دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

گزارش کار آزمایشگاه سیستم عامل – شماره ۷ بخش دوم

ارائه سرویس به پردازه ها

حسنا اویارحسینی – ۹۸۲۳۰۱۰

استاد درس: جناب آقای مهندس کیخا

نیمسال دوم سال تحصیلی ۰۱-۱۴۰۰

بخش ۱)

در این آزمایش قصد داریم تعدادی پردازه را با داشتن زمان های ورود آن ها به سیستم و همچنین cpu burst time هر یک به نحوی برنامه ریزی کنیم که بدون اتلاف وقت هر یک از آنها اجرا شوند.

پردازه ها به شرح زیر می باشند:

۱. پردازش اول در لحظه ۰ درخواست می دهد و به ۱۰ واحد زمانی زمان نیاز دارد تا پردازش خود را تکمیل نماید.(قابلیت اجرا بصورت قطعه قطعه شده دارد)
۲. پردازش دوم در لحظه ۳ درخواست می دهد و به ۶ واحد زمانی زمان نیاز دارد تا پردازش خود را تکمیل نماید.(قابلیت اجرا بصورت قطعه قطعه شده دارد)
۳. پردازش سوم در لحظه ۵ درخواست می دهد و به ۲ واحد زمانی زمان نیاز دارد تا پردازش خود را تکمیل نماید.(قابلیت اجرا بصورت قطعه قطعه شده دارد)
۴. پردازش چهارم در لحظه ۹ درخواست می دهد و به ۲۰ واحد زمانی زمان نیاز دارد تا پردازش خود را تکمیل نماید.(قابلیت اجرا بصورت قطعه قطعه شده دارد)
۵. پردازش پنجم در لحظه ۱۱ درخواست می دهد و به ۱۰ واحد زمانی زمان نیاز دارد تا پردازش خود را تکمیل نماید و قابلیت اجرا بصورت جزء به جزء ندارد و باید بصورت یکپارچه اجرا گردد.
۶. پردازش ششم در لحظه ۲۹ درخواست می دهد و به ۵ واحد زمانی زمان نیاز دارد تا پردازش خود را تکمیل نماید.(قابلیت اجرا بصورت قطعه قطعه شده دارد)
۷. پردازش هفتم در لحظه ۳۹ درخواست می دهد و به ۵ واحد زمانی نیاز دارد تا پردازش خود را تکمیل نماید و قابلیت اجرا بصورت جزء به جزء ندارد و باید بصورت یکپارچه اجرا گردد.
۸. پردازش هشتم در لحظه ۵۰ درخواست می دهد و به ۱۰ واحد زمانی زمان نیاز دارد تا پردازش خود را تکمیل نماید.(قابلیت اجرا بصورت قطعه قطعه شده دارد)

باتوجه به اینکه برخی از پردازه ها قابلیت قطعه قطعه شدن ولی بخی دیگر ندارند ما از یک الگوریتم non-preemptive استفاده میکنیم تا به مشکلی برای پردازه هایی که امکان قطعه شدن ندارند، نخوریم.

برای این کار الگوریتم FCFS (First come first serve) را پیاده سازی میکنیم:

```c
#include<stdio.h>
void findWaitingTime(int processes[], int n, int bt[],
                                int wt[], int at[])
{
    int service_time[n];
    service_time[0] = at[0];
    wt[0] = 0;
    for (int i = 1; i < n ; i++)
    {
        service_time[i] = service_time[i-1] + bt[i-1];
        wt[i] = service_time[i] - at[i];
        if (wt[i] < 0)
            wt[i] = 0;
    }
}
void findTurnAroundTime(int processes[], int n, int bt[],
                                    int wt[], int tat[])
{
    for (int i = 0; i < n ; i++)
        tat[i] = bt[i] + wt[i];
}
void findavgTime(int processes[], int n, int bt[], int at[])
{
    int wt[n], tat[n];

    findWaitingTime(processes, n, bt, wt, at);
    findTurnAroundTime(processes, n, bt, wt, tat);

    printf("Processes \t Burst Time \t Arrival Time \t Waiting Time \t Turn-
Around Time \t Completion Time \n");
    int total_wt = 0, total_tat = 0;
    for (int i = 0 ; i < n ; i++)
    {
        total_wt = total_wt + wt[i];
        total_tat = total_tat + tat[i];
        int compl_time = tat[i] + at[i];
        printf("%d\t\t\t%d\t\t%d\t\t%d\t\t%d\t\t\t%d\n", i+1, bt[i], at[i],
wt[i], tat[i], compl_time);
    }

    printf("Average waiting time = %f", (float)total_wt / (float)n);
    printf("\nAverage turn around time = %f", (float)total_tat / (float)n);
}
```

در این الگوریتم ابتدا waiting time را برای هر پردازه حساب میکنیم، به این صورت که آرایه ای تحت عنوان Service_time میسازیم که در واقع زمان سرویس دهی به هر پردازه را مشخص میکند، زمان سروری دهی به هر پردازه برابر با زمان سرویس دهی به پردازه قبلی به علاوه زمان cpu burst پردازه قبلی می باشد (زیرا پردازه ها به ترتیب زمان ورود در همان ابتدا وارد آرایه ها شده اند) یعنی وقتی کار پردازه قبلی تمام شده. با بدست آوردن زمان ارائه سرویس waiting time که میشود زمان ورود تا زمان ارائه سرویس را میتوانیم محاسبه کنیم و اگر زمان انتظار یک پردازه منفی شد یعنی قبل از اینکه پردازنده را به این پردازه اختصاص دهیم پردازه در حالت ready بوده است پس wait نکرده.

در نهایت نیز Turn around time که همان مدت زمان وجود پردازه از ابتدای ورودی تا اتمام آن بوده است را حساب میکنیم که چون از الگوریتم FCFS استفاده کردیم این مقدار برابر با زمان wait + burst است چون هر چردازه با گرفتن cpu کار خود را به صورت کامل انجام میدهد.

خروجی این کد به صورت زیر خواهد بود:

```
Average turn around time = 20.375000lab@ubuntu:~/Desktop/HW7-2$ gcc FCFS.c -o FCFS
lab@ubuntu:~/Desktop/HW7-2$ ./FCFS
Processes        Burst Time      Arrival Time    Waiting Time    Turn-Around Time        Completion Time
1                       10              0               0               10                      10
2                       6               3               7               13                      16
3                       2               5               11              13                      18
4                       20              9               9               29                      38
5                       10              11              27              37                      48
6                       5               29              19              24                      53
7                       5               39              14              19                      58
8                       10              50              8               18                      68
Average waiting time = 11.875000
lab@ubuntu:~/Desktop/HW7-2$
```

- ایراد این الگوریم این است که ممکن است دچار پدیده کاروان شویم یعنی پردازه هایی با cpu bust پایین پشت پردازه هایی با Cpu burst بالا گیر کنند.