به نام خدا

گزارش آزمایش هشتم
آزمایشگاه سیستم عامل

استاد : حسینی

سید سعید شفیعی

آذر ۱۴۰۰

بخش اول : پیاده سازی الگوریتم First Come-First Served

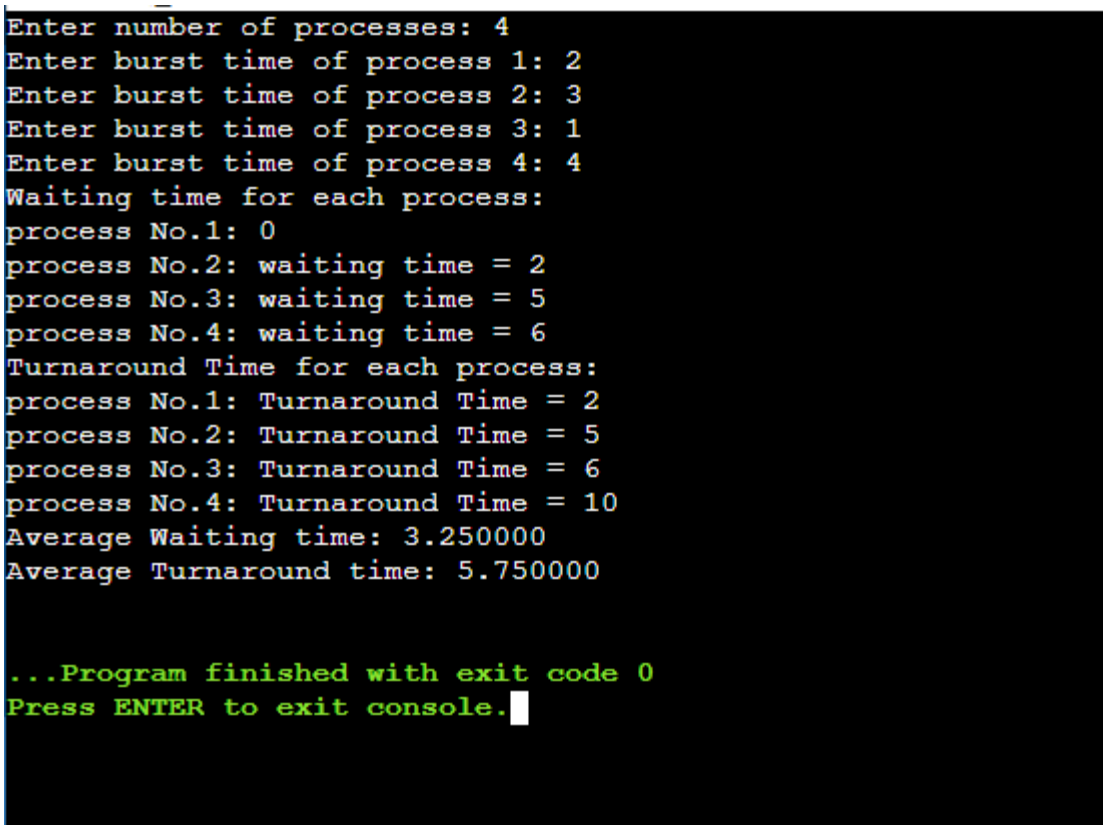
در ورودی تعداد پردازه و مدت زمان اجرای آن دریافت می شود. Waiting Time و Turnaround Time و میانگین این دو محاسبه می شود. پیاده سازی مربوط به همه الگوریتم ها به صورت جداگانه در فایل زیپ موجود است. نتیجه کار به صورت زیر است.

از آنجایی که این الگوریتم ترتیب اجرای پردازه ها ، همان ترتیب ورود آنهاست پس نیاز به کار اضافی برای تغییر جای پردازه های درون صف نداریم. تنها از ۴ تابع مختلف برای بدست آوردن زمان های خواسته شده استفاده شده است.

برای بدست آوردن زمان های خواسته شده از روابط زیر استفاده شده است:

p[i].waiting_time = p[i - 1].waiting_time + p[i - 1].burst_time;

p[i].turnaround_time = p[i].waiting_time + p[i].burst_time;



بخش دوم : پیاده سازی الگوریتم Shortest Job First

این بخش هم مانند بخش قبل در ورودی تعداد پردازه و مدت زمان اجرای آن دریافت می شود. Waiting Time و Turnaround Time و میانگین این دو محاسبه می شود. نکته خاصی که وجود دارد این است که چون در دستور کار ذکر شده است که پردازه اول Waiting Time صفر دارد و به محض ورود پردازنده در اختیارش قرار می گیرد، پس پردازه اول مورد مناسبی برای مقایسه نیست.

با مقایسه پردازه های دوم و سوم با زمان های اجرای به ترتیب ۴ و ۱، کاملا مشخص است که پردازه سوم ابتدا انجام شده و پس از آن پردازه دوم به پایان رسیده که این به معنی درستی عملکرد الگوریتم ماست.

بخش مرتب سازی زمان های اجرا، توسط الگوریتم مرتب سازی حبابی صورت گرفته و مرتب سازی از پردازه دوم صورت گرفته است. چون همانطور که گفته شد پردازه اول بدون هیچ تاخیری CPU را در اختیار خود می گیرد. تابع دیگری که نسبت به بخش قبل اضافه شده است، تابع Swap است. در حین مقایسه زمان های اجرا ، در صورتی که لازم باشد تغییر در صف اجرا رخ دهد ، توسط این تابع انجام می شود. (چون مرتب سازی انجام می دهیم اندیس پردازه ها در لیست برهم می خورد. بنابراین با ذخیره کردن اندیس اولیه (initial_index) این مشکل را برطرف می کنیم و در نمایش بر اساس initial_index نشان می دهیم).

بقیه توابع همان توابع موجود در بخش اول هستند که تنها برای محاسبه زمان های خواسته شده به کار می روند. نمونه خروجی به صورت زیر است :

```
Enter number of processes: 3
Enter burst time of process 1: 2
Enter burst time of process 2: 4
Enter burst time of process 3: 1
Waiting time for each process:
process No.1: 0
process No.3: waiting time = 2
process No.2: waiting time = 3
Turnaround time for each process:
process No.1: turnaround time = 2
process No.3: turnaround time = 3
process No.2: turnaround time = 7
Average waiting time: 1.666667
Average turnaround time: 4.000000


...Program finished with exit code 0
Press ENTER to exit console.
```

بخش سوم : پیاده سازی الگوریتم Priority

این الگوریتم هم خیلی شبیه بخش دوم است. تابع مرتب سازی با الگوریتم مرتب سازی حبابی روی اولویت های پردازه ها اجرا می شود به جای زمان اجرای پردازه ها. بقیه توابع تکراری هستند.

نکته ای که در بخش های قبل ذکر شد در مورد این پیاده سازی هم صدق می کند یعنی پردازه اول صرف نظر از اولویت اول از همه اجرا می شود بنابراین برای راحتی کار در نمونه زیر به آن اولویت ۱ داده شده است تا اجرا منطقی تر به نظر برسد. تغییر در صف اجرا در مورد پردازه دوم و سوم کاملا مشخص است و نشانه عملکرد درست الگوریتم است.

نمونه خروجی :

```
Enter number of processes: 3
Enter burst time of process 1: 1
Enter priority of process 1: 1
Enter burst time of process 2: 3
Enter priority of process 2: 3
Enter burst time of process 3: 2
Enter priority of process 3: 2
Waiting time for each process:
process No.1: 0
process No.3: waiting time = 1
process No.2: waiting time = 3
turnaround time for each process:
process No.1: turnaround time = 1
process No.3: turnaround time = 3
process No.2: turnaround time = 6
Average waiting time: 1.333333
Average turnaround time: 3.333333


...Program finished with exit code 0
Press ENTER to exit console.
```

بخش چهارم: پیاده سازی الگوریتم Round Robin

در این بخش الگوریتم Round-Robin به صورت قبضه شدنی با تایم کوانتومی که از کاربر میگیریم پیاده سازی می کنیم. در هر مرحله با بررسی اینکه زمان باقیمانده هر پردازه چقدر است و مقایسه آن با تایم کوانتوم، به اندازه min(p[i].remaining_time, q) از مقدار باقیمانده پردازه کم میکنیم. سپس پردازه‌ای که اجرا شده را در توالی اجرای پردازه ها چاپ میکنیم و به اندازه ای که از زمان باقیمانده کم شد، به متغیر time passed که زمان سپری شده را نگه‌داری میکند اضافه می کنیم.

بعد از این مرحله، اگر این زمان باقیمانده به صفر برسد، متغیر time passed را capture میکنیم و به عنوان زمان خاتمه(Completion Time) پردازه در نظر میگیریم. دو رابطه اصلی با توجه به تعریف زمان انتظار و زمان اجرا داریم:

Turn-around time = Completion time (time passed) - Arrival time (= 0)

Waiting time = Turn-around time – burst time

نمونه خروجی :

```
Enter number of processes: 4
Enter burst time of process 1: 5
Enter burst time of process 2: 10
Enter burst time of process 3: 13
Enter burst time of process 4: 6
Enter quantum time: 3
process 1
process 2
process 3
process 4
process 1
process 2
process 3
process 4
process 2
process 3
process 2
process 3
process 3
Waiting time for each process:
process No.1: 0
process No.2: waiting time = 20
process No.3: waiting time = 21
process No.4: waiting time = 17
turnaround time for each process:
process No.1: turnaround time = 14
process No.2: turnaround time = 30
process No.3: turnaround time = 34
process No.4: turnaround time = 23
Average waiting time: 16.750000
Average turnaround time: 25.250000


...Program finished with exit code 0
Press ENTER to exit console.
```

بخش پنجم: مقایسه الگوریتم ها

به نوعی هر کدام از این الگوریتم ها براساس ورودی ها و نوع پردازه ها و هدفی که پردازه ها دارند، کاربرد خاص خود را دارند و نمی توان بهترین را برای هر شرایطی انتخاب کرد. پس انتخاب الگوریتم زمان بند وابسته به شرایط موجود است. با توجه به ۴ الگوریتم بالا، ۴ حالتی که استفاده از هرکدام از آنها سودمند خواهد بود، بررسی شده است.

۱. زمانی که پردازه ها زمان سرویس دهی کمی دارند و نیازی به انجام این پردازه ها در ترتیب خاصی نیست، بهترین روش FCFS است زیرا در مقایسه با روشهای دیگر، هر پردازه بسیار کوتاهتر در صف انتظار می‌ماند و Starvationبه کمترین مقدار خود می رسد.
۲. حالت دوم هم وقتی بخواهیم با لیستی دلخواه کمترین زمان انتظار را بدست بیاوریم که در این حالت با توجه به اینکه منطق الگوریتم SJF پردازه های کوتاه را سریعتر انجام میدهد (به طور شهودی و هم قابل اثبات) میتوان گفت که بهترین و کمترین میانگین زمان اجرا را بین سایر الگوریتمها دارد.
۳. استفاده از الگوریتم Priority در زمان هایی کابردی است که اولویت اجرای هر پردازه برای ما مهم باشد. این حالات بیشتر در کار کردن با وقفه ها و تغییر حالت User Mode و Kernel Mode رخ می دهد. این الگوریتم برای برنامه هایی مناسب است که زمان اجرا و دسترسی به منابع آن در طول زمان در نوسان است و ثابت نیست. پس اولویت اجرا مهم است.
۴. زمانی که لیست پردازه ها ترکیبی از burst time های کوتاه و بلند است و برنامه فقط وقتی انجام میشود که تمام پردازه ها در زمان داده شده، اجرا شوند: در این حالت الگوریتم Round-Robin بهترین روش است زیرا Starvation تولید نمیکند و همچنین زمان یکسانی را به هر پردازه میدهد.

پرسش این بخش در اصل برای تعداد زیادی پردازه است. اما نمی توان تنها براساس تعداد نتیجه گیری کرد چون تنها تعداد عنصر مقایسه برای انتخاب نیست. همانطور که در ۴ حالت بالا گفته شده است نوع عملکرد ، زمان اجرا، میانگین زمان اجرا ها و نوع پردازه ها و .... همه عناصر مهمی در انتخاب الگوریتم زمان بند هستند.

اما اگر بخواهیم یک نمونه بزرگ که خیلی با آن کار می کنیم را بررسی کنیم، Windows نمونه خوبی است. در دنیای واقعیه سیستم عامل ها استفاده از این الگوریتم ها به شیوه کتابی و تنها نیست و سازندگان تلاش می کنند تا یک نمونه ترکیبی و بهینه با توجه به هدف سیستم عامل خود ، برای آن بسازند. برای نمونه Windows مانند بسیاری دیگر از سیستم عامل های پیشرفته دیگر، الگوریتم Round Robin را به عنوان الگوریتم مرکزی داراست. این الگوریتم هسته با استفاده از Multi-Level Feedback Queue با استفاده از الگوریتم Priority بهینه سازی می شود تا پردازه هایی با اولویت کم با پردازه های مهم هسته، تداخل نکنند. این نوع پیاده سازی از زمان عرضه نسخه Vista توسط شرکت Microsoft استفاده می شود. البته نسخه های امروزی بهینه سازی های جزئی فرعی دارند.

بخش امتیازی : پیاده سازی الگوریتم SRJF

پیاده سازی این الگوریتم در فایل زیپ موجود است. در این الگوریتم برای اینکه نمایش خروجی منطقی داشته باشیم و عملکرد را بررسی کنیم، باید زمان ورود را هم از کاربر به عنوان ورودی بگیریم. چون این بخش از دستور کار خارج بود ، پردازه اول هم از نوع قبضه شدنی فرض شده و در نمونه زیر درستی الگوریتم مشخص است. کامنت های مناسب برای پیاده سازی قرار داده شده است.

```
Enter Number of the Process:2
Enter the Arrival time of the Process 0:  0
Enter Burst Time of the Process 0:  4
Enter the Arrival time of the Process 1:  1
Enter Burst Time of the Process 1:  2

 PROCESS 1: Finish Time==> 6 Turnaround Time==>6 Watiting Time==>2

 PROCESS 2: Finish Time==> 3 Turnaround Time==>2 Watiting Time==>0

 Watiting Time Average==>       1.000000
 Turnaround Time Average==>     4.000000


...Program finished with exit code 0
Press ENTER to exit console.
```

دو پردازه در زمان های ۰ و ۱ وارد سیستم می شوند. پردازه اول ۴ ثانیه زمان اجرا دارد و در ثانیه ۱ که پردازه دوم با زمان اجرای ۲ ثانیه وارد می شود، پردازه دوم به دلیل زمان کمتر اولویت بالاتری دارد و پردازنده از پردازه اول گرفته می شود و به پردازنده دوم داده می شود.

به همین دلیل زمان انتهایی برای پردازه اول ثانیه ششم است و زمان صبر آن هم به ۲ ثانیه تغییر می کند.

این قبض شدن پردازنده از پردازه و نمایش زمان های درست نشانه عملکرد درست الگوریتم است.