

به نام خدا



دانشگاه صنعتی امیرکبیر  
(پلی تکنیک تهران)

سیستم‌های عامل (پاییز ۱۴۰۰)

## فاز سوم

پیاده سازی الگوریتم‌های زمانبندی

استاد درس:

آقای دکتر جوادی

دانشجویان:

کیانا آقاکثیری 9831006

سارا تاجرنسیا 9831016

## مقدمه

در این قسمت می خواهیم الگوریتم های زمانبندی خودمان را جایگزین الگوریتم پیش فرض در xv6 کنیم. توصیه می کنیم که با جستجو و مطالعه فایل های سیستم عامل به دنبال چگونگی و کارکرد زمانبندی و تخصیص CPU به پردازه ها بگردید. زمانبندی یکی از مهم ترین و پایه ای ترین مفاهیم موجود در هر سیستم عاملی است جایی که زمانبند باید یک سری اهداف که در بعض اتضاد هم هستند را برآورده کند، مانند:

- زمان پاسخ سریع<sup>۱</sup>
- بازده خوب برای پردازه های پس زمینه<sup>۲</sup>
- جلوگیری از قحطی<sup>۳</sup>
- برآورده کار کردن توامان نیازهای پردازه های با الویت پایین و الویت بالا<sup>۴</sup>
- و ... .

به مجموعه ای از قوانین که با استفاده از آنها یک پردازه برای اجرا انتخاب می شود را خطمشی زمانبندی scheduling policy) می گوییم. ابتدا باید با مطالعه کد xv6 و منابع موجود درباره ان یاد بگیریم که xv6 policy موجود در xv6 چیست و چه پردازه ای وظیفه انتخاب پردازه ها برای اجرا را بر عهده دارد. بخش زمانبندی در کد xv6 را پیدا کنید و سعی کنید به سوالات زیر پاسخ دهید (صرفا برای شروع کار):

- خطمشی پیش فرض چه پردازه ای برای اجرای انتخاب می کند؟
- وقتی که یک پردازه از رخداد IO برمی گردد، چه اتفاقی می افتد؟
- وقتی که یک پردازه ایجاد می شود، چه اتفاقی می افتد و زمانبندی در چه زمان هایی و با چه فاصله زمانی انجام می شود؟

برای پیاده سازی این پروژه ما به برخی توابع نیاز داریم تا بتوانیم thread ها را و زمان رسیدگی به آن ها را در process ها کنترل کنیم.

و در ادامه برای اینکه چگونه این زمان بندی را انجام دهیم از الگوریتم های متفاوتی با تست های متفاوت کمک خواهیم گرفت که بیشتر این تغییرات در کلاس های trap.c , proc.c , exec.c sysproc.c .

انواع توابع شامل گرفتن CBT هم waiting time هم turnaround time هم همچنین initialize priority آپدیت کردن custom wait ها صفحه ها و StateDurations است.

## بخش اول : پیاده‌سازی الگوریتم‌های زمانبندی

### ۱) الگوریتم Round-Robin در xv6

در کد پیش‌فرض، زمانبند یا scheduler 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 پردازه‌ها چگونه بهبود می‌یابد. ابتدا باید در فایل param.h یک پارامتر جدید مانند QUANTUM را با مقدار اولیه‌ای مانند ۱۰ تعریف کنیم.

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 بینید که عملکرد پردازه‌ها بهبود می‌یابد یا خیر.

### roundRobinTest (۱-۳)

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 در این خط یک شمارنده از ۱ تا ۱۰۰۰ است.

/PID/ : /۱/

در الگوریتم round robin ما به این صورت عمل می‌کنیم که timing به عنوان ۱۰ در نظر می‌گیریم و اگر به عنوان فرزند بودن آنها را print می‌کنیم. در ادامه waiting time را برای هر یک صدا می‌زنیم و در ادامه با استفاده از توابعی که در proc.c پیاده سازی کرده بودیم می‌توانیم به جزییات هر کدام از thread‌ها از جمله child number یا turnaround time و CBT دسترسی داشته باشیم.

در ادامه برای اجرای thread‌ها به صورت round robin با استفاده از Q به این صورت عمل می‌کنیم که در صف به اندازه زمان Q انجام شده و نوبت به thread بعدی میرسد.

با انجام چندین مثال مختلف برای  $Q$  و آزمون خطا متوجه شدیم که در این آزمایش بهترین مقدار برای  $Q$  برابر ۱۵ است به این صورت که در  $Q < 15$  و  $Q > 15$  مقدار waiting time بیشتر است.

## (۲) الگوریتم زمانبندی طبق اولویت‌بندی (non-preemptive priority scheduling)

در این قسمت می‌خواهیم الگوریتم زمانبندی طبق اولویت را پیاده‌سازی کنیم که `cpu` طبق اولویت پردازه‌ها تخصیص داده می‌شود. این الگوریتم را از نوع غیر-قبهای پیاده کنید، به این معنی که اگر پردازه‌ای جدیدی ایجاد شود یا اینکه پردازه‌ای از رخداد  $IO$  برگردد و اولویت آن از پردازه در حال اجراء باشد، پردازه در حال اجراء، کار خود را تا زمانی که کوانتم زمانی اجازه دهد، برای یک  $IO$  بلوکه شود یا `terminate` شود، ادامه می‌دهد. در این الگوریتم به هر پردازه 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، پردازه با عدد اولویت کمتر (به سمت ۱) شанс بیشتری برای اجرا شدن نسبت به دیگر پردازه‌ها دارد. از طرف دیگر، پردازه‌ها با اولویت ۶ کمترین شанс برای اجرا شدن را دارند. برای پیاده‌سازی، به ساختمان داده 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) اسلامی ۱۳ از lecture13) به پردازه‌های 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 `setPriority` call به نام `system` داریم. در این سیستم کال یک عدد به عنوان ورودی گرفته می‌شود و طبق توضیحات گفته شده باید به عنوان اولویت پردازه‌ای که سیستم کال را صدای ده ثبت شود.

## prioritySchedTest (۲-۳)

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

/ChildNumber/ : /i/

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

در الگوریتم زمانبندی طبق اولویت بندی (non-preemptive priority scheduling) ما مانند الگوریتم round robin عمل میکنیم با این تفاوت که برای اجرای threadها با استفاده از  $Q$  به این صورت عمل میکنیم که در صف به اندازه زمان  $Q$  انجام شده و براساس priority نوبت به thread بعدی میرسد. و از آنجایی که non preemptive است پس هنگام ورود یک thread با اولویت بالا تر باید حتما مقدار time  $Q$  برای اولویت در حال اجرا طی شود سپس به سراغ thread با اولویت بالا تر برود.

در تابع `test` آن هم با `thread` هایی را با اولویت های 6-1 در صف قرار میدهیم و هر دفعه با چک کردن یک سری `if` ها به `thread` میپردازیم که اولویت بالاتری نسبت به بقیه `thread` ها دارد.

### ۳) الگوریتم زمانبندی طبق صفت چند لایه (به شکل preemptive)

در این قسمت می‌خواهیم زمانبندی چند صفحه‌ی یا صفحه‌های چند لایه پیاده‌سازی کنیم. در روند پیاده‌سازی ابتدا باید ۶ صفحه‌ی (صفحه‌ای شماره ۱ تا ۶) و هر صفحه با الگوریتم زمانبندی round-robin اجرا می‌شود (با این تفاوت که صفحه‌ها با شماره کمتر دارای QUANTUM بیشتر نسبت به صفحه‌ها با شماره بیشتر هستند). اینکه QUANTUM زمانی صفحه‌ای با اولویت بیشتر (شماره کمتر) چقدر بیشتر باشد در اختیار شما است. دقت کنید که پردازه‌های موجود در صفحه شماره ۲ در صورتی اجرا می‌شوند، که صفحه شماره ۱ خالی باشد. به همین ترتیب پردازه‌های موجود در صفحه شماره ۳، تنها در صورتی اجرا می‌شوند که صفحه شماره ۱ و ۲ خالی باشند. به شکل کلی پردازه‌های موجود در صفحه ۱ تنها در صورتی اجرای می‌شوند که صفحه‌ای شماره ۱ تا ۱-i خالی باشند. دقت کنید که صفحه مورد نظر یک پردازه با استفاده از سیستم کال setPriority و با توجه به توضیحات بخش ۲ تعیین می‌شوند. به تفاوت کلیدی دیگر این بخش با بخش ۲ دقت کنید. در این بخش اگر پردازه‌ای با اولویت بالاتر به سیستم وارد شود در حالی که پردازه با اولویت پایین‌تر در حال اجرا است، پردازه با اولویت بالا بایستی که جایگزین پردازه با اولویت پایین‌تر شود. این تفاوت را جدی بگیرید و برای پیاده‌سازی آن زمان بگذارید و به نظر ما خیلی جذاب است.

### multiLayeredQueuedTest (۳-۳)

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

در این قسمت ما با استفاده از مقداری تعریف چندین فرزند آنها را در صفحه‌ای متناسب هر یک قرار میدهیم و آنها را طوری تعریف می‌کنیم که بتوان مقدار priority آن را آپدیت کرد.

در پی این کار هر دفعه که اولویت را آپدیت می‌کنیم (همه‌ی اطلاعات از جمله waiting time یا CBT را بروز رسانی می‌کنیم) برای thread turnaround time بعدی گزینه‌ای را انتخاب می‌کنیم که اولویت بالاتری دارد.

این کار را تا زمانی که به پایین ترین اولویت هم برسیم تکرار می‌کنیم و سپس میتوان خروجی از جمله waiting time را بدست آورد.