به نام خدا

گزارش آزمایش هشتم آزمایشگاه سیستم های عامل

زهرا رحیمی

شماره دانشجوئی: ۹۸۳۱۰۲۶

استاد آزمایشگاه: سرکار خانم حسینی

پاییز ۱۴۰۰

بخش اول: الگوریتم First Come First Serve(FCFS):

```c
#include <stdio.h>
struct process{
    int pid;
    int bt;
    int wt,tt;
};
int main() {
    int process_num;
    scanf("%d", &process_num);
    struct process p[process_num];

    for (int i = 0; i<process_num; i++){
        int service_time;
        scanf("%d", &service_time);
        p[i].bt = service_time;
        p[i].pid = i;
    }
    p[0].wt = 0;
    p[0].tt = p[0].bt;
    for (int j = 1; j < process_num ; j++) {
        p[j].wt = p[j-1].tt; //waiting time
        p[j].tt = p[j].wt + p[j].bt; //process execution time
    }
    int avr_waiting = 0, avr_execution = 0;
    for (int k = 0; k < process_num; ++k) {
        avr_waiting += p[k].wt;
        avr_execution += p[k].tt;
    }
    printf("Average Waiting Time Is : %.1f\n",(float ) avr_waiting/process_num);
    printf("Average Execution Time Is : %.1f\n", (float ) avr_execution/process_num);
    return 0;
}
```

اگر سه فرآیند داشته باشیم که فرآیند اول burst time ۲۴ ثانیه داشته باشد و فرآیند دوم و سوم هر کدام ۳ ثانیه سرویس دهی شوند(مانند شکل زیر)، خروجی به شکل زیر خواهد شد:

Process	Burst Time
P_1	24
P_2	3
P_3	3

Suppose that the processes arrive in the order: P_1 , P_2 , P_3
The Gantt Chart for the schedule is:

Waiting time for P_1 = 0; P_2 = 24; P_3 = 27
Average waiting time: (0 + 24 + 27)/3 = 17

```
E:\uni\Term5\os\lab\8\cmake-build-debug\8.exe
3
24
3
3
Average Waiting Time Is : 17.0
Average Execution Time Is : 27.0

Process finished with exit code 0
```

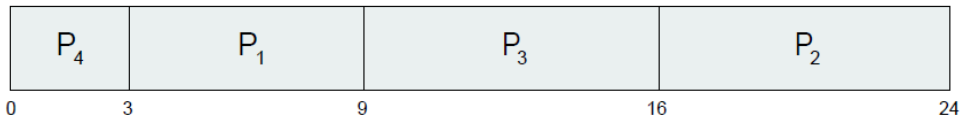
بخش دوم: الگوریتم Shortest Job First(SJF):

```c
#include <stdio.h>
#define INT_MAX 2147483647
struct process{
    int pid;
    int bt;
    int wt,tt;
};
int sort_shortest_job(int num, int burst_time[num]){
    int min_burst = INT_MAX, min_index;
    for (int i = 0; i < num; i++) {
        if(burst_time[i] < min_burst){
            min_burst = burst_time[i];
            min_index = i;
        }
    }
    burst_time[min_index] = INT_MAX;
    return min_burst;
}
int main(){
    int process_num;
    scanf("%d", &process_num);
    struct process p[process_num];
    int tmp_burst_time[process_num];

    for (int i = 0; i<process_num; i++){
        int service_time;
        scanf("%d", &service_time);
        tmp_burst_time[i] = service_time;
    }
    for (int j = 0; j < process_num; j++) {
        p[j].bt = sort_shortest_job(process_num, tmp_burst_time);
    }
    //Now The problem is like the previous one (FCFS)
    p[0].wt = 0;
    p[0].tt = p[0].bt;
    for (int j = 1; j < process_num ; j++) {
        p[j].wt = p[j-1].tt; //waiting time
        p[j].tt = p[j].wt + p[j].bt; //process execution time
    }
    int avr_waiting = 0, avr_execution = 0;
    for (int k = 0; k < process_num; ++k) {
        avr_waiting += p[k].wt;
        avr_execution += p[k].tt;
    }
    printf("Average Waiting Time Is : %.1f\n",(float ) avr_waiting/process_num);
    printf("Average Execution Time Is : %.1f\n", (float ) avr_execution/process_num);

}
```

اگر چهار فرآیند داشته باشیم که به ترتیب دارای burst time های ۶، ۸، ۷ و ۳ باشند (مانند شکل زیر)، خروجی به شکل زیر خواهد شد:

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

SJF scheduling chart



Average waiting time = (3 + 16 + 9 + 0) / 4 = 7

خروجی همان طور که انتظار می رفت به ترتیب زیر است:

```
E:\uni\Term5\os\lab\8\cmake-build-debug\8.exe
4
6
8
7
3
Average Waiting Time Is : 7.0
Average Execution Time Is : 13.0

Process finished with exit code 0
```

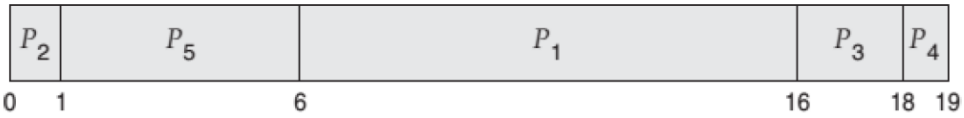
بخش سوم: الگوریتم اولویت دار(priority)

```c
#include <stdio.h>
#define INT_MAX 2147483647
struct process{
    int pid;
    int bt;
    int wt,tt;
    int prio;
};
int find_high_prio(int num, int *tmp_burst_time, int *tmp_priority){
    int min_prio = INT_MAX, min_index;
    for (int i = 0; i < num; i++) {
        if(tmp_priority[i] < min_prio){
            min_prio = tmp_priority[i];
            min_index = i;
        }
    }
    tmp_priority[min_index] = INT_MAX;
    return tmp_burst_time[min_index];
}
int main(){
    int process_num;
    scanf("%d", &process_num);
    struct process p[process_num];
    int tmp_burst_time[process_num];
    int tmp_priority[process_num];

    for (int i = 0; i<process_num; i++){
        int service_time, prio;
        scanf("%d", &service_time);
        scanf("%d", &prio);
        tmp_burst_time[i] = service_time;
        tmp_priority[i] = prio;
    }
    for (int j = 0; j < process_num; j++) {
        p[j].bt = find_high_prio(process_num, tmp_burst_time, tmp_priority);
    }//Now The problem is like FCFS
    p[0].wt = 0;
    p[0].tt = p[0].bt;
    for (int j = 1; j < process_num ; j++) {
        p[j].wt = p[j-1].tt; //waiting time
        p[j].tt = p[j].wt + p[j].bt; //process execution time
    }
    int avr_waiting = 0, avr_execution = 0;
    for (int k = 0; k < process_num; ++k) {
        avr_waiting += p[k].wt;
        avr_execution += p[k].tt;
    }
    printf("Average Waiting Time Is : %.1f\n",(float ) avr_waiting/process_num);
    printf("Average Execution Time Is : %.1f\n", (float ) avr_execution/process_num);
    return 0;
}
```

اگر بخواهیم ۵ فرآیند را با اولویت ها و زمان های سرویس دهی زیر با الگوریتم اولویت‌دار زمان بندی کنیم:

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

Priority scheduling Gantt Chart



Average waiting time = 8.2 msec

نتیجه به صورت زیر خواهد بود:

```
E:\uni\Term5\os\lab\8\cmake-build-debug\8.exe
5
10
3
1
1
2
4
1
5
5
2
Average Waiting Time Is : 8.2
Average Execution Time Is : 12.0
```

بخش امتیازی: الگوریتم زمان بندی Shortest Remaining Time First(SRT):

```c
#include <stdio.h>
#define INT_MAX 2147483647
struct process{
    int pid;
    int bt;
    int wt,tt;
    int at;
};
int pre_process, pre_time;

int rem_shortest_job(int num, struct process *p, int curr_time){
    int min_burst = INT_MAX, min_index = INT_MAX;
    if (curr_time > 0){
        p[pre_process].bt = p[pre_process].bt -(curr_time-pre_time);
    }
    for (int i = 0; i <= num; i++) {
        if(p[i].at <= curr_time) {
            if (p[i].bt > 0 && p[i].bt < min_burst) {
                min_burst = p[i].bt;
                min_index = p[i].pid;
            }
        }
    }
    pre_process = min_index;
    pre_time = curr_time;
}
```

```c
int main(){
    int process_num;
    int burst_time_sum = 0;
    scanf("%d", &process_num);
    struct process p[process_num];
    for (int i = 0; i<process_num; i++){
        int service_time, arrival_time;
        scanf("%d", &arrival_time);
        scanf("%d", &service_time);
        p[i].bt = service_time;
        burst_time_sum += service_time;
        p[i].at = arrival_time;
        p[i].pid = i;
    }
    for (int j = 0; j < burst_time_sum; j++) {
        if (p[j].at == j) {
            rem_shortest_job(process_num, p, j);
            printf("%d  ", pre_process);
            printf("%d  \n", pre_time);
        }
    }//Now The problem is like FCFS
    p[0].wt = 0;
    p[0].tt = p[0].bt;
    for (int j = 1; j < process_num ; j++) {
        p[j].wt = p[j-1].tt; //waiting time
        p[j].tt = p[j].wt + p[j].bt; //process execution time
    }
    int avr_waiting = 0, avr_execution = 0;
    for (int k = 0; k < process_num; ++k) {
        avr_waiting += p[k].wt;
        avr_execution += p[k].tt;
    }
    printf("Average Waiting Time Is : %.1f\n",(int ) avr_waiting/process_num);
    printf("Average Execution Time Is : %.1f\n", (int ) avr_execution/process_num);
}
```

این الگوریتم با استفاده از یک فیلد arrival time (av) در استراکت فرآیند و موقع ورود فرآیند جدید چک کردن اینکه کدام فرآیند زمان باقی مانده کمتری دارد، پیاده سازی شده است. به نظر به طور کلی الگوریتم باید چنین شکلی داشته باشد. در ضمن در تابع rem_shortest_job که با ورود فرآیند جدید صدا زده می شود دو متغیر global برای نگهداری آیدی آخرین فرآیند انجام شده و آخرین تایمی که آن فرآیند سپری کرده است تعریف شده است.

متاسفانه خروجی مطلوب نیست اما چون الگوریتم را تا قسمتی انجام داده بودم در گزارش آوردم.