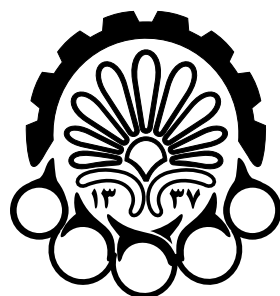


به نام خدا



دانشگاه صنعتی امیر کبیر
(پلی تکنیک تهران)

سیستم‌های عامل (بهار ۱۴۰۱)

فاز سوم

پیاده سازی الگوریتم‌های زمانبندی

استاد درس:

آقای دکتر جوادی

مهلت نهایی ارسال پاسخ:

۱۱ تیر ۱۴۰۱ (۸:۰۰ صبح)

تمدیدی نخواهیم داشت (به دلیل مهلت نهائی کردن نمرات درس).

مقدمه

در این قسمت می‌خواهیم الگوریتم‌های زمانبندی خودمان را جایگزین الگوریتم پیش فرض در xv6 کنیم. توصیه می‌کنیم که با جستجو و مطالعه فایل‌های سیستم عامل به دنبال چگونگی و کارکرد زمانبندی و تخصیص CPU به پردازنده‌ها بگردید. زمانبندی یکی از مهم‌ترین و پایه‌ای‌ترین مفاهیم موجود در هر سیستم عاملی است جایی که زمانبند باید یک سری اهداف که در بعضا تضاد هم هستند را برآورده کند، مانند:

- زمان پاسخ سریع^۱
- بازده خوب برای پردازنده‌های پس‌زمینه^۲
- جلوگیری از قحطی^۳
- برآورده کار کردن توان نیازهای پردازنده‌های با الویت پایین و الویت بالا^۴
- و ...

به مجموعه‌ای از قوانین که با استفاده از آنها یک پردازنده برای اجرا انتخاب می‌شود را خط‌مشی زمانبندی^۵ می‌گوییم. ابتدا باید با مطالعه کد xv6 و منابع موجود درباره آن یاد بگیریم که خط‌مشی زمانبندی موجود در xv6 چیست و چه پردازنده‌ای وظیفه انتخاب پردازنده‌ها برای اجرا را بر عهده دارد. بخش زمانبندی در کد xv6 را پیدا کنید و سعی کنید به سوالات زیر پاسخ دهید (صرفاً برای شروع کار):

- خط‌مشی پیش‌فرض چه پردازنده‌ای برای اجرای انتخاب می‌کند؟
- وقتی که یک پردازنده از رخداد IO برمی‌گردد، چه اتفاقی می‌افتد؟
- وقتی که یک پردازنده ایجاد می‌شود، چه اتفاقی می‌افتد و زمانبندی در چه زمان‌هایی و با چه فاصله زمانی انجام می‌شود؟

بخش اول : پیاده‌سازی الگوریتم‌های زمانبندی

۱) الگوریتم Round-Robin در xv6

در کد پیش‌فرض، زمانبند از سیاست round-robin و با فاصله زمانی یک clock tick کار تخصیص CPU را انجام می‌دهد. در این پروژه می‌خواهیم با تعیین زمان بین دو تخصیص CPU، بینیم روند پردازنده‌ها چگونه بهبود می‌یابد. ابتدا باید در فایل param.h یک پارامتر جدید مانند QUANTUM را با مقدار اولیه‌ای مانند ۱۰ تعریف کنیم.

```
#define QUANTUM <Number>
```

¹ fast process response time

² good throughput for background jobs

³ avoidance of process starvation

⁴ reconciliation of the needs of low-priority and high priority processes

⁵ scheduling policy

سپس در مکان مربوط به الگوریتم‌های زمان‌بندی، تغییرات لازم برای اینکه زمان‌بند به جای اینکه هر یک clock tick کار زمان‌بندی را انجام دهد، هر QUANTUM clock tick کار زمان‌بندی را انجام دهد.

لازم به ذکر است که در فایل گزارش خود طبق آزمون‌های تعریف شده در جلوتر، باید با تغییر مقدار QUANTUM ببینید که عملکرد پردازش‌ها بهبود می‌یابد یا خیر.

۲) الگوریتم زمان‌بندی طبق اولویت‌بندی (preemptive priority scheduling)

در این قسمت می‌خواهیم الگوریتم زمان‌بندی طبق اولویت را پیاده‌سازی کنیم که CPU طبق اولویت پردازش‌ها تخصیص داده می‌شود. این الگوریتم را از نوع **قبضه‌ای** پیاده کنید، به این معنی که اگر پردازش جدیدی ایجاد شود یا اینکه پردازش از رخداد IO برگردد و اولویت آن از پردازش در حال اجرا بیشتر باشد، این پردازش بایستی جایگزین پردازش در حال اجرا شود (برای پیاده‌سازی این ویژگی آموزنده وقت بگذارید). در این الگوریتم به هر پردازش باید یک عدد بین ۱ تا ۶ به عنوان اولویت تخصیص داده شود. اولویت پیش فرض را ۳ در نظر بگیرید و همچنین اگر اولویتی خارج از این بازه وارد شد به عنوان ۵ آن را بشناسد. در این روش، پردازش با عدد اولویت کمتر (به سمت ۱) شانس بیشتری برای اجرا شدن نسبت به دیگر پردازش‌ها دارد. از طرف دیگر، پردازش‌ها با اولویت ۶ کمترین شانس برای اجرا شدن را دارند. برای پیاده‌سازی، به ساختمان داده proc در فایل proc.h باید متغیرهای لازم اضافه شوند.

این خط‌مشی زمان‌بندی باید اینگونه کار کند که تخصیص CPU خود را به طور چرخشی (round-robin، اسلاید ۱۳ از lecture13) به پردازش‌های دارای اولویت بالاتر بدهد و پس از نبود برنامه در اولویت‌های بالاتر، به سراغ اولویت‌های کمتر برود. در ادامه برای تعیین و تغییر اولویت یک رویه نیاز به یک system call به نام setPriority داریم. در این سیستم‌کال یک عدد به عنوان ورودی گرفته می‌شود و طبق توضیحات گفته شده باید به عنوان اولویت پردازش‌های که سیستم‌کال را صدا زده، ثبت شود.

۳) الگوریتم زمان‌بندی طبق صف چند لایه (به شکل preemptive)

در این قسمت می‌خواهیم زمان‌بندی چند صفی و یا صف‌های چند لایه پیاده‌سازی کنیم. در روند پیاده‌سازی ابتدا باید ۶ صف بسازیم (صف‌های شماره ۱ تا ۶) و هر صف با الگوریتم زمان‌بندی round-robin اجرا می‌شود (با این تفاوت که صف‌ها با شماره کمتر دارای QUANTUM بیشتری نسبت به صف‌ها با شماره بیشتر هستند). اینکه QUANTUM زمانی صف‌های با اولویت بیشتر (شماره کمتر) چقدر بیشتر باشد در اختیار شما است. دقت کنید که پردازش‌های موجود در صف شماره ۲ در صورتی اجرا می‌شوند، که صف شماره ۱ خالی باشد. به همین ترتیب پردازش‌های موجود در صف شماره ۳، تنها در صورتی اجرا می‌شوند که صف شماره ۱ و ۲ خالی باشند. به شکل کلی پردازش‌های موجود در صف i تنها در صورتی اجرای می‌شوند که صف‌های شماره ۱ تا i-1 خالی باشند. دقت کنید که صف مورد نظر یک پردازش با استفاده از سیستم‌کال setPriority و با توجه به توضیحات بخش ۲ تعیین می‌شوند. دقت کنید که این الگوریتم هم از نوع قبضه‌ای است. به این معنی که اگر پردازش‌های با اولویت بالاتر به سیستم وارد شود در حالی که پردازش با اولویت پایین‌تر در حال اجرا است، پردازش با اولویت بالا بایستی که جایگزین پردازش با اولویت پایین‌تر شود.

۴) الگوریتم زمانبندی ؟ (اختیاری-نمره اضافی)

این بخش بزودی اعلام می‌شود.

بخش دوم: کدهای کمکی یا ارزیابی پیاده‌سازی‌های انجام شده

۱) changePolicy

جدای موارد خواسته شده همانطور که مشخص است نیاز داریم تا روند تخصیص CPU و زمان‌بندی را تغییر دهیم. برای این کار نیاز به پیاده‌سازی system call به نام changePolicy داریم. برای این کار ما به هر الگوریتم و قاعده زمان‌بندی‌مان یک عدد مانند ۰ و ۱ و ۲ اختصاص می‌دهیم (سعی کنید به الگوریتم پیش‌فرض عدد صفر و به باقی الگوریتم‌ها اعداد متناظر آنها در متن پروژه را تخصیص دهید). در این system call باید یک عدد به عنوان ورودی بگیریم و سپس روند زمان‌بندی برنامه را تغییر دهیم. با استفاده از این system call باید بتوانیم بین الگوریتم‌های گفته شده بالا و همچنین الگوریتم اصلی و پیش‌فرض خود XV6 تغییر وضعیت دهیم (مجموعاً ۴ الگوریتم و با الگوریتم امتیازی مجموعاً ۵ الگوریتم).

۲) قابلیت اندازه‌گیری زمان

در این قسمت می‌خواهیم به ساختمان داده هر پردازش در proc.h متغیرهایی اضافه کنیم تا با این‌ها بتوانیم بینیم CBT (Cpu Burst Time) و waitingTime و turnaroundTime هر برنامه چقدر است.

برای این کار نیاز است تا متغیرهایی مانند ready time, running time, termination time, creation time و sleeping time را نگه داریم و با هر کلاک CPU این مقادیر را برای هر process با توجه به موقعیت آن به روزرسانی کنیم.

لازم به ذکر است که برای باز پس‌گیری این مقادیر هنگام پایان کار نیز نیازمند به متدها و یا فراخوانی‌های سیستمی دارید که طبق سلیقه خودتان آنها را پیاده‌سازی کنید.

۳) تست نویسی

برای هر کدام از الگوریتم‌های پیاده‌سازی شده، نیاز است فایل تستی نوشته شود تا صحت عملکرد آن چک شود. همچنین نیاز داریم در انتها با قابلیت‌های اضافه شده برای اندازه‌گیری زمان، بتوانیم به بررسی این الگوریتم‌ها بپردازیم.

۳-۱) roundRobinTest

ابتدا برای الگوریتم Round-Robin نیاز به یک تست داریم. در این تست برنامه اصلی ما بوسیله fork، ۱۰ فرزند می‌سازد و سپس هر کدام از فرزندان در یک حلقه ۱۰۰۰ بار خط زیر را چاپ می‌کنند که i در این خط یک شمارنده از ۱ تا ۱۰۰۰ است.

/PID/ : /i/

در انتها نیز Turn Around Time، Waiting Time و CBT هر کدام از فرزندان باید نمایش داده شود و همچنین میانگین این‌ها نیز گفته شود.

سپس در ادامه چک شود با افزایش و کاهش مقدار Quantum خروجی ما چه تغییری می‌کند؟

۲-۳ prioritySchedTest

این بار برای الگوریتم Priority Scheduling که در قبل پیاده سازی کردیم یک تست باید بنویسیم. در این تست ابتدا پردازش ما باید ۳۰ فرزند تولید کند که ۵ فرزند اول دارای اولویت ۶، ۵ فرزند بعدی اولویت ۵ و ... و در انتها به ۵ فرزند آخر اولویت ۱ داده شود. سپس هر کدام از فرزندان در یک حلقه ۲۵۰ بار خط زیر را چاپ می‌کنند که i در این خط یک شمارنده از ۱ تا ۲۵۰ است.

```
/ChildNumber/ : /i/
```

برای این نیز Turn Around Time، Waiting Time و CBT هر کدام از فرزندان باید نمایش داده شود و همینطور میانگین این پارامترها برای کل فرزندان و هر کلاس اولویت گفته شود.

۳-۳ multiLayeredQueuedTest

این بار برای الگوریتم Multi layered queue که در قبل پیاده‌سازی کرده‌اید یک تست بنویسید. برای این کار باید ۶۰ فرزند تولید کنید و ۱۰ فرزند اول را به صف اول، ۱۰ فرزند دوم به صف دوم و ... اختصاص دهید. در نتیجه هر فرزند در یک حلقه ۲۰۰ بار خط زیر را چاپ کند که i در این خط یک شمارنده از ۱ تا ۲۰۰ است.

```
/ChildNumber/ : /i/
```

برای این نیز Turn Around Time، Waiting Time و CBT هر کدام از فرزندان باید نمایش داده شود و همینطور میانگین این پارامترها برای کل فرزندان و هر لایه صف گفته شود.

۴-۳ ؟ (اختیاری-نمره اضافی)

این بار برای الگوریتم ؟ که در قبل پیاده سازی کرده‌اید یک تست بنویسید.

نحوه تحویل پروژه

- گزارش پروژه (شامل خروجی تمامی تست‌ها) و فایل‌های اضافه شده یا تغییر داده شده در XV6 را در قالب یک فایل zip با نام project_report_group_id_sid1_sid2.zip بارگذاری کنید.
- پروژه دارای تحویل به صورت آنلاین است و توضیح کد و توضیح عملکرد سیستم عامل پرسیده می‌شود و **بخش مهمی از نمره** را در بر می‌گیرد. پس ضروری است که همه اعضای گروه به XV6 و پیاده‌سازی انجام شده تسلط داشته باشند.

- با توجه به انجام پروژه به شکل گروهی، تعداد commit های هر فرد باید متناسب باشد و خود گیت در حین تحویل چک می‌شود.
- لازم به ذکر است که تمامی پروژه‌ها پس از تحویل بررسی می‌شوند و هرگونه شباهت، به منزله نمره 0 برای پروژه خواهد بود.

موفق باشید

تیم درس سیستم‌های عامل