job First

کد:

در الگوریتم shortest job first ابتدا به صف خود نگاه می‌کنیم و بعد لازم است در هر زمان که می‌خواهیم پردازه‌ها را از داخل آن انتخاب کنیم به این نکته توجه داشته باشیم که آن پردازه کم ترین زمان اجرا (burst time) را در میان بقیه دارد و آن را به پردازنده برای اجرا تحویل دهیم.

همچنین این الگوریتم به صورت انحصاری می‌باشد و در واقع non preemptive می‌باشد. و اگر نوبت به پردازه‌ای رسید و cpu را به آن دادیم تا زمانی که کارش به انتها نرسد نوبت به پردازه دیگر نمی‌رسد.

تابع Sort by burst time برای مرتب سازی است:

```c
typedef struct Process Process;

struct Process {
    int pid;
    int bt;
    int wt;
    int tt;
};

void sortByBurstTime(int taskNum, Process *processes) {
    for (int i = 0; i < taskNum; i++) {
        for (int j = i + 1; j < taskNum; j++) {
            if (processes[i].bt > processes[j].bt) {
                Process temp = processes[i];
                processes[i] = processes[j];
                processes[j] = temp;
            }
        }
    }
}
```

برای محاسبه مقادیر WT و TT به صورت زیر عمل می‌کنیم:

```
int main() {
    int taskNum = 0;
    printf("Enter the number of tasks:\n");
    scanf("%d", &taskNum);
    Process processes[taskNum];
    for (int i = 0; i < taskNum; i++) {
        printf("Enter the burst time of P%d: ", i);
        processes[i].pid = i;
        scanf("%d", &processes[i].bt);
        processes[i].wt = i;
        processes[i].tt = i;
    }

    sortByBurstTime(taskNum, processes);


    processes[0].wt = 0;
    processes[0].tt = processes[0].bt;
    for (int i = 1; i < taskNum; i++) {
        processes[i].wt = processes[i - 1].wt + processes[i - 1].bt;
        processes[i].tt = processes[i].wt + processes[i].bt;
    }

    float sumTT = 0, sumWT = 0;
    printf("Process     BT          WT          TT\n");
    for (int i = 0; i < taskNum; i++) {
        printf("P%-11d%-12d%-12d%-12d\n", processes[i].pid, processes[i].bt, processes[i].wt, processes[i].tt);
        sumTT += (float)processes[i].tt;
        sumWT += (float)processes[i].wt;
    }

    printf("Average waiting time: %.2f\n Average Turn around time: %.2f\n", sumWT / taskNum, sumTT / taskNum);
    return 0;
}
```

خروجی:

```
negar@ubuntu:~/Desktop$ ./sjf
Enter the number of tasks:
10
Enter the burst time of P0: 18
Enter the burst time of P1: 10
Enter the burst time of P2: 20
Enter the burst time of P3: 6
Enter the burst time of P4: 3
Enter the burst time of P5: 15
Enter the burst time of P6: 25
Enter the burst time of P7: 9
Enter the burst time of P8: 12
Enter the burst time of P9: 1
Process     BT          WT          TT
P9          1           0           1
P4          3           1           4
P3          6           4           10
P7          9           10          19
P1          10          19          29
P8          12          29          41
P5          15          41          56
P0          18          56          74
P2          20          74          94
P6          25          94          119
Average waiting time: 32.80
 Average Turn around time: 44.70
negar@ubuntu:~/Desktop$
```

Processes	Burst Time	Waiting Time	Turn Around Time
P9	1	0	1
P4	3	1	4
P3	6	4	10
P7	9	10	19
P1	10	19	29
P8	12	29	41
P5	15	41	56
P0	18	56	74
P2	20	74	94
P6	25	94	119

بخش سوم: Priority

کد:

```c
#include <stdio.h>

typedef struct {
    int pid;
    int bt;
    int wt;
    int tt;
    int pr;
} process;

#define sort(array, n, member) (                        \
{                                                       \
    int c, d;                                           \
    process swap;                                       \
    for (c = 0 ; c < n - 1; c++)                        \
    {                                                   \
        for (d = 0 ; d < n - c - 1; d++)                \
        {                                               \
            if (array[d].member > array[d+1].member)     \
            {                                           \
                swap       = array[d];                  \
                array[d]   = array[d+1];                \
                array[d+1] = swap;                      \
            }                                           \
        }                                               \
    }                                                   \
} )

#define calc_average(processes, n, member)( \
{                                           \
    int sum = 0,i;                          \
    for ( i = 0; i < n; ++i)                \
        sum += processes[i].member;         \
    (((double) sum) / n);                   \
    }                                       \
)

```

```
38
39  void print_process(process p, int alg_num)
40  {
41      printf("PID: %3d\t|\t", p.pid);
42      printf("Burst Time: %3d\t|\t", p.bt);
43      printf("Waiting Time: %3d\t|\t", p.wt);
44      printf("Turnaround Time: %3d\t|\t", p.tt);
45      printf("Priority: %3d\n", p.pr);
46  }
47
48
49  int main()
50  {
51      int n, alg_num;
52      int j,i;
53      process processes[100];
54      printf("Priority\n\n");
55      alg_num=1;
56      printf("Enter number of processes: ");
57      scanf("%d", &n);
58      printf("\n");
59      for ( j = 0; j < n; ++j)
60      {
61          printf("Enter burst time for process #%d: ", j);
62          int burst_time;
63          scanf("%d", &burst_time);
64
65
66          printf("Enter priority number for process #%d: ", j);
67          int prior_num;
68          scanf("%d", &prior_num);
69          processes[j].pr = prior_num;
70
71          processes[j].pid = j;
72          processes[j].bt = burst_time;
73      }
74
```

```
75
76      if (alg_num == 1)
77          sort(processes, n, pr);
78      else
79      {
80          printf("Error!\n");
81          return -1;
82      }
83
84      processes[0].wt = 0;
85      processes[0].tt = processes[0].bt;
86
87      for ( i = 1; i < n; ++i)
88      {
89          processes[i].wt = processes[i-1].tt;
90          processes[i].tt = processes[i].wt + processes[i].bt;
91      }
92
93      printf("\n");
94      for ( i = 0; i < n; ++i)
95          print_process(processes[i], alg_num);
96
97      printf("\nAverage waiting time is: %g\n", calc_average(processes, n, wt));
98      printf("Average turnaround time is: %g\n", calc_average(processes, n, tt));
99      return 0;
100 }
101
```

در الگوریتم priority، که به همان صورت non preemptive می‌باشد و لازم است تا sort کنیم منتها sort کردن بر روی priority ها انجام می‌شود و براساس این که کدام یک اولویت بالاتری دارد آن را به پردازنده می‌دهیم. در ضمن اولویت بالاتر را اینگونه در نظر گرفتیم که شماره اولویت مقدار کم تری داشته باشد.

خروجی:

```
Priority

Enter number of processes: 10

Enter burst time for process #0: 18
Enter priority number for process #0: 6
Enter burst time for process #1: 10
Enter priority number for process #1: 9
Enter burst time for process #2: 20
Enter priority number for process #2: 3
Enter burst time for process #3: 6
Enter priority number for process #3: 10
Enter burst time for process #4: 3
Enter priority number for process #4: 7
Enter burst time for process #5: 15
Enter priority number for process #5: 11
Enter burst time for process #6: 25
Enter priority number for process #6: 4
Enter burst time for process #7: 9
Enter priority number for process #7: 6
Enter burst time for process #8: 12
Enter priority number for process #8: 5
Enter burst time for process #9: 1
Enter priority number for process #9: 7

ID:   2       |       Burst Time:  20 |       Waiting Time:   0       |       Turnaround Time:  20    |       Priority:   3
ID:   6       |       Burst Time:  25 |       Waiting Time:  20       |       Turnaround Time:  45    |       Priority:   4
ID:   8       |       Burst Time:  12 |       Waiting Time:  45       |       Turnaround Time:  57    |       Priority:   5
ID:   0       |       Burst Time:  18 |       Waiting Time:  57       |       Turnaround Time:  75    |       Priority:   6
ID:   7       |       Burst Time:   9 |       Waiting Time:  75       |       Turnaround Time:  84    |       Priority:   6
ID:   4       |       Burst Time:   3 |       Waiting Time:  84       |       Turnaround Time:  87    |       Priority:   7
ID:   9       |       Burst Time:   1 |       Waiting Time:  87       |       Turnaround Time:  88    |       Priority:   7
ID:   1       |       Burst Time:  10 |       Waiting Time:  88       |       Turnaround Time:  98    |       Priority:   9
ID:   3       |       Burst Time:   6 |       Waiting Time:  98       |       Turnaround Time: 104    |       Priority:  10
ID:   5       |       Burst Time:  15 |       Waiting Time: 104       |       Turnaround Time: 119    |       Priority:  11

Average waiting time is: 65.8
Average turnaround time is: 77.7
```

Process ID	Burst Time	Waiting Time	Turnaround Time	Priority
20	0	20	2	3
45	20	25	6	4
57	45	12	8	5
75	57	18	0	6
84	75	9	7	6
87	84	3	4	7
88	87	1	9	7
98	88	10	1	9
104	98	6	3	10
119	104	15	5	11

بخش چهارم: Round Robin

کد:

برای محاسبه مقادیر و شبیه سازی، الگوریتم را به صورت زیر پیاده سازی کرده‌ایم:

به اینصورت عمل می کنیم که در یک لوپ بینهایت بررسی می‌کنیم که اگر نوبت هر فرآیند بود آیا یک کوانتوم را کامل مصرف می کند یا خیر و بر اساس آن زمان باقی مانده از فرآیند را تغییر می دهیم.

همچنین از متغیری به نام time استفاده شده است که در اصل زمان است و هنگامی که فرآیندی شروع می‌شود یا پایان می‌یابد، متغیر time را فیلد مربوطه به آن فرآیند ذخیره می‌کنیم.

```
typedef struct Process Process;

struct Process {
    int pid;
    int bt;
    int wt;
    int tt;
    int startPoint;
    int endPoint;
    int remainTime;
};

int allDone(int taskNum, Process *processes) {
    for (int i = 0; i < taskNum; i++) {
        if (processes[i].remainTime != 0)
            return 0;
    }
    return 1;
}

void simulate(int taskNum, Process *processes, int timeQ) {
    int time = 0;
    for (int i = 0;; i++) {
        if (allDone(taskNum, processes))
            break;
        int turn = i % taskNum;
        if (processes[turn].remainTime == 0) {
            continue;
        }

        if (processes[turn].remainTime > timeQ) {
            if (processes[turn].startPoint == -1)
                processes[turn].startPoint = time;
            time += timeQ;
            processes[turn].remainTime -= timeQ;
        } else {
            time += processes[turn].remainTime;
            processes[turn].remainTime = 0;
            if (processes[turn].endPoint == -1)
                processes[turn].endPoint = time;
        }
    }
```

پس از اتمام شبیه سازی بالا در ادامه برای محاسبه TT و WT می‌توان به صورت زیر عمل کرد:

```
    for (int i = 0; i < taskNum; i++) {
        processes[i].wt = processes[i].endPoint - processes[i].bt;
        processes[i].tt = processes[i].endPoint;
    }
}

int main() {
    int taskNum = 0, timeQ = 0;
    printf("Enter the number of tasks:\n");
    scanf("%d", &taskNum);
    Process processes[taskNum];
    for (int i = 0; i < taskNum; i++) {
        printf("Enter the burst time of P%d: ", i);
        processes[i].pid = i;
        scanf("%d", &processes[i].bt);
        processes[i].wt = i;
        processes[i].tt = i;
        processes[i].remainTime = processes[i].bt;
        processes[i].startPoint = -1;
        processes[i].endPoint = -1;
    }

    printf("Enter Time quantum:\n");
    scanf("%d", &timeQ);


    simulate(taskNum, processes, timeQ);

    float sumTT = 0, sumWT = 0;
    printf("Process     BT          WT          TT\n");
    for (int i = 0; i < taskNum; i++) {
        printf("P%-11d%-12d%-12d%-12d\n", processes[i].pid, processes[i].bt, processes[i].wt, processes[i].tt);
        sumTT += (float)processes[i].tt;
        sumWT += (float)processes[i].wt;
    }

    printf("Average waiting time: %.2f\n Average Turn around time: %.2f\n", sumWT / taskNum, sumTT / taskNum);
    return 0;

}
```

مقدار WT زمانی است که فرآیند در انتظار بوده پس اگر از زمان پایان، زمان اجرا روی CPU را کم کنیم به WT می‌رسیم.

مقدار TT زمانی است که فر آیند وارد صف شده تا زمانی که به اتمام رسیده، پس این عدد همان زمان پایان این فرآیند می‌باشد.

خروجی:

```
negar@ubuntu:~/Desktop$ gcc RR.c -o rr
negar@ubuntu:~/Desktop$ ./rr
Enter the number of tasks:
10
Enter the burst time of P0: 18
Enter the burst time of P1: 10
Enter the burst time of P2: 20
Enter the burst time of P3: 6
Enter the burst time of P4: 3
Enter the burst time of P5: 15
Enter the burst time of P6: 25
Enter the burst time of P7: 9
Enter the burst time of P8: 12
Enter the burst time of P9: 1
Enter Time quantum:
10
Process     BT          WT          TT
P0          18          69          87
P1          10          10          20
P2          20          77          97
P3          6           30          36
P4          3           36          39
P5          15          87          102
P6          25          94          119
P7          9           59          68
P8          12          102         114
P9          1           78          79
Average waiting time: 64.20
 Average Turn around time: 76.10
```

Processes	Burst Time	Waiting Time	Turnaround Time
P0	18	69	87
P1	10	10	20
P2	20	77	97
P3	6	30	36
P4	3	36	39
P5	15	87	102
P6	25	94	119
P7	9	59	68
P8	12	102	114
P9	1	78	79

مقایسه کلی:

Algorithm	Average waiting time	Average turnaround time
first come first serve(FCFS)	10.7	12.4
shortest job first	32.8	44.7
Priority	65.8	77.7
Round Robin	64.2	76.1