به نام خدا

دانشکده مهندسی کامپیوتر دانشگاه اصفهان

«شبیه‌ساز سیستم عامل»

عنوان درس: ساختمان داده‌ها

استاد درس: دکتر رضا رمضانی

طراحان:

زهرا مهدوی — علی طالبی

دریا ظهیری — پویان جای درنام

مهدی رضایی — امیررضا محمدی یگانه

زمستان ۱۴۰۴

فهرست مطالب

۱) هدف پروژه

هدف این پروژه طراحی و پیاده‌سازی یک **شبیه‌ساز سیستم‌عامل تک‌پردازنده** مبتنی بر زمان‌بندی کوانتومی است که به‌صورت **چرخه‌ای و مبتنی بر ورودی دسته‌ای (Batch-Oriented Input)** عمل می‌کند. در این سیستم، پردازش در قالب بازه‌های زمانی گسسته انجام می‌شود. در ابتدای هر بازه:

- سیستم‌عامل تعداد مشخصی دستور از ورودی دریافت می‌کند.
- سپس به مدت مشخصی (بر حسب تعداد کوانتوم) به اجرای فرایندها می‌پردازد.

این چرخه تا زمان دریافت دستور **SHUTDOWN** ادامه خواهد داشت.

۲) مدل چرخه اجرای سیستم

سیستم‌عامل در قالب چرخه‌های متوالی اجرا می‌شود. هر چرخه شامل دو فاز اصلی است:

۱. فاز دریافت ورودی (Input Phase)

در فاز دریافت ورودی، سیستم‌عامل حداکثر y دستور از ورودی استاندارد دریافت و ثبت می‌کند (اندازه y بر اساس پیکربندی سیستم‌عامل تعیین می‌شود).

۲. فاز اجرای سیستم (Execution Phase)

در فاز اجرای سیستم، سیستم‌عامل به مدت x کوانتوم زمانی عملیات زمان‌بندی و اجرای فرایندها را انجام می‌دهد (اندازه x بر اساس پیکربندی سیستم‌عامل تعیین می‌شود). پس از پایان فاز اجرا، سیستم وارد چرخه بعدی شده و این روند تا دریافت دستور SHUTDOWN ادامه می‌یابد.

۳) تعریف دقیق مسئله

- سیستم تک‌پردازنده‌ای (Single CPU) است و زمان آن، در گام‌های کوانتومی جلو می‌رود.
- در هر کوانتوم فقط یک فرایند می‌تواند RUNNING باشد.
- فقط فرایندهای READY می‌توانند وارد ReadyPriorityQueue شوند.
- فرایندهایی که به دلیل نبود فایل یا پر بودن ReadyPriorityQueue قابل اجرا نیستند، وارد WaitingQueue می‌شوند.

۳.۱) فایل پیکربندی سیستم‌عامل

سیستم‌عامل در هنگام راه‌اندازی از یک فایل پیکربندی با فرمت JSON استفاده می‌کند. این فایل شامل پارامترهای ثابت سیستم است که در طول اجرا تغییر نمی‌کنند. در این سیستم WaitingQueue ظرفیت نامحدود داشته و در فایل پیکربندی با مقدار ۱- برای پارامتر waitingQueueLimit مشخص می‌شود.

۳.۲) پارامترهای فایل پیکربندی (نمونه)

```
{
 "quantumSize": 20,
 "executionQuantumsPerCycle": 5,
 "inputCommandLimit": 3,
 "READY_LIMIT": 5,
 "waitingQueueLimit": -1,
}
```

۳.۳) قوانین استفاده از فایل پیکربندی

- عدم وجود فایل پیکربندی یا نامعتبر بودن آن باعث توقف سیستم با خطای ConfigurationException می‌شود.
- تمامی پارامترها قبل از شروع چرخه اجرا بارگذاری می‌شوند.

۴) قابلیت‌های سیستم

سیستم موظف است قابلیت‌های زیر را پشتیبانی کند:

- ایجاد فرایند
- نگه‌داری همه فرایندها در Process Tree تا زمان اتمام
- نگه‌داری فرایندهای آماده اجرا در ReadyPriorityQueue ظرفیت‌دار
- نگه‌داری فرایندهای غیرقابل اجرا درWaitingQueue
- اجرای دقیقاً یک فرایند در هر کوانتوم زمانی
- اعمال Timeout و تغییر اولویت طبق قانون مشخص
- جلوگیری از اجرای فرایندهایی که فایل موردنیاز آن‌ها موجود نیست
- ثبت startTime و finishTime برای هر PID

۵) ساختمان داده‌ها

- PCB (Process Control Block)

هر فرایند شامل فیلدهای زیر است:

- pid : int
- name : string
- priority : int
- burstTime : int

- remainingTime : int
- state : { READY , RUNNING , WAITING }

- needsFile : bool
- filePath : string (if needsFile==false → "")

- waitReason : { FILE , READY_LIMIT } (زمانی که state=WAITING باشد)

- parent : Process*
- children : list<Process*>

- startTime : int
  - اولین زمانی که پردازنده گرفته
  - Default value = -1 (به معنای آنکه تاکنون این فرایند زمانی از پردازنده نگرفته است)
- finishTime : int (زمان اتمام یا حذف)

- Process Tree (General Tree)

- تمام فرایندها تا قبل از اتمام، در این درخت نگه‌داری می‌شوند.
- هر نود می‌تواند چند فرزند داشته باشد.
- قانون اتمام پدر: اگر remainingTime فرایند پدر به صفر برسد یا با رویداد حذف شود:
  - کل زیردرخت آن فوراً حذف می‌شود
  - finishTime تمام آن‌ها برابر زمان اتمام پدر ثبت می‌شود.
  - اگر فرایندی در READY یا WAITING باشد، از صف‌ها حذف می‌شود.
  - اگر فرایندی در RUNNING باشد:
    - CPU آزاد می‌شود.
    - remainingTime آن تغییری نمی‌کند.

- ReadyPriorityQueue

- Max-Heap بر اساس priority
  - در صورت برابری اولویت دو فرایند (Tie-Break)، ترتیب FIFO رعایت شود.
- ظرفیت محدود (READY_LIMIT)
- توابع:
  - insert(p)
  - extract_max()
  - peek_max()
  - remove(pid)

WaitingQueue -

- شامل تمام فرایندهایی با state=WAITING
- ترتیب FIFO
- فرایندها بر اساس waitReason در این صف قرار می‌گیرند

FileTree -

این ساختار یک درخت عمومی است که هر نود آن می‌تواند نمایانگر یک فایل یا یک دایرکتوری باشد. در این پروژه، سیستم از یک درخت فایل ساده برای مدل‌سازی وجود یا عدم وجود فایل‌ها استفاده می‌کند و فقط جهت بررسی در دسترس بودن فایل موردنیاز فرایندها به کار می‌رود. هدف آن شبیه‌سازی کامل سیستم فایل نیست.

هر نود شامل فیلدها و عملیات زیر است:

- name : نام فایل یا دایرکتوری
- isFile : مشخص می‌کند نود فایل است یا دایرکتوری
- children : لیست فرزندان نود
- exists(path)

بررسی می‌کند آیا فایل با مسیر مشخص‌شده در درخت وجود دارد یا خیر.

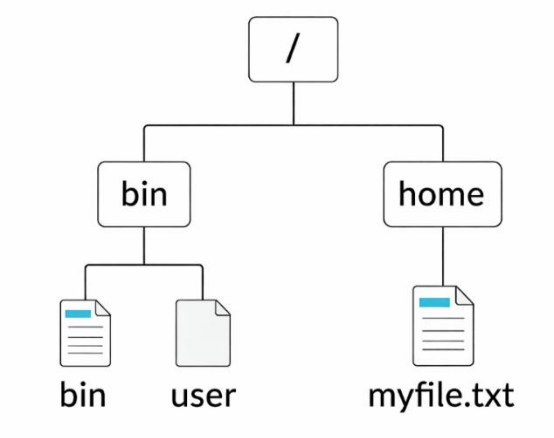
  - **نکته:** ریشه‌ی درخت نمایانگر مسیر / می‌باشد.

- create(path)

فایل مشخص‌شده را در مسیر داده شده ایجاد می‌کند. فرض می‌شود دایرکتوری‌های مسیر از قبل وجود دارند.

- delete(path)

فایل مشخص‌شده از مسیر داده شده حذف می‌کند. در صورتی که چنین مسیر یا فایلی وجود نداشته باشد، خطای مناسب پرتاب می‌شود.



شکل ۱- نمونه‌ای از یک درخت فایل ساده

۶) **قوانین** وابستگی فرایند به فایل

در ابتدای انتخاب فرایند برای اجرا در هر کوانتوم:

- اگر فرایند نیازی به فایل نداشته باشد، همواره از نظر دسترسی به فایل مورد نیاز، قابل اجرا محسوب می‌شود.
- اگر فرایند به فایل نیاز داشته باشد:
  - در صورت وجود فایل در File Tree، قابل اجرا است.
  - در صورت عدم وجود فایل، فرایند وارد حالت WAITING با دلیل FILE می‌شود.

۶.۱) وابستگی فرایند به فایل

هر فرایند می‌تواند حداکثر به یک فایل وابسته باشد که متناسب با آن، در ساختار PCB فیلدهای زیر اضافه می‌شوند:

- needsFile : bool
- filePath : string (در صورت عدم نیاز به فایل، مقدار آن خالی است)

۶.۲) بررسی وضعیت اجرای فرایند به دلیل نبود فایل

در ابتدای هر کوانتوم، اگر فرایند انتخابی برای اجرا به فایلی وابسته باشد که دیگر در FileTree وجود ندارد، فرایند:

- از حالت READY خارج می‌شود
- وارد حالت WAITING برای دسترسی به FILE می‌شود
- پردازنده آزاد شده و فرایند بعدی انتخاب می‌شود

بررسی وجود فایل فقط در ابتدای کوانتوم انجام می‌شود و در میانه‌ی اجرای کوانتوم وقفه‌ای ایجاد نمی‌شود.

۶.۳) به‌روزرسانی وضعیت فرایند پس از ایجاد فایل مورد نیاز آن

پس از اجرای هر دستور مربوط به ساخت یک فایل، سیستم‌عامل صف WaitingQueue را برای یافتن فرایندهایی که نیازمند فایل ساخته شده جدید برای اجرا هستند، بررسی می‌کند. برای چنین فرایندهایی، مقدار متغیر filePath براساس آدرس فایل جدید تعیین می‌شود. این فرایندها ممکن است به یکی از حالت‌های زیر به‌روز شوند:

- ReadyPriorityQueue ظرفیت دارد: فرایند مربوطه، از WaitingQueue خارج و به ReadyPriorityQueue اضافه می‌شود.
- ReadyPriorityQueue ظرفیت ندارد: فرایند مربوطه، از WaitingQueue خارج نمی‌شود اما waitReason آن به READY_LIMIT تغییر می‌یابد.

۶.۴) وضعیت فرایند پس از حذف فایل مورد نیاز آن

در صورت اجرای دستور FILE_DELETE، فایل مشخص‌شده از FileTree حذف می‌شود. بررسی حذف فایل و اعمال تغییر وضعیت فرایندی که فایل مورد نیاز آن حذف شده است، فقط در ابتدای کوانتومی که این فرایند برای اجرا در آن انتخاب شده است، انجام می‌شود.

برای نمونه، اگر فرایند p برای اجرا به فایل f نیاز داشته باشد و فایل f حذف شود، تنها در ابتدای کوانتومی که p برای اجرا انتخاب شده است، وجود یا عدم وجود فایل مورد نیاز آن بررسی می‌شود و در هنگام حذف فایل این بررسی انجام نمی‌گیرد. بنابراین فرایند p در ابتدای کوانتوم اجرای خود به دلیل عدم وجود فایل مورد نیاز، از ReadyPriorityQueue خارج می‌شود، حالت آن از READY به WAITING(FILE) تغییر کرده و وارد WaitingQueue می‌شود.

۷) حالت‌های فرایند

هر فرایند در هر لحظه دقیقاً در یکی از حالت‌های زیر است:

i. RUNNING

- در حال استفاده از CPU
- در هر لحظه فقط یک فرایند می‌تواند RUNNING باشد.

ii. READY

- قابل اجرا است
- در ReadyPriorityQueue قرار دارد

iii. WAITING

- قابل اجرا نیست
- در WaitingQueue قرار دارد
- دلایل انتظار:
  - WAITING(FILE): فایل موردنیاز موجود نیست.
  - WAITING(READY_LIMIT): به معنای غیرقابل اجرا بودن موقت به دلیل محدودیت ظرفیت ReadyPriorityQueue است.

۸) زمان‌بندی CPU و کوانتوم

- در هر کوانتوم فقط یک فرایند اجرا می‌شود.
- اگر هیچ فرایند قابل اجرایی وجود نداشته باشد، CPU بیکار است و خروجی معادل NONE خواهد بود.

۸.۱) انتخاب فرایند برای اجرا

در ابتدای هر کوانتوم:

```
// At the start of each Execution Quantom
while ReadyPriorityQueue not empty:
    p = extract_max()
    if p is executable:
       process p is chosen for execution
       break
    else:
       p.state = WAITING
       p.waitReason = FILE
       WaitingQueue.enqueue(p)
```

۸.۲) اجرای فرایند در کوانتوم

در زمان‌بندی کوانتومی، یک بازه‌ی زمانی ثابت به نام کوانتوم (مثلاً ۲۰ واحد زمانی) تعیین می‌شود و زمان‌بند سیستم‌عامل در هر مرحله، فرایندی با بالاترین اولویت را از صف اولویت انتخاب کرده و وضعیت آن را به RUNNING تغییر می‌دهد؛ سپس آن فرایند، حداکثر به اندازه‌ی یک کوانتوم اجرا می‌شود، به‌طوری‌که اگر زمان باقی‌مانده‌ی اجرای آن بیشتر یا مساوی کوانتوم باشد، به اندازه‌ی کوانتوم از آن کم می‌شود و فرایند برای نوبت بعدی Timeout می‌گردد. اگر زمان باقی‌مانده اجرای فرایند، کمتر از کوانتوم باشد، فرایند فقط به همان مقدار اجرا شده و کارش به پایان می‌رسد. بنابراین هنگام پایان کوانتوم یکی از حالت‌های زیر رخ می‌دهد:

- اگر زمان باقی مانده از اجرای فرایند برابر صفر شود (remainingTime = 0)، فرایند پایان می‌یابد.
- اگر با پایان کوانتوم، زمان باقی مانده فرایند مخالف صفر باشد (exec = quantum و remainingTime > 0)، Timeout رخ می‌دهد.

۸.۳) Timeout

پس از وقوع Timeout، فرایند از لیست فرایندهای ReadyPriorityQueue خارج شده و اولویت آن بر اساس میزان استفاده از CPU به شکل زیر به‌روزرسانی می‌شود. در عبارت زیر، q تعداد کوانتوم‌هایی است که فرایند تا این لحظه برای اجرا گرفته است و α ضریب شدت کاهش اولویت می‌باشد.

$$newPriority = currentPriority / (\alpha \times q + 1)$$

این روش باعث کاهش تدریجی اولویت فرایندهایی می‌شود که سهم بیشتری از CPU مصرف کرده‌اند، در حالی که رفتار سیستم قابل پیش‌بینی و منصفانه باقی می‌ماند. در این سیستم، حداقل اولویت فرایندها برابر ۱ و α نیز برابر ۰.۵ در نظر گرفته می‌شود.

پس از تغییر اولویت فرایند Timeout شده، با توجه به اولویت جدید فرایند و فرایندهای منتظر در waitingQueue، ممکن است فرایند یا فرایندهای جدیدی از waitingQueue که حالت WAITING(FILE) دارند، برای افزودن به ReadyPriorityQueue انتخاب شوند. شبه کد زیر برای درک بهتر الگوریتم بیان شده، ارائه شده است.

```
of Timeout of process p:

    update priority of p

     while (!waitingQueue.isEmpty()):

        if (waitingQueue.peak().waitReason == FILE)
            break

        if (waitingQueue.peak().priority > p.priority):
            ReadyPriorityQueue.insert(waitingQueue.dequeue())
        else:
            break

    if (ReadyPriorityQueue is not full):
        ReadyPriorityQueue.insert(p)
    else:
        waitingQueue.enqueue(p)
```

۹) آزاد شدن ظرفیت ReadyPriorityQueue

در صورت پایان یافتن اجرای یک فرایند، ممکن است ظرفیتی در ReadyPriorityQueue آزاد شود. در این حالت، سیستم‌عامل بلافاصله WaitingQueue را بررسی می‌کند تا در صورت وجود فرایندهای متوقف شده به دلیل محدودیت ظرفیت ReadyPriorityQueue، یک فرایند قابل اجرا را به ReadyPriorityQueue منتقل کند. بنابراین تنها فرایندهایی که به دلیل READY_LIMIT در WaitingQueue قرار گرفته‌اند، واجد شرایط انتقال فوری به ReadyPriorityQueue هستند.

در صورت وجود ظرفیت در ReadyPriorityQueue:

- یک فرایند از WaitingQueue که دلیل توقف آن، READY_LIMIT بوده است، انتخاب می‌شود.
- وضعیت آن از WAITING به READY تغییر می‌یابد و فرایند به ReadyPriorityQueue افزوده می‌شود.

قوانین این بخش فقط در صورت پایان کامل اجرای فرایند اعمال می‌شوند و شامل حالت Timeout نمی‌گردند. بنابراین این بررسی‌ها فقط در نقاط پایان اجرای فرایند انجام می‌شود و منجر به وقفه یا قطع اجرای کوانتوم جاری نمی‌گردد.

به صورت کلی، هنگامی که ظرفیتی در ReadyPriorityQueue خالی می‌شود، فرایندهای قابل انتقال براساس معیار **اولویت بیشتر** و در صورت تساوی اولویت‌ها به ترتیب FIFO، منتقل می‌شوند. این عمل تا تکمیل ظرفیت ReadyPriorityQueue یا عدم وجود فرایند برای افزودن به ReadyPriorityQueue ادامه می‌یابد.

۱۰) ساختار ورودی

ورودی سیستم به‌صورت جریان دستورات متنی دریافت می‌شود و دستورات فقط در فاز دریافت ورودی خوانده می‌شوند. اجرای آن‌ها در چرخه‌های بعدی انجام می‌شود. در هر چرخه اجرا، سیستم‌عامل حداکثر y دستور از ورودی دریافت می‌کند. در صورتی که تعداد دستورات موجود در ورودی بیش از y باشد، دستورات باقی‌مانده به چرخه‌های بعدی موکول می‌شوند.

۱۰.۱) دستور ایجاد فرایند

PROCESS_CREATE PID=<int> PARENT=<int> PRIORITY=<int> BURST=<int> [FILE=<path>]

قوانین:

- اگر هدر FILE در دستور PROCESS_CREATE وجود نداشته باشد، مقدار needsFile برابر false خواهد بود.
- remainingTime در لحظه ایجاد برابر burstTime است.

مانند:

PROCESS_CREATE PID=3 PARENT=1 PRIORITY=4 BURST=30 FILE=/bin/user

۱۰.۲) دستور حذف فرایند

PROCESS_DELETE PID=<int>

۱۰.۳) ساختار دستور ایجاد فایل

FILE_CREATE PATH=<path>

مانند:

FILE_CREATE PATH=/bin/user/mehdi.file

۱۰.۴) ساختار دستور حذف فایل

FILE_DELETE PATH=<path>

مانند:

FILE_DELETE PATH=/bin/user/mehdi.file

۱۰.۵) دستور پایان برنامه – خاموش کردن سیستم‌عامل

پس از دریافت SHUTDOWN، سیستم پس از پایان چرخه جاری متوقف می‌شود.

SHUTDOWN

در صورت عدم تطابق هر یک از دستورات با وضعیت سیستم، خطاهای مناسب باید پرتاب و مدیریت شوند.

۱۱) مدیریت استثنا

سیستم‌عامل موظف است در برابر خطاهای منطقی و ورودی نامعتبر، رفتار مشخص و قابل پیش‌بینی داشته باشد. بنابراین لازم است در صورت بروز خطا، پیام مناسب ثبت شده و سیستم به اجرای خود ادامه دهد. تنها حالتی که بروز خطا موجب توقف برنامه می‌شود، حالتی است که خطای دریافت شده مربوط به پیکربندی سیستم‌عامل باشد. بروز دیگر خطاها، سبب توقف برنامه نشده و همچنین خطاهای مربوط به یک دستور، فقط همان دستور را رد می‌کنند.
خطاهای احتمالی در این سیستم عبارتند از:

- InvalidParentException: در صورت ایجاد فرایند با شناسه پدر ناموجود
- ProcessNotFoundException: در صورت حذف فرایندی که وجود ندارد
- FileNotFoundException: در صورت نیاز فرایند به فایل ناموجود یا حذف فایل ناموجود
- InvalidCommandException: در صورت نادرست بودن ساختار دستور
- و دیگر خطاهای احتمالی در عملکرد سیستم‌عامل

نکته: پیاده‌سازی خطاهای سیستم به صورت کامل ضروری است.

۱۲) خروجی برنامه

۱۲.۱) لاگ پایانی هر چرخه اجرا

در پایان اجرای هر بازه که شامل x کوانتوم متوالی است، سیستم‌عامل یک لاگ خلاصه از وضعیت فرایندها تولید می‌کند. این لاگ فقط برای ارائه نمای کلی از اجرای سیستم در آن بازه است و شامل جزئیات زمان‌بندی داخلی نمی‌شود.

- قوانین ساده تولید لاگ:
  - فقط فرایندهایی که در طول این بازه وجود داشته‌اند، نمایش داده می‌شوند.
  - این لاگ فقط در پایان اجرای x کوانتوم تولید می‌شود.
  - برای هر فرایند یکی از موارد زیر چاپ می‌شود:
    - EXECUTED: فرایند حداقل یک کوانتوم اجرا شده است.

- WAITING(FILE): فرایند به دلیل نبود فایل موردنیاز اجرا نشده است.
- WAITING(READY_LIMIT): فرایند به دلیل پر بودن صف آماده اجرا نشده است.
- TERMINATED: فرایند در این بازه به پایان رسیده یا حذف شده است.
- ERROR: پیام خطای منطقی در ایجاد یا ابتدای اجرای فرایند رخ داده است.

برای نمونه، برای ۵ فرایند با PIDهای ۱ تا ۵ بدین صورت لاگ تولید می‌شود:

---CYCLE SUMMARY---

PID | RESULT

1|EXECUTED

2|WAITING(FILE)

3|EXECUTED

4|WAITING(READY_LIMIT)

5|ERROR

۱۲.۲) گزارش نهایی فایل‌ها

پس از ورود دستور SHUTDOWN، درخت فایل‌ها از ریشه تا رسیدن به هر فایل، با استفاده از نام فایل چاپ شود. مانند:

/bin/user/mehdi.file

۱۳) بخش امتیازی

در این پروژه، علاوه بر الزامات اصلی، قابلیت‌های زیر به‌عنوان بخش‌های امتیازی در نظر گرفته شده‌اند که باعث افزایش دقت شبیه‌سازی و نزدیک‌تر شدن رفتار سیستم به یک سیستم‌عامل واقعی می‌شوند.

۱۳.۱) رابط کاربری گرافیکی

به‌عنوان یکی از بخش‌های امتیازی، برای این پروژه یک رابط کاربری گرافیکی (GUI) در نظر گرفته شده است که هدف آن نمایش بصری وضعیت سیستم و تسهیل بررسی رفتار زمان‌بندی فرایندها می‌باشد.

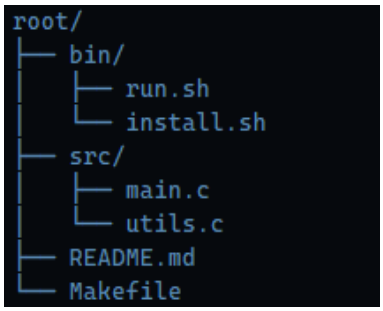
رابط گرافیکی سیستم می‌تواند شامل موارد زیر باشد:

- نمایش درخت فرایندها به‌صورت گرافیکی و سلسله‌مراتبی
- نمایش وضعیت هر فرایند با رنگ‌بندی مجزا
- نمایش صف READY و WaitingQueue به‌صورت لیستی و به‌روزرسانی در هر کوانتوم
- نمایش زمان سیستم و بازه‌ی کوانتوم جاری

- نمایش پیام‌های رویدادها مانند:
  - ایجاد یا حذف فرایند
  - ایجاد فایل در File Tree

۱۳.۲) نمایش درخت فایل به سبک Linux

برای نمایش این نوع خروجی، قالب نمایش به سبک نمایش سیستم‌عامل لینوکس در نظر گرفته می‌شود؛ یعنی ساختار درختی با تورفتگی (Indentation) مناسب نمایش داده می‌شود، به‌گونه‌ای که دایرکتوری‌ها و فایل‌های داخل آنها با فاصله یا نمادهای متداول لینوکس مانند نمایش درختی دستور tree، زیر هم قرار می‌گیرند و خواننده می‌تواند به‌سادگی ساختار کلی سیستم فایل را درک کند. تصویر زیر نمونه‌ای از خروجی مورد انتظار است.



شکل ۲- تصویر نمونه خروجی ساختار فایل به سبک سیستم‌عامل لینوکس

۱۳.۳) نمایش بلادرنگ وضعیت زمان‌بند و پردازه‌ها

به‌منظور افزایش درک نحوه عملکرد الگوریتم زمان‌بندی و شفاف‌سازی فرایند اجرای پردازه‌ها، یک بخش نمایش بلادرنگ (Real-Time Monitoring) به شبیه‌ساز سیستم‌عامل افزوده شده است. در این بخش، گذر زمان سیستم با استفاده از یک Logical Clock شبیه‌سازی شده و در هر واحد زمانی، وضعیت پروسه در حال اجرا بر روی CPU به‌صورت پویا نمایش داده می‌شود.

همچنین در هر لحظه، اطلاعات مربوط به پروسه فعال (Running Process)، میزان زمان باقی‌مانده از کوانتوم و وضعیت صف‌های سیستم در خروجی کنسول نمایش داده می‌شود.

با اتمام کوانتوم یا وقوع رویدادهای زمان‌بندی، عملیات لازم انجام شده و فرایندها، مطابق با الگوریتم زمان‌بندی مورد نظر، بین صف‌ها جابه‌جا می‌شوند. این تغییرات به‌صورت آنی در وضعیت صف‌ها و پردازه‌ها به‌روزرسانی شده و به کاربر نمایش داده می‌شود.

هدف از پیاده‌سازی این بخش، تبدیل خروجی شبیه‌ساز از حالت ایستا به یک خروجی پویا و زمان‌محور است تا امکان مشاهده دقیق رفتار زمان‌بند، نحوه تخصیص پردازنده و تأثیر کوانتوم زمانی بر اجرای پردازه‌ها فراهم شود.

۱۳.۴) جلوگیری از گرسنگی

برای جلوگیری از گرسنگی (Starvation)، مکانیزم Aging ساده پیاده‌سازی می‌شود. فرایندهایی که برای بیش از تعداد مشخصی کوانتوم در حالت READY یا WAITING باقی بمانند، به‌صورت تدریجی افزایش اولویت دریافت می‌کنند. پیاده‌سازی این مکانیزم در این سیستم‌عامل، از امتیاز ویژه‌ای برخوردار خواهد بود. توجه داشته باشید این مکانیزم در ابتدای هر چرخه اجرا اعمال می‌شود.

برای درک بهتر مفاهیم گرسنگی و Aging می‌توانید از منابع زیر استفاده کنید.

- منبع ۱ – GeeksForGeeks
- منبع ۲ – TutorialsPoint

۱۳.۵) State Transition Log

سیستم‌عامل تمامی تغییر وضعیت فرایندها را در قالب لاگ زمانی ثبت می‌کند. این لاگ شامل زمان، شناسه فرایند، وضعیت قبلی و وضعیت جدید است. این لاگ زمانی می‌تواند به صورت نوشتاری یا گرافیکی باشد. مانند:

[t=60] PID=3 WAITING → READY

۱۳.۶) لاگ آماری نهایی

در پایان اجرای سیستم، گزارش آماری شامل میزان استفاده از CPU و میانگین زمان انتظار تولید می‌شود. مانند:

CPU Utilization = 82%
Average Waiting Time = 14

۱۳.۷) ارائه راهکارهایی برای بهبود سیستم

ارائه کتبی هرگونه راهکار یا الگوریتم که سبب بهبود عملکرد و کارایی سیستم در اجرا و مدیریت فرایندها شود، از امتیاز ویژه برخوردار خواهد بود. تسلط در ارائه دقیق منطق روش پیشنهادی معیار مهمی در امتیاز این بخش می‌باشد.

نکات تکمیلی

- **زبان پیاده‌سازی:** زبان‌های قابل ارزیابی شامل C++، Java و C# می‌باشد.
- **گروه‌بندی: این** پروژه به‌صورت **فردی و گروهی (حداکثر ۲ نفره)** انجام می‌شود.
- **بستر پیاده‌سازی:** توسعه و پیاده‌سازی این پروژه در بستر **گیت‌هاب** و **سامانه کوئرا** انجام می‌شود.
  - روند توسعه پروژه در قالب کامیت‌های **متوالی و معنادار** باید انجام گیرد.
  - ساخت **برنچ‌های متعدد** و رعایت **اصول نوشتار صحیح متن کامیت** توصیه می‌شود.
- **نحوه ارزیابی:** ارزیابی عملکرد پروژه به‌صورت تست‌کیس-محور و همچنین **ارائه حضوری** انجام می‌شود.
  - ارزیابی نهایی در قالب ارائه ۲۰ دقیقه‌ای و براساس **بارم‌بندی بخش‌های مختلف** انجام می‌شود.