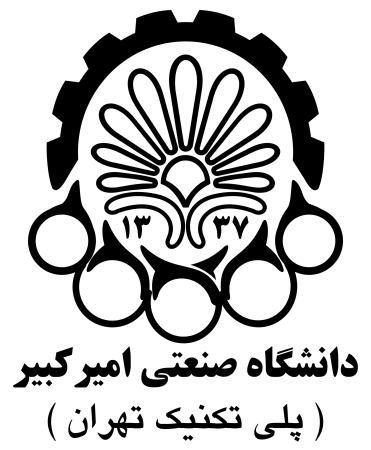
**به نام خدا**



سیستم‌های عامل (پاییز ۱۴۰۰)

فاز سوم

**پیاده سازی الگوریتم های زمانبندی**

**استاد درس:**

آقای دکتر جوادی

مهلت نهایی ارسال پاسخ:

---

***---***

**مقدمه**

در این قسمت می خواهیم الگوریتم‌های زمانبندی خودمان را جایگزین الگوریتم پیش فرض آن در xv6 کنیم. تماما توصیه می‌کنیم که با گشتن و مطالعه فایل‌های سیستم عامل به دنبال چگونگی و کارکرد زمانبندی و تخصیص cpu به رویه‌ها بگردید. زمانبندی یکی از مهم ترین و پایه ای ترین مفاهیم موجود در هر سیستم عاملی می باشد. زمانبند باید یک سری اهداف که در تضاد هم هستند را برآورده کند مانند fast process response time، good throughput for background jobs، avoidance of process starvation، reconciliation of the needs of low-priority and high priority processes و ... .

به مجموعه ای از قوانین که با استفاده از آنها یک پردازه برای اجرا انتخاب می شود را scheduling policy می گوییم. ابتدا باید بفهمیم که scheduling policy موجود در xv6 چیست و چه پردازه ای وظیفه ی انتخاب پردازه ها برای اجرا را داراست.

**بخش اول : پیاده سازی الگوریتم های زمانبندی**

1. **الگوریتم Round-Robin**

در اینجا میخواهیم با تعیین زمان بین دو تخصیص CPU، ببینیم روند برنامه چگونه بهبود می یابد. ابتدا باید در فایل param.h یک پارامتر جدید مانند QUANTUM را با مقدار اولیه‌ای مانند 10 تعریف کنیم. سپس در مکان مربوط به الگوریتم‌های زمان‌بندی، تغییرات لازم برای اینکار انجام دهید.

لازم به ذکر است که در فایل گزارش خود طبق تست‌های تعریف شده در جلوتر، باید با تغییر مقدار QUANTUM ببینید که عملکرد برنامه بهبود می یابد یا خیر.

1. **الگوریتم زمانبندی طبق اولویت بندی**

در این قسمت می خواهیم الگوریتم زمانبندی طبق اولویت را پیاده سازی کنیم که cpu طبق اولویت process ها قابل پس گیری است. در این الگوریتم به هر process باید یک عدد بین 1 تا 6 به عنوان اولویت داده شود. اولویت پیش فرض را 3 در نظر بگیرید و همچنین اگر اولویتی خارج از این بازه وارد شد به عنوان 5 آن را بشناسد. در این روند در نظر ما اولویت 1 از بقیه بیشتر بوده و به ترتیب تا عدد 6 اولویت‌ها همینطور کمتر می‌شوند و عدد 6 کمترین اولویت را دارد. برای پیاده‌سازی، به ساختمان داده proc در فایل proc.h باید متغیر‌های لازم اضافه شوند.

حال برنامه ما اینگونه باید کار کند که تخصیص CPU خود را به طور چرخشی به process های دارای اولویت بالاتر بدهد و پس از نبود برنامه در اولویت های بالاتر، به سراغ صف اولویت های کمتر برود.

در ادامه برای تعیین و تغییر اولویت یک رویه نیاز به یک system call به نام setPriority داریم. در این سیستم کال یک عدد به عنوان ورودی گرفته می شود و طبق سیستم گفته شده باید به عنوان اولویت پردازش ما ثبت شود.

1. **الگوریتم زمانبندی طبق صف چند لایه**

در این قسمت می خواهیم زمان بندی چند صف و یا صف های چند لایه پیاده سازی کنیم. در روند پیاده‌سازی ابتدا باید 4 صف بسازیم و هر صف با یک الگوریتم زمانبندی متفاوت با قبلی اجرا شود. به طور مثال پردازش‌های صف اول به طریق حالت پیش فرض XV6 زمانبندی شوند؛ پردازش های صف دوم به حالت اولویت بندی، پردازش‌های صف سوم به حالت عکس اولویت‌بندی قسمت قبل و پردازش‌های صف آخر به حالت  
 Round-Robin تخصیص CPU داده می شوند.

1. **الگوریتم زمانبندی طبق صف چند لایه ی پویا (اختیاری-نمره اضافی)**

این قسمت مانند قسمت قبل است با این تفاوت که یک پردازه نمی تواند به صورت دستی اولویت خود را تغییر دهد.

رعایت قوانین زیر الزامی است:

۱) فراخوانی سیستم کال exec اولویت پردازه را به حالت دیفالت ریست می کند(default priority)

۲) برگشت از حالت sleep mode(IO) باعث افزایش اولویت پردازه به بیشترین اولویت می شود(highest priority)

۳) واگذاری(Yielding) cpu به صورت دستی اولویت پردازه را تغییر نمی دهد

۴) اجرا شدن به اندازه ی یک full quanta باعث کاهش اولویت پردازه به اندازه ی ۱ واحد می شود.

توجه شود که برای این قسمت ممکن است نیاز به تعریف سیستم کال هایی جدید یا متغیرهای جدیدی برای پیاده سازی خود شوید.

**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 زمانبندی مان یک عدد مانند 0 و یا 1 و 2 اختصاص می‌دهیم. در این system call باید یک عدد به عنوان ورودی بگیریم و سپس روند زمان بندی برنامه را تغییر دهیم. با استفاده از این system call باید بتوانیم بین الگوریتم‌های گفته شده بالا و همچنین الگوریتم اصلی و پیش فرض خود XV6 تغییر وضعیت دهیم.

**۲) قابلیت اندازه گیری زمان**

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

برای این کار نیاز است تا متغیر هایی مانند creation time ، termination time، running time، ready 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 انها را پیاده سازی 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.

**۱-۳) roundRobinTest**

ابتدا برای الگوریتم Round-Robin نیاز به یک تست داریم. در این تست برنامه اصلی ما بوسیله fork، 10 فرزند می‌سازد و سپس هر کدام از فرزندان در یک حلقه 1000 بار خط زیر را چاپ می کنند که i در این خط یک شمارنده از 1 تا 1000 است.

/pID/ : /i/

در انتها نیز Turn Around Time، Waiting Time و CBT هرکدام از فرزندان باید نمایش داده شود و همچنین میانگین این ها نیز گفته شود.

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

**۲-۳) prioritySchedTest**

این بار برای الگوریتم Priority Scheduling که در قبل پیاده سازی کردیم یک تست باید بنویسیم. در این تست ابتدا process ما باید 30 فرزند تولید کند که 5 فرزند اول دارای اولویت 6، 5 فرزند بعدی اولویت 5 و در انتها به 5 فرزند آخر اولویت 1 داده شود. سپس هر کدام از فرزندان در یک حلقه 250 بار خط زیر را چاپ می کنند که i در این خط یک شمارنده از 1 تا 250 است.

/ChildNumber/ : /i/

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

**۳-۳) multiLayeredQueuedTest**

این بار برای الگوریتم Multi layered queue که در قبل پیاده سازی کرده اید یک تست بنویسید. برای این کار باید 40 فرزند تولید کنید و 10 فرزند اول را به صف اول، 10 فرزند دوم به صف دوم و ... اختصاص دهید. برای فرزندان درون صف های اولویت دار نیز اولویتی بدهید. در نتیجه هر فرزند در یک حلقه 200 بار خط زیر را چاپ کند که i در این خط یک شمارنده از 1 تا 200 است.

/ChildNumber/ : /i/

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

**4-3) DynamicMultiLayeredQueuedTest (اختیاری-نمره اضافی)**

این بار برای الگوریتم Dynamic Multi Layered Queue که در قبل پیاده سازی کرده اید 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 انلاین است و توضیح کد و توضیح عملکرد سیستم عامل پرسیده می شود و ***بخش مهمی از نمره*** را در بر می گیرد. 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 تمام پروژه و تمرین‌ها خواهد بود.

موفق باشید

تیم درس سیستم‌های عامل