دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)

پروپوزال درس اصول سیستم‌های عامل

پروژه اول

نگارش
علی قاسمی
یاسمین ناجی

استاد راهنما
دکتر قربانعلی

حل‌تمرین‌های مشاور
مهراد اسمعیل‌زاد
امین‌حسین سید قلعه

اردیبهشت ۱۴۰۳

چکیده

این گزارش سه الگوریتم برنامه‌ریزی را برای زمان‌بندی فرایندها، یعنی FCFS، Round Robin و SRTF بررسی می‌کند. این الگوریتم‌ها با هدف توزیع مناسب فرایندها در CPU و بهبود عملکرد سیستم‌های کامپیوتری طراحی شده‌اند. هر الگوریتم مورد بررسی قرار گرفته و نحوه عملکرد آن توضیح داده شده است. برای هر الگوریتم، مراحل پیاده‌سازی و شبه کد مربوطه آورده شده است. در ادامه، با استفاده از مجموعه‌ای از فرایندها، هر یک از الگوریتم‌ها را اجرا و نتایج به دست آمده را مورد بررسی قرار داده است. برای هر الگوریتم، زمان انتظار و زمان برگشت هر فرایند محاسبه شده و میانگین این مقادیر نیز گزارش شده است. همچنین با برنامه‌ی نوشته شده، با استفاده از داده‌های اولیه مرتبط به فرایندها، زمان شروع، ادامه و پایان هر کدام از پروسه‌ها را برنامه‌ی ما چاپ می‌کند. با توجه به نتایج به دست آمده، الگوریتم SRTF بهترین عملکرد را از نظر کاهش زمان انتظار و زمان برگشت دارد. در ادامه، مقایسه‌ای بین عملکرد این سه الگوریتم ارائه شده و اهمیت استفاده از هر الگوریتم برای شرایط مختلف سیستم‌های کامپیوتری بررسی شده است.

واژگان کلیدی: FCFS, Round Robin, SRTF، شبیه‌سازی

فهرست مطالب

صفحه

فهرست اشکال

صفحه

فصل اول

مقدمه

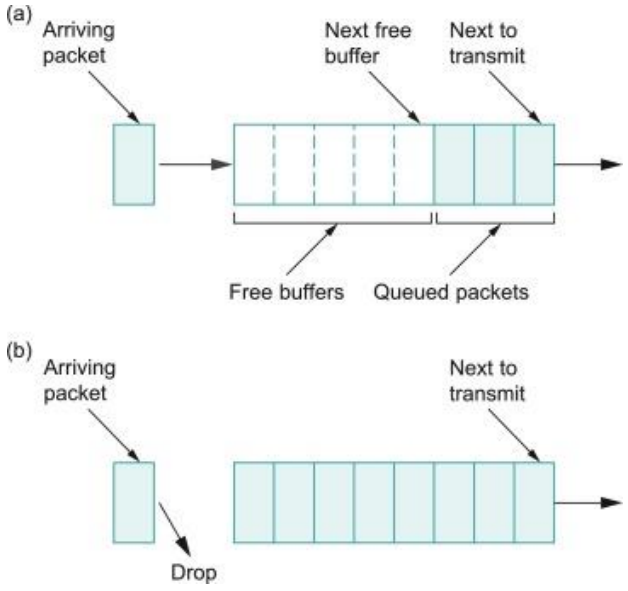
مقدمه

در این پروژه، یک برنامه نوشته خواهد شد که با هدف شبیه‌سازی چندین سیاست زمان‌بندی CPU طراحی شده است. این برنامه قادر خواهد بود الگوریتم‌های مختلف زمان‌بندی را پیاده‌سازی کرده و عملکرد آن‌ها را مورد بررسی قرار دهد. شبیه‌سازی بر اساس الگوریتم‌های زمان‌بندی، وظایف مختلف را از صف اجرا انتخاب می‌کند و برای آن‌ها CPU مناسب را انتخاب می‌کند. در واقع با توجه به هر کدام از این سیاست‌های زمان‌بندی، تسک‌های آماده در صف انتخاب می‌شوند و به CPU داده می‌شوند. هنگامی که یک وظیفه زمان‌بندی می‌شود و در واقع برای اختصاص دادن CPU آن را انتخاب می‌کنیم، شبیه‌ساز به سادگی چاپ می‌کند که کدام وظیفه برای اجرا در یک زمان مشخص انتخاب شده است.

در این پروژه، سه الگوریتم زمان‌بندی مورد استفاده قرار خواهد گرفت:

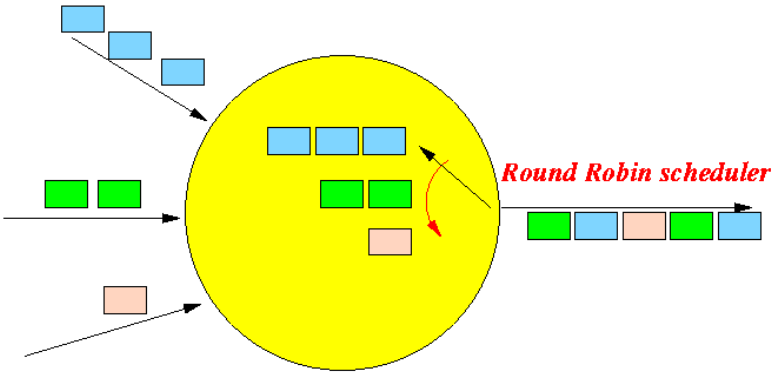
1. First-Come, First-Served (FCFS)
2. Round Robin (RR)
3. Shortest Remaining Time First (SRTF)*

الگوریتم اول(FCFS)، یکی از ساده‌ترین الگوریتم‌های زمان‌بندی CPU است. در این الگوریتم، وظایف بر اساس زمان ورود به صف، به ترتیب ورود به CPU اجرا می‌شوند. به این معنی که وظیفه‌ای که اولین بار وارد صف شده است، اولین وظیفه‌ای است که اجرا می‌شود. این الگوریتم بدین معنی است که وظایف بر اساس ترتیب ورود به صف CPU اجرا می‌شوند، بدون در نظر گرفتن زمان اجرا یا طول زمان اجرا برای هر وظیفه.[۱] الگوریتم FCFS به عنوان یک الگوریتم غیرقابل برگشت(قبضه‌نشدنی) معرفی می‌شود، به این معنی که وظایفی که وقتی وارد صف می‌شوند، تا زمان اتمام اجرا یا اتمام تمام وظایف قبلی، منتظر می‌مانند. این الگوریتم برای الگوریتم‌هایی با زمان بلندتر مفیدتر است تا به نسبت برای الگوریتم‌هایی با زمان کوتاه‌تر. برعکس مکانیزم‌هایی مثل SJF در این الگوریتم گرسنگی وجود ندارد و همچنین اینکه سربار حداقل است چون نیاز به اطلاعات قبلی در مورد فرایندها ندارد و فقط بحث ترتیب در این الگوریتم مورد توجه است[۲]. در عکس ۱، شمایی از این الگوریتم قابل مشاهده است.



عکس ۱(FCFS)

در الگوریتم Round Robin (RR) یا همان نوبت گردشی، هر وظیفه برای یک بازه زمانی ثابت اجـرا می‌شود که به آن به عنوان "کوانتوم زمانی" یا time quantum اطلاق می‌شود. وظایف به ترتیب ورود به صف اجرا می‌شوند. اگر وظیفه در زمان کوانتومی اجرا نشده باشد، از صف خارج شده و در انتهای صف قرار می‌گیرد تا در اجراهای بعدی مورد توجه قرار گیرد. ایـن الگـوریتم از دوره‌هـای کوتـاه زمـانی بـرای اجرای وظایف استفاده می‌کند، که این امر باعث می‌شود هیچ وظیفه‌ای به تنهایی به طور مداوم CPU را در اختیار نداشته باشد و از CPU به اندازه کافی بهره‌مند شود. [۳]در واقع این الگوریتم شـبیه FCFS است اما با این تفاوت که زمان‌بند پردازه بین فرایندها در یک صف چرخشی حرکت می‌کند. همان طـور که مشخص است این الگوریتم قبضه‌شدنی است. همچنین گرسنگی در آن وجود ندارد چون با توجه به آن کوانتوم زمانی، همه‌ی پروسه‌ها شانس این را پیدا می‌کنند تا از CPU بهره ببرند و ایـن الگـوریتم در سیستم‌های اشتراک زمانی بسیار موثر است[۴]. همچنین اینکه برای تعیین کوانتـوم زمـانی بایـد دقـت شود که اگر خیلی کوچک باشد توان عملیاتی پایین می‌آید و همچنین اینکه اگر ایـن کوانتـوم زمـانی از طولانی‌ترین تسک، بیشتر باشد این الگوریتم عینا مثل FCFS عمل می‌کند. در عکس ۲، شمایی از این الگوریتم نشان داده شده. به حالتی که هر پروسه(مستطیل‌های هم‌رنگ) با توجه به همان کوانتوم زمـانی بخش بخش شده‌اند تا به CPU داده شوند.



عکس ۲(Round Robin)

و امـا در الگـوریتم سـوم، الگـوریتم Shortest Remaining Time First (SRTF) وظیفـه‌ای کـه کمترین زمان باقی‌مانده اجرا را دارد، انتخاب و اجرا می‌شود. بدین ترتیب، همواره وظیفه با کمترین زمان باقی‌مانده اجرا، اولویت اجرا را دارد. اگر وظایفی با زمان اجرای یکسان و باقی‌مانده متفاوت وجود داشـته باشند، وظیفه‌ای که کوچک‌ترین زمان اجرا را دارد انتخاب می‌شود.[۵] الگوریتم SRTF به عنـوان یـک نسخه بهبود یافته از الگوریتم Shortest Job First (SJF) مطرح می‌شود که در آن، زمان باقی‌مانـده اجرا به جای زمان اجرا در نظر گرفته می‌شود. این الگوریتم به کمک محاسبه دقیـق‌تر زمـان باقی‌مانـده اجرا برای هر وظیفه، بهبود قابل توجهی در کارایی زمان‌بندی CPU ایجاد می‌کند. این الگوریتم در واقع یک نوع الگوریتم SJF قبضه‌شدنی است. امکان گرسنگی برای پروسه‌هایی با زمان اجرای طولانی وجـود دارد. همچنین اینکه turnaround time برای این الگوریتم از SJF بهتر است.[۶]

در این برنامه، ورودی‌ها به این حالت دریافت می‌شوند:

Pid،arrival_time،burst_time

pidیک شناسه فرآیند است که عددی منحصـر بـه فـرد اسـت. در آن arrival_time زمـان ورود آن وظیفه یا همان تسکی است که باید به CPU اختصاص داده شود و burst_time نیز زمان درخواسـتی آن تسک است که از CPU می‌خواهد.

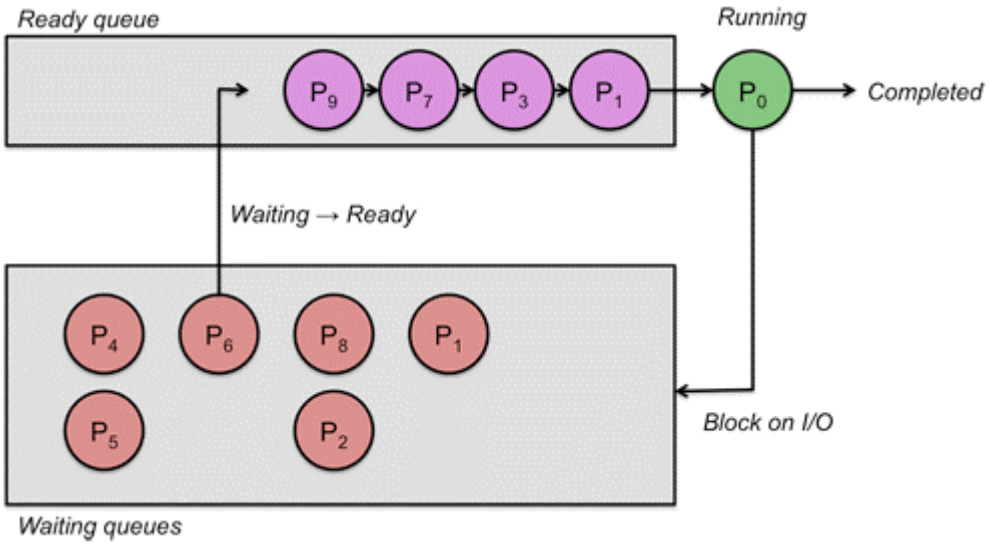
فصل دوم

روش پیشنهادی

روش پیشنهادی

FCFS:

در این بخش، ما روش پیاده‌سازی الگوریتم اولویت دهی به ترتیب ورودی (FCFS)را بررسی و توضیح خواهیم داد. این الگوریتم اولین فرآیندی را که به سیستم عامل داده می‌شود، بر روی CPU اجرا می‌کند و تا زمانی که کاملاً اجرا شود، دیگر هیچ فرآیندی را اجرا نمی‌کند[۷]. البته در صورتی که یک وقفه برای I/Oایجاد شود، فرآیند از صف خارج شده و بعد از اتمام وقفه به ته صف فرستاده می شود(البته در این پیاده سازی ما وقفه های I/O و cpu را در نظر نمی گیریم و فرآیند ها بدون وقفه انجام می شوند)



عکس ۳(FCFS)

- ساختار داده : Queueبرای پیاده‌سازی FCFS، ابتدا باید یک داده‌ساختار Queue ایجاد کنیم که فرآیندها را درون آن ذخیره کنیم. این داده‌ساختار باید عملیات اضافه کردن (enqueue)و حذف کردن (dequeue)را پشتیبانی کند.
- تعریف فرآیند : هر فرآیند باید دارای ویژگی‌هایی مانند شناسه)ID(، زمان ورود به سیستم (arrival time)و زمان اجرا (burst time)باشد. برای هر فرآیند یک ساختار داده باید تعریف شود.

- الگوریتم اجرا : برای اجرای الگوریتم FCFS، ما از داده‌ساختار Queue استفاده می‌کنیم. زمانی که یک فرآیند وارد سیستم می‌شود، به انتهای صف اضافه می‌شود. زمانی که فرآیند از CPU خارج می‌شود، فرآیند بعدی آن در صف وارد CPU می شود و اجرای آن آغاز می‌شود.

- بخش اجرای الگوریتم : در این بخش یک حلقه اجرا می‌شود که تمامی فرآیندها را به ترتیب ورودی درون صف قرار داده و اجرای هر فرآیند را شبیه‌سازی می‌کند.

- محاسبه زمان اجرا: برای شبیه‌سازی و محاسبه زمان اجرا هر فرآیند، از زمان ورود به سیستم و زمان اجرا استفاده می‌کنیم. با توجه به این دو مقدار، می‌توانیم زمان انتظار و زمان اتمام هر فرآیند را محاسبه کنیم. محاسبه زمان انتظار هر فرآیند با توجه به زمان انتظار و زمان انفجار فرآیند قبلی انجام می شود که در شبه کد زیر آمده است.

- گزارش نتایج: پس از اجرای الگوریتم FCFS بر روی مجموعه داده‌های آزمایشی، نتایج به دست آمده شامل زمان انتظار، زمان اتمام و میانگین این مقادیر خواهد بود. در بخش نتایج بیشتر به این قسمت می پردازیم.

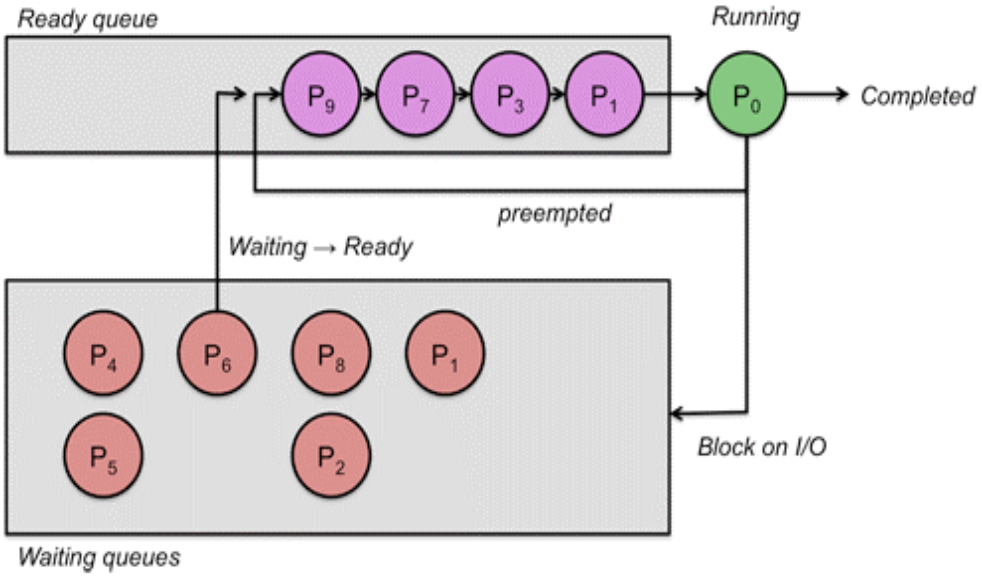
شبه کد الگوریتم : FCFS

```
1-  Input the processes along with their burst time (bt).
2-  Find waiting time (wt) for all processes.
3-  As first process that comes need not to wait so
    waiting time for process 1 will be 0 i.e. wt[0] = 0.
4-  Find waiting time for all other processes i.e. for
     process i ->
       wt[i] = bt[i-1] + wt[i-1] .
5-  Find turnaround time = waiting_time + burst_time
    for all processes.
6-  Find average waiting time =
                 total_waiting_time / no_of_processes.
7-  Similarly, find average turnaround time =
                 total_turn_around_time / no_of_processes.
```

Round Robin (with static quantum time):

در این بخش، ما روش پیاده‌سازی الگوریتم چرخه‌ای (Round Robin) را بررسی و توضیح خواهیم داد. این الگوریتم یک روش اولویت‌بندی فرآیندها برای اجرای قسمت‌های کوچکی از زمان استفاده می‌کند و سپس فرآیندها را به ترتیب دوره‌ها (time slice) اجرا می‌کند.[۸]

الگوریتم کوانتومی زمان ایستا به این معنی است که کوانتوم زمان دارای یک مقدار ثابت خواهد بود که در طول اجرای الگوریتم تغییر نمی کند و مقدار از قبل تعیین شده است. این الگوریتم از نظر پیاده سازی ساده تر از الگوریتم کوانتومی زمانی پویا است، اما عملکرد آن بستگی به کوانتوم زمانی تعیین شده دارد. معمولاً با مقدار کوانتومی زمان بالا، این الگوریتم مانند الگوریتم FCFS عمل می کند.



عکس ۴(RR)

مراحل پیاده‌سازی

- تعیین زمان دوره (Quantum Time): برای پیاده‌سازی الگوریتم Round Robin، ابتدا باید زمان دوره را تعیین کنیم. این زمان مشخص می‌کند که هر فرآیند چه مدت زمانی را در CPU اجرا می‌شود. این مقدار باید به طور متوسط و قابل تنظیم باشد.
- ساختار داده Queue: مانند FCFS، برای پیاده‌سازی Round Robin نیاز به یک Queue داریم. این Queue باید فرآیندها را به ترتیب ورود به صف قرار دهد و قابلیت عملیات اضافه کردن و حذف کردن فرآیندها را داشته باشد.

- الگوریتم اجرا: الگوریتم اجرای Robin Round به این صورت است که هر فرآیند به مدت زمان دوره در CPU اجرا می‌شود و سپس به انتهای صف اضافه می‌شود. اگر فرآیند هنوز ادامه دارد، به صف برمی‌گردد و سایر فرآیندها اجرا می‌شوند.

- بخش اجرای الگوریتم: در اینجا، یک حلقه اجرا می‌شود که فرآیندها را به ترتیب ورود به صف قرار داده و آن‌ها را به مدت زمان دوره در CPU اجرا می‌کند. در صورتی که فرآیند هنوز ادامه داشته باشد، به انتهای صف اضافه می‌شود.

- بررسی و انتقال به فرآیند بعدی: بعد از گذشت زمان دوره، فرآیند فعلی از CPU خارج می‌شود و به انتهای صف اضافه می‌شود. سپس فرآیند بعدی در صف به CPU ارسال می‌شود و فرآیند جدید شروع به اجرا می‌کند. همچنین اگر فرآیندی قبل از رسیدن به time quantum به اتمام برسد، cpu دیگر منتظر اتمام time quantum نمی ماند و بلافاصله فرآینده بعدی را از صف خارج کرده و پردازش را شروع می کند

شبه کد برای اجرای مرحله به مرحله الگوریتم round robin [۹]

```
queue = empty Queue // فرآیندها نگهداری برای خالی صف یک ایجاد

current_time = 0 //فعلی زمان اولیه مقداردهی

while processes not empty or queue not empty: // صف یا ورودی فرآیندهای که زمانی تا
    باشند نشده خالی

    while processes not empty and processes[0].arrival_time <=
    current_time:

    صف به شده وارد فرآیندهای کردن اضافه //

    if queue is not empty: // نباشد خالی صف اگر

        اجرا برای فرآیند انتخاب //

        اجرا زمان محاسبه //

        فعلی زمان به اجرا زمان افزودن //

        فرآیند اجرای باقی‌مانده زمان کاهش //
```

if current_process.remaining_time > 0: // اگر اجرای فرآیند تمام نشده باشد

// فرآیند به صف اضافه می‌شود

else:

// فرآیند برای اجرا پایان زمان اطلاعات ثبت می‌شود

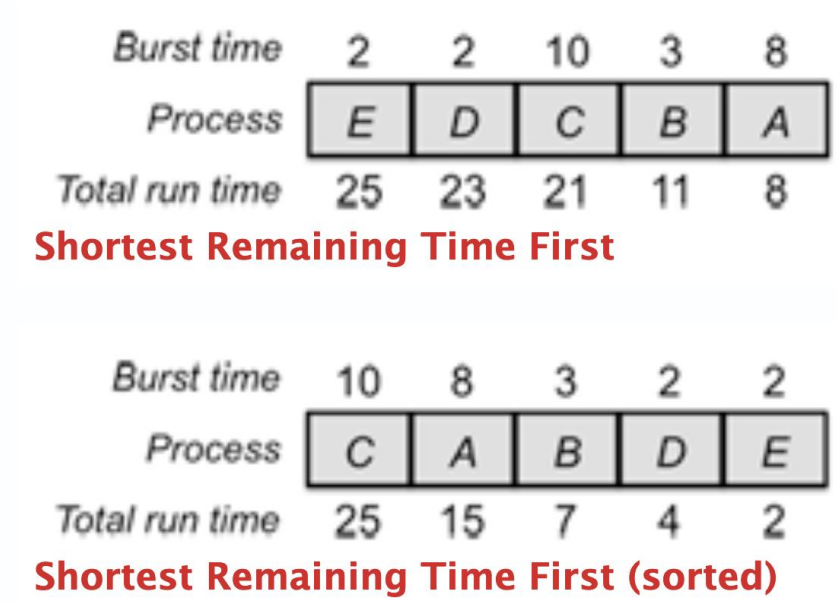
else:

// فرآیند به فعلی زمان ورود زمان تنظیم بعدی می‌شود

return processes // بازگرداندن فرآیندها لیست با اطلاعات پایانی

:Shortest Remaining Time First

در این بخش، ما روش پیاده‌سازی الگوریتم کوتاه ترین باقی‌مانده زمان اولویت (SRTF)را بررسی و توضیح خواهیم داد. این الگوریتم به صورت انتخاب فرآیند با کوتاه‌ترین زمان باقی‌مانده کار می کند.



عکس ۵(SRTF)

مراحل پیاده‌سازی

- الگوریتم اجرا: الگوریتم SRTF به این صورت است که همیشه فرآیندی که کوتاه‌ترین زمان باقی‌مانده را دارد را برای اجرا انتخاب می‌کند. در هر زمان، فرآیندی که کوتاه‌ترین زمان باقی‌مانده را دارد، انتخاب و برای اجرا ارسال می‌شود. اگر یک فرآیند با زمان کوتاه‌تر وارد سیستم شود، فرآیند جاری متوقف شده و به صف اضافه می‌شود. پس از رسیدن همه فرآیندها، دیگر وقفه ای در حین اجرای فرآیند برای جایگزینی فرآیند ها ایجاد نمی شود و الگوریتم به صورت SJF کار می‌کند.

- محاسبه زمان اجرا و گزارش نتایج: برای هر فرآیند، می‌توانیم با استفاده از زمان ورود به سیستم و زمان اجرا، زمان انتظار و زمان اتمام را محاسبه کنیم. پس از اجرای الگوریتم بر روی مجموعه داده‌های آزمایشی، نتایج شامل زمان انتظار، زمان اتمام و میانگین این مقادیر خواهد بود.[۱۰]

پیاده سازی مرحله به مرحله این الگوریتم به صورت زیر خواهد بود :

- Traverse until all process gets completely executed.
    - Find process with minimum remaining time at every single time lap.
    - Reduce its time by 1.
    - Check if its remaining time becomes 0
    - Increment the counter of process completion.
    - Completion time of current process = current_time + 1;
    - Calculate waiting time for each completed process.
        - **wt[i]= Completion time – arrival_time-burst_time**
    - Increment time lap by one.
- Find turnaround time (waiting_time + burst_time).

فصل سوم

نتایج

نتایج

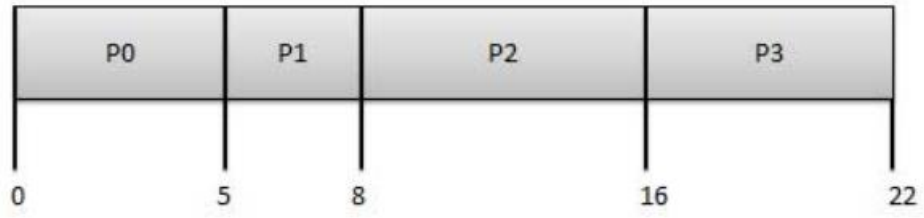
یک زمان‌بند فرایندها، برای فرایندهای مختلف برنامه‌ریزی می‌کند تا CPU را در دسترس آن‌ها قرار دهد. حال یکی از گزارش‌های اخیر را در مورد این موضوع مورد بررسی قرار می‌دهیم. الگوریتم‌های مورد بررسی در این گزارش[۱۱]، FCFS، Round Robin و SRTF هستند. این الگوریتم‌ها یک سری قبضه‌شدنی و یک سر قبضه‌نشدنی هستند.

اگر فرایندهای زیر را داشته باشیم:

Process	Arrival Time	Execute Time
P0	0	5
P1	1	3
P2	2	8
P3	3	6

عکس ۶(processes)

الگوریتم FCFS برای آن‌ها به این حالت عمل می‌کند.



عکس ۷(FCFS on processes)

همانطور که مشخص است، به ترتیب ورود هر کدام از پروسه‌ها، به آن‌ها CPU را اختصاص می‌دهد.

زمان برگشت هر کدام از پروسه‌ها:

P0 = 5 – 0

P1 = 8 – 1

P2 = 16 – 2

P3 = 22 – 3

که به صورت میانگین خواهیم داشت.

$(5 + 7+ 14 + 19)/4 = 11.25$

برای زمان انتظار خواهیم داشت:

$P0 = 5 – 5 – 0$

$P1 = 8 – 3 – 1$

$P2 = 16 – 2 – 8$

$P3 = 22 – 6 – 3$

که به صورت میانگین خواهیم داشت:

$(0 + 4 + 6 + 13)/4 = 5.75$

همانطور که در این الگوریتم مشخص است، هیچ کدام از تسک‌ها paused نشده انـد. در واقـع وقتـی کـه شروع شده‌اند تا زمان اتمام تسک، CPU را در اختیار داشته‌اند.

خواهیم داشت:

Time 0 : task 0 starts

Time 5: task 0 finished

Time 5: task 1 starts

Time 8: task 1 finished

Time 8: task 2 starts

Time 16: task 2 finished

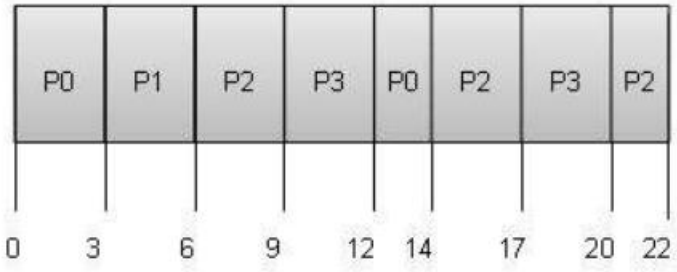
Time 16: task 3 starts

Time 22: task 3 finished

Average waiting time : 5.75

Average turnaround time: 11.25

حال برای همان پروسه‌ها اگر Round Robinرا بخواهیم اجرا کنیم(با کوانتوم زمانی ۳)، خواهیم داشت:



عکس ۸(RR on processes)

زمان برگشت هر کدام از پروسه‌ها:

P0 = 14 – 0

P1 = 6 – 1

P2 = 22 – 2

P3 = 20 – 3

که به صورت میانگین خواهیم داشت:

(14 + 5 + 20 + 17)/4 = 14

زمان انتظار:

P0 = 14 - 5 – 0

P1 = 6 – 1 – 3

P2 = 22 – 2 – 8

P3 = 20 – 3 – 6

که به صورت میانگین خواهیم داشت:

(9 + 2 + 12 + 11)/4 = 8.5

در این الگوریتم برعکس الگوریتم قبل، pause شدن تسک‌ها اتفاق می‌افتد.

Time 0 : task 0 starts

Time 3 : task 0 paused, task 1 starts

Time 6: task 1 finished, task 2 starts

Time 9: task 2 paused, task 3 starts

Time 12: task 3 paused, task 0 resumes

Time 14: task 0 finished, task 2 resumes

Time 17: task 2 paused, task 3 resumes

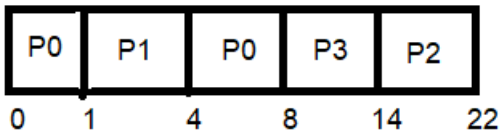
Time 20: task 3 finished, task 2 resumes

Time 22: task 2 finishes, all tasks finished

Average waiting time : 8.5

Average turnaround time: 14

حال برای SRTF خواهیم داشت:



عکس ۹(SRTF on processes)

زمان برگشت:

P0 = 8 – 0

P1 = 4 – 1

P2 = 22 – 2

P3 = 14 – 3

میانگین:

$(8 + 3 + 20 + 11)/4 = 10.5$

زمان انتظار:

$P0 = 8 – 0 – 5$

$P1 = 4 – 1 – 3$

$P2 = 22 – 8 – 2$

$P3 = 14 – 3 – 6$

میانگین:

$(3 + 0 + 12 + 5)/4 = 5$

خواهیم داشت:

Time 0 : task 0 starts

Time 1: task 0 paused, task 1 starts

Time 4: task 1 finished, task 0 resumes

Time 8: task 0 finished, task 3 starts

Time 14: task 3 finished, task 2 starts

Time 22: task 2 finished, all tasks finished

Average waiting time : 5

Average turnaround time: 10.5

با توجه به نتایج به دست آمده از این سه الگوریتم می‌توانیم متوجه شویم که در موضوع زمان انتظار، میانگین روش SRTF از دو روش دیگر کمتر بوده و RR از دو روش دیگر بیشتر است. همچنین در مورد زمان بازگشت هم به همین ترتیب است.با مقایسه نتایج برای این پروسه‌ها متوجه می‌شویم که برای این مجموعه از فرایندها، SRTF بهتر عمل کرده است و هم در حیطه‌ی زمان انتظار و هم زمان برگشت، به صورت میانگین، زمان کمتری را نسبت به بقیه الگوریتم‌ها داشته و RR هم به نسبت دو الگوریتم دیگر، از لحاظ زمانی بدتر عمل کرده. اما خب باید این را در نظر داشته باشیم که در سیستم‌هایی که از RR استفاده می‌شود، احتمال رخداد گرسنگی برای فرایندها وجود ندارد، و اگر این داستان برای ما حائز اهمیت باشد، قطعا RR را در اولویت قرار می‌دهیم. همچنین مطابق داکیومنتیشن پروژه، در صورت پیاده‌سازی کد مورد نظر(که در بخش روش پیشنهادی توضیح داده شده است)، خروجی برنامه را برای این مجموعه از پروسه‌ها نیز نمایش دادیم. در الگوریتم‌های قبضه شدنی، برای مثال در Round Robin، در صورتی که کوانتوم زمانی هر پروسه تمام شود، آن پروسه پاز می‌شود و CPU در اختیار پروسه‌ی بعدی که در صف قرار دارد می‌رسد. در صورتی که کوانتوم زمانی برای اتمام پروسه کافی باشد، آن تسک به اتمام می‌رسد و در صورتی که خیر، در انتهای صف قرار می‌گیرد تا دوباره resume شود و از کوانتوم زمانی بعدی که به آن اختصاص داده می‌شود بتواند استفاده کند. همچنین برای SRTF در زمان صفر چون تنها پروسه‌ی صفر حاضر بود، در دست CPU قرار گرفت و در زمان یک، چون پروسه‌ی یک تایم کمتری برای اتمام پروسه‌ی خود نسبت به پروسه‌ی صفر می‌خواست، پروسه‌ی صفر پاز شد و CPU در دست پروسه‌ی یک قرار گرفت و بعد از اتمام پروسه‌ی یک، چون تایم باقی‌مانده برای اتمام پروسه‌ی صفر از دو پروسه‌ی باقی‌مانده کمتر بود، CPU باز در دست پروسه‌ی صفر قرار گرفت و بعد از امام آن، پروسه‌های دو و سه به ترتیب اینکه کدامشان کمترین زمان را برای اجرا می‌خواهند، در دست CPU قرار گرفتند. برای الگوریتم FCFS هم که نیز، با توجه به ترتیب ورود و بدون امکان قبضه شدن، پروسه‌ها CPU را اختیار کردند و بعد از اتمام هر پروسه، پروسه‌ی بعدی در دست CPU قرار گرفت.

فصل چهارم

جمع‌بندی

جمع‌بندی

با توجه به مقایسه‌ی عملکرد سه الگوریتم برنامه‌ریزی FCFS، Round Robin و SRTF برای زمان‌بندی فرایندها، مشخص شد که هرکدام از این الگوریتم‌ها ویژگی‌ها و مزایا و معایب خاص خود را دارند. الگوریتم FCFS به عنوان یک الگوریتم ساده و آسان برای پیاده‌سازی شناخته می‌شود. در این الگوریتم، فرایندها به ترتیب ورود به صف CPU، بر روی پردازنده قرار می‌گیرند. این الگوریتم به‌طور متوسط زمان برگشت و زمان انتظار کمی دارد. اما این الگوریتم ممکن است باعث ایجاد پدیده گرسنگی برای فرایندها شود. الگوریتم Round Robin یک الگوریتم کارا برای جلوگیری از گرسنگی فرایندها است. در این الگوریتم، هر فرایند تنها به مدت کوتاهی روی CPU اجرا می‌شود و پس از آن به صف باز می‌گردد. این الگوریتم باعث می‌شود که همه‌ی فرایندها به طور منصفانه از منابع سیستم استفاده کنند. اما الگوریتم Round Robin ممکن است زمان برگشت و انتظار طولانی‌تری را نسبت به الگوریتم‌های دیگر داشته باشد. با توجه به مقایسه‌ی عملکرد سه الگوریتم برنامه‌ریزی FCFS، Round Robin و SRTF برای زمان‌بندی فرایندها، مشخص شد که هرکدام از این الگوریتم‌ها ویژگی‌ها و مزایا و معایب خاص خود را دارند. الگوریتم SRTF به عنوان یکی از بهترین الگوریتم‌های برنامه‌ریزی فرایندها شناخته می‌شود. در این الگوریتم، فرایندهایی که زمان کمتری برای اتمام دارند، اولویت بیشتری در استفاده از CPU دارند. این باعث می‌شود که زمان برگشت و انتظار فرایندها به طور قابل توجهی کاهش یابد. اما الگوریتم SRTF نسبت به الگوریتم‌های دیگر پیاده‌سازی پیچیده‌تری دارد و نیازمند محاسبات دقیق‌تری است. در کل نیز با توجه به بخش نتایج متوجه می‌شویم ک SRTF بهتر عمل می‌کند از لحاظ میانگین زمان برگشت و زمان انتظار. اما به دلیل رخداد گرسنگی ممکن است این الگوریتم برای همه‌ی موقعیت‌ها مطلوب نباشد.

.

فصل پنجم

منابع و مآخذ

منابع و مراجع

[1] Hassin, Refael. "NOTES AND COMMENTS ON THE OPTIMALITY OF FIRST COME LAST SERVED QUEUES." *Econometrica* 53.1 (1985).

[2] Schwiegelshohn, Uwe, and Ramin Yahyapour. "Analysis of first-come-first-serve parallel job scheduling." *SODA*. Vol. 98. 1998.

[3] Rasmussen, Rasmus V., and Michael A. Trick. "Round robin scheduling–a survey." *European Journal of Operational Research* 188.3 (2008): 617-636.

[4] Singh, Ajit, Priyanka Goyal, and Sahil Batra. "An optimized round robin scheduling algorithm for CPU scheduling." *International Journal on Computer Science and Engineering* 2.07 (2010): 2383-2385.

[5] Gao, Chengxi, Victor CS Lee, and Keqin Li. "D-SRTF: Distributed shortest remaining time first scheduling for data center networks." *IEEE Transactions on Cloud Computing* 9.2 (2018): 562-575.

[6] Aboalama, Mohammed, and Adil Yousif. "Enhanced job scheduling algorithm for cloud computing using shortest remaining job first." *International Journal of Computer Science & Management Studies* 15.6 (2015): 65-68.

[7] https://www.geeksforgeeks.org/program-for-fcfs-cpu-scheduling-set-1/

[8] https://www.geeksforgeeks.org/round-robin-scheduling-with-different-arrival-times/

[9] https://www.studocu.com/in/document/zeal-college-of-engineering-and-research/computer-engineering/round-robin-cpu-scheduling-algorithm/38985799

[10] https://people.cs.rutgers.edu/~pxk/416/notes/07-scheduling.html

[11] https://techzzers.wordpress.com/process-scheduling-algorithms-fcfssjfpriority-round-robin