به نام خدا

سیستم‌های عامل- گروه ۱ (نیم‌سال اول ۹۹-۴۰۰)

پروژه پایانترم درس سیستم عامل

آخرین تاریخ بارگذاری پاسخ در courses:

ساعت ۲۳:۵۹ روز ۱۸ بهمن ۱۳۹۹

مقدمه

در این پروژه قصد داریم تا با سیستم عامل XV6 آشنا شویم و بتوانیم برخی پیاده‌سازی‌های ابتدایی را در این سیستم عامل انجام دهیم. روش است که اولین گام، بررسی و فهم چگونگی کارکرد این سیستم عامل است.

بخش اول: سوالات در مورد XV6

پاسخ سوالات زیر را فایل گزارش خود ارسال کنید. پاسخ به این سوالات در راستای فهم خود سیستم عامل XV6 است و در تسهیل موارد پیاده سازی به شما کمک می کند.

۱) به فایل proc.c مراجعه کنید. چه توابعی می بینید؟ این توابع چه کارایی‌هایی دارند و چگونه کار می کنند؟
۲) الگوریتم زمانبندی پیش فرض XV6 چیست؟ به کدام فایل‌ها برای تغییر در الگوریتم‌های زمانبندی نیاز داریم؟ این عملکرد را توضیح دهید.
۳) فراخوانی‌های سیستمی (System call) چگونه هستند و چگونه تعریف می‌شوند؟ تابع syscall در فایل syscall.c چه کارایی‌ای دارد و چه می‌کند؟

بخش دوم: کار با فراخونی‌های سیستمی

فراخونی‌های سیستمی زیر را پیاده سازی کنید.

۱) getParentID

برای پیاده‌سازی الگوریتم‌های زمانبندی، در ابتدا بایستی که چگونگی اضافه کردن فراخونی‌های سیستمی را یاد بگیریم. برای این فراخوانی سیستمی، می خواهیم هنگام فراخوانی شدن، process فعلی ما شماره process پدر (والد) خودش را برگرداند. به عنوان مثال رویه ای که PID پدرش 5 است، باید در قالب رشته و یا int مقدار 5 را برگرداند.

همچنین پس از ساخت و اضافه کردن این system call، نیاز داریم یک فایل به نام getParentIDtest.c نیز بسازید که بتوانید صحت کارکرد این system call را چک کنید. در این برنامه نیاز است که یک یا چند بار

تابع fork صدا زده شود و هر کدام از آنها خود و پدر خود را معرفی کند. به عنوان مثال خروجی یک process می تواند مانند زیر باشد:

“This is process 7 and the parent id is 5”

بدیهی است که برنامه تست ما باید در خود محیط XV6 اجرا شوند زیرا system call ما در این محیط تعریف شده است.

۲) getChildren

در این system call میخواهیم هنگام فراخوانی شدن، لیستی از PID تمام فرزندان process فعلی بازگردانده شود. به عنوان مثال اگر رویه‌ای با شماره 5 دارای 4 فرزند با شماره های 6 تا 9 باشد، باید این system call این شماره‌ها را در فرمتی دلخواه مانند ”6/7/8/9“ برگرداند.

همچنین پس از اضافه کردن این system call، نیاز دارید یک فایل به نام getChildrenTest.c نیز بسازید که بتوانید صحت کارکرد این system call را چک کنید. در این برنامه نیاز است که چند بار تابع fork صدا زده شود و تابع پدر این system call را صدا بزند و در انتها تمام فرزندانش را معرفی کند. به عنوان مثال، خروجی می‌تواند مانند مقابل باشد:

“This is process 5 and children are 6/7/8/9”

۳) getSyscallCounter

در این system call می‌خواهیم هنگام فراخوانی، یک شماره system call به عنوان ورودی وارد کنیم، و این فراخوانی باید چک کند که در process فعلی ما، system call با آن شماره تا به الان چند بار فراخوانی شده است (لیست فراخوانی‌های سیستمی در فایل syscall.h موجود است).

همچنین همانند موارد قبل، یک برنامه سطح کاربر به نام getSyscallCounterTest.c نیز برای تست system call نیاز است. که به طور مثال با دستور “getSyscallCounterTest 12” به ما تعداد دفعات فراخوانی system call شماره 12 را بدهد.

راهنمایی: برای پیاده سازی این قسمت می توان ساختمان داده هر process را طبق نیاز ما در فایل proc.h تغییر داد و هنگام فراخوانی توابع در proc.c مقادیر لازم را به روزرسانی کرد.

بخش سوم: پیاده سازی الگوریتم های زمانبندی

حال پس از پیاده‌سازی‌های بالا که کمی بیشتر با روند کارایی و کلیت XV6 آشنا شدیم، در این قسمت می خواهیم الگوریتم‌های زمانبندی خودمان را جایگزین الگوریتم پیش فرض آن کنیم. تماما توصیه می‌کنیم که با گشتن و مطالعه فایل‌های سیستم عامل به دنبال چگونگی و کارکرد زمانبندی و تخصیص cpu به رویه‌ها بگردید.

۱) 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 ۱۰ 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.

۲) 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 ۱ تا ۶ به عنوان اولویت 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 روند در نظر ما اولویت ۱ از بقیه بیشتر بوده و به ترتیب تا عدد ۶ اولویت‌ها همینطور کمتر می‌شوند و عدد ۶ 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.

۳) الگوریتم زمانبندی طبق صف چند لایه (اختیاری-نمره اضافی)

در این قسمت می خواهیم زمان بندی چند صف و یا صف های چند لایه پیاده سازی کنیم. در روند پیاده‌سازی ابتدا باید ۴ 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.

بخش چهارم: کدهای کمکی یا ارزیابی پیاده‌سازی‌های بالا

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.

۳-۱) 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؟

۳-۲) priorityTest

این بار برای الگوریتم Priority Scheduling که در قبل پیاده سازی کردیم یک تست باید بنویسیم. در این تست ابتدا process ما باید ۳۰ فرزند تولید کند که ۵ فرزند اول دارای اولویت ۶، ۵ فرزند بعدی اولویت ۵ 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 می کنند که i در این خط یک شمارنده از ۱ تا ۲۵۰ 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.

۴) ۳-۳ multiLayeredQueuedTest (اختیاری-نمره اضافی)

این بار برای الگوریتم Multi layered queue که در قبل پیاده سازی کرده اید یک تست بنویسید. برای این کار باید ۴۰ فرزند تولید کنید و ۱۰ فرزند اول را به صف اول، ۱۰ 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 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 هرکدام از فرزندان باید نمایش داده شود و همینطور میانگین این پارامتر ها برای کل فرزندان و هر لایه صف گفته شود.

نحوه تحویل پروژه

- گزارش پروژه (شامل خروجی تمامی تست‌ها) وفایل‌های 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.
- این پروژه ۳ نمره از ۲۰ نمره خواهد داشت. همچنین پروژه داری ۱ نمره امتیازی (از ۲۰) برای پیاده‌سازی الگوریتم multi layered queue و تست آن است. در صورت مشاهده گزارش کامل و حرفه‌ای نمره‌ای امتیازی به آن تعلق می گیرد.
- 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 تدریسیاری درس سیستم عامل