گزارش دستورکار هفتم آزمایشگاه سیستم‌های عامل، قسمت دوم

نگار موقتیان، ۹۸۳۱۰۶۲

در این آزمایش می‌خواهیم تعدادی پردازه را با توجه به زمان ورودی و burst time آن‌ها زمانبندی کرده و waiting time و turnaround time را برای هر یک از پردازه‌ها محاسبه کنیم. برای این کار از الگوریتم FCFS استفاده شده‌است. در این الگوریتم هر پردازه‌ای که زودتر وارد سیستم شود زودتر پردازش خواهد شد.

در برنامهٔ نوشته شده یک struct به نام job تعریف شده‌است که نمایندهٔ هر یک از پردازه‌های موجود در سیستم می‌باشد. هر job یک arrival_time یا زمان ورود، یک burst_time یا مدت زمانی که می‌خواهد از CPU استفاده کند و یک متغیر sliceable دارد که مشخص می‌کند این کار قابل انجام شدن به صورت تکه تکه هست یا خیر.

از آن‌جایی که در این برنامه از الگوریتم FCFS استفاده می‌شود نیاز است که تابعی برای sort کردن پردازه‌ها بر حسب زمان ورود آن‌ها تعریف کنیم. این تابع یک تابع سادهٔ sort است که از الگوریتم bubble sort استفاده می‌کند.

در main برنامهٔ نوشته شده ابتدا تعداد کاراها و پس از آن اطلاعات مربوط به آن‌ها را دریافت کرده و در یک آرایه ذخیره‌سازی می‌کنیم. سپس این کارها را بر اساس زمان ورودشان مرتب می‌کنیم.

حال تا زمانی که تمام کارها پردازش نشده‌اند ادامه می‌دهیم. متغیر time نشان‌دهندهٔ زمان فعلی، start_time نشان‌دهندهٔ زمانی که پردازهٔ فعلی کار خود را شروع کرده و متغیر curr نشان‌دهندهٔ این است که نوبت به اجرای کدام پردازه رسیده. همچنین یک متغیر از نوع Boolean داریم که مشخص می‌کند در حال حاضر پردازه‌ای در حال اجرا هست یا خیر.

درون while برنامه ابتدا بررسی می‌کنیم که پردازه‌ای در حال اجرا هست یا خیر. اگر نبود بررسی می‌کنیم که در زمان فعلی پردازهٔ فعلی وارد سیستم شده‌است یا خیر. اگر نشده بود باید همچنان صبر کنیم و اگر شده بود متغیر running را true کرده و start_time را برابر با زمان فعلی قرار می‌دهیم.

حال اگر پردازه‌ای در حال اجرا بود و زمان اجرا آن تمام شده بود turnaround time (کل زمانی که پردازه به سیستم وارد شده تا کار آن به طور کامل تمام شده) و waiting time (زمانی که پردازه درون سیستم و صف ready بوده اما CPU به آن نرسیده) آن را چاپ کرده و به سراغ پردازهٔ بعدی می‌رویم. همچنین متغیر running را false می‌کنیم تا در دور بعدی پردازهٔ بعدی انتخاب شود.

در نهایت نیز یک واحد به زمان فعلی اضافه کرده و به ابتدای حلقه باز می‌گردیم.

* در صورتی که بخواهیم از قابلیت تکه تکه شدن پردازه‌ها نیز استفاده کنیم می‌توانیم از الگوریتم‌ها قبضه‌ای مانند Round Robin استفاده کنیم. در این الگوریتم کافیست زمانی که مدت زمان time quantum سپری شده‌است بررسی کنیم که پردازه قابل تکه شدن هست یا خیر. اگر بود context switch را انجام می‌دهیم و در غیر این صورت به اجرای پردازه تا به انتها ادامه می‌دهیم.

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<stdbool.h>

struct job {
	int arrival_time;
	int burst_time;
	int sliceable; // 0 -> cannot be sliced, 1 -> can be sliced
};

#define MAX_PROCESS_COUNT 20

int n;
struct job jobs[MAX_PROCESS_COUNT];

// swap two jobs by reference (used in sort algorithm)
void swap(struct job* a, struct job* b) {
    struct job tmp = *a;
    *a = *b, *b = tmp;
}

// sort processes according to their arriving time
void sort() {
    for (int i=0; i<n-1; i++)
        for (int j=0; j<n-i-1; j++)
            if (jobs[j].arrival_time > jobs[j+1].arrival_time)
                swap(&jobs[j], &jobs[j+1]);
}

int main() {
	printf("Number of processes: ");
	scanf("%d", &n);

	printf("\nPlease enter 'arrival time' and 'burst time' corresponding to
each process\nand determine if it 'can be sliced (1) or not (0)':\n");
	for (int i=0; i<n; i++)
		scanf("%d%d%d", &jobs[i].arrival_time, &jobs[i].burst_time,
&jobs[i].sliceable);
```

```
	sort();

	printf("\nSchedule:\n");
	int time = 0, start_time = 0, curr = 0;
	bool running = false;
	while (curr < n) {
		if (!running && jobs[curr].arrival_time <= time) { // select a new
eligible job to run
			running = true;
			start_time = time;
		}
		if (running && time - start_time == jobs[curr].burst_time) {
			int tat = time - jobs[curr].arrival_time;
			int wt = start_time - jobs[curr].arrival_time;
			printf("%d -> %d : Process %d (turnaround time = %d, waiting
time = %d)\n", start_time, time, ++curr, tat, wt);
			running = false;
			continue;
		}
		time++;
	}

	return 0;
}
```

خروجی این برنامه به ازای مثال داده شده مانند شکل زیر می‌باشد.

```
E:\Uni\5th Semester\Operating Systems\Lab\Lab 07 - 2\codes\OSLab07_Pt.2_Scheduling.exe
Number of processes: 8

Please enter 'arrival time' and 'burst time' corresponding to each process
and determine if it 'can be sliced (1) or not (0)':
0 10 1
3 6 1
5 2 1
9 20 1
11 10 0
29 5 1
39 5 0
50 10 1

Schedule:
0 -> 10 : Process 1 (turnaround time = 10, waiting time = 0)
10 -> 16 : Process 2 (turnaround time = 13, waiting time = 7)
16 -> 18 : Process 3 (turnaround time = 13, waiting time = 11)
18 -> 38 : Process 4 (turnaround time = 29, waiting time = 9)
38 -> 48 : Process 5 (turnaround time = 37, waiting time = 27)
48 -> 53 : Process 6 (turnaround time = 24, waiting time = 19)
53 -> 58 : Process 7 (turnaround time = 19, waiting time = 14)
58 -> 68 : Process 8 (turnaround time = 18, waiting time = 8)

--------------------------------
Process exited after 12.18 seconds with return value 0
Press any key to continue . . .
```