به نام خدا

آزمایشگاه سیستم عامل - بهار ۱۴۰۴

استاد: دکتر مهدی کارگهی

پروژه سوم: زمان‌بندی در xv6

طراحان: بابک حسینی محتشم - مهدی حاجی - علی عشقی موحد

---

1. اهداف پروژه

در این پروژه با مفهوم زمان‌بندی[1] به طور کلی و به پیاده‌سازی آن مفاهیم در سیستم عاملِ xv6 آشنا خواهید شد. در این راستا، ابتدا الگوریتم زمان‌بندی سیستم عامل xv6 بررسی خواهد شد. سپس به پیاده‌سازی الگوریتم زمان‌بندی با فرض ناهمگون[2] بودن پردازنده‌ها می‌پردازید. در نهایت با استفاده از فراخوانی‌های سیستمی و برنامهٔ‌های سطح کاربر، از صحت پیاده‌سازی خود اطمینان حاصل کرده و الگوریتم‌های زمان‌بندی پیاده‌سازی شده را ارزیابی خواهید کرد.

2. مقدمه

یکی از مهمترین وظایف سیستم عامل‌ها، تخصیص منابع سخت‌افزاری به برنامه‌های سطح کاربر است. در این امر، پردازنده که یکی از منابع فعال[3] می‌باشد، نیازمند یک زمان‌بند مناسب به منظور اجرا شدنِ هر چه بهترِ پردازه‌ها می‌باشد. از آنجایی که زمان‌بند، نیازمندِ دانستنِ اطلاعاتِ سیستم و همچنین وضعیت هرکدام از پردازه‌ها[4] می‌باشد، زمان‌بند سیستم‌عامل xv6 در سطح کرنل اجرا می‌شود.

---

[1] Scheduling

[2] Heterogeneous

[3] Active

[4] Processes

xv6 سیستم عاملی است که از مدل چند پردازشی متقارن[5] پشتیبانی می‌کند. با این قابلیت، هر پردازنده، یک کپی از کدِ اجراییِ زمان‌بند (تابع زمان‌بند) و همچنین متنِ[6] زمان‌بندِ خود را دارد و پس از انتخاب پردازه از صفِ پردازه‌هایِ آماده به اجرا[7]، توسط زمان‌بند (طبق الگوریتم زمان‌بندی)، با انجام عملیاتِ تعویض متن[8] از متنِ زمان‌بند به متن پردازه، اجرا خواهد شد.

یکی از ساده‌ترین و در عین حال کاربردی‌ترین الگوریتم‌های زمان‌بندی، زمان‌بندیِ نوبت گردشی[9] است که xv6 نیز از همین الگوریتم استفاده می‌کند. البته سیستم‌های عامل مدرن به‌دلیل نیازهای پیچیده‌تر مانند پاسخ‌دهی سریع به رویدادها، سربار کم، و اولویت‌بندی مناسب، از الگوریتم‌های بسیار پیشرفته‌تری استفاده می‌کنند.

با پیشرفت معماری کامپیوترها، چالش‌هایی مانند گرمایش زیاد[10] و محدودیت توان مصرفی مانع افزایش تعداد هسته‌های کاملاً قدرتمند در کنار یکدیگر شدند. هم‌زمان افزایش استفاده از دستگاه‌های کم‌مصرف مانند IOT و تلفن‌های همراه باعث شد مصرف انرژی به یک معیار کلیدی تبدیل شود.

به همین دلیل، در بسیاری از CPUهای مدرن—برای مثال در معماری‌های ARM big.LITTLE و همچنین پردازنده‌های هیبریدی اینتل (P-core / E-core)—ترکیبی از هسته‌های قدرتمند و هسته‌های کم‌مصرف در کنار یکدیگر قرار گرفته‌اند. در جدول ۱، چند مورد از الگوریتم‌های مورد استفاده در سیستم‌عامل‌های مختلف که ناهمگونی هسته‌ها را در نظر می‌گیرند، ارائه شده است.

جدول ۱. الگوریتم‌های زمان‌بندی استفاده شده در سیستم عامل‌های مختلف که ناهمگونی هسته‌ها را در نظر می‌گیرند.

سیستم عامل	الگوریتم زمان‌بندی	توضیحات
Windows 10/11 (Intel Hybrid / Alder Lake)	Hybrid-aware scheduler	ویندوز با کمک Thread Director اطلاعات زمان‌واقعی از نوع بار کاری دریافت می‌کند و بر اساس حساسیت به تأخیر، پردازه‌ها را روی هسته‌های قدرتمند P-cores[11] و پردازه‌های پس‌زمینه یا بهینه از نظر انرژی را

[5] Symmetric Multiprocessing (SMP)
[6] Context
[7] Ready Queue
[8] Context Switch
[9] Round Robin
[10] Overheating
[11] Performance cores

		روی E-cores[12] اجرا می‌کند. تصمیم‌گیری به‌صورت پویا و بر اساس feedback سخت‌افزاری انجام می‌شود.
Linux	CFS[13] + EAS[14]	الگوریتم CFS به هر پردازه، بر اساس سطح اولویت و میزان استفاده، زمان مناسبی برای پردازنده اختصاص دهد و اولویت‌های ثابت در آن کمتر دخیل هستند. زمان‌بندی بر اساس زمان مجازی[15] انجام می‌شود و اولویت‌ها به‌صورت پویا با وزن (nice values از ۲۰- تا ۱۹+) تنظیم می‌شوند. EAS ظرفیت انرژی/قدرت هر هسته را مدل‌سازی کرده و برای سیستم‌های big.LITTLE انتخاب هسته را با توجه به توازن بار[16]، مصرف انرژی، و تعاملی بودن وظیفه انجام می‌دهد.
macOS 11.0 (Darwin 20.0)	QoS-based heterogeneous scheduling	macOS از کلاس‌های کیفیت سرویس (QoS)[17] همراه با اطلاعات معماری ARM (مانند کلاسترهای big/LITTLE) برای تخصیص پردازه‌ها به هسته‌های مناسب استفاده می‌کند. وظایف تعاملی به هسته‌های قدرتمند و کارهای سبک به هسته‌های کم‌مصرف هدایت می‌شوند.
xv6	RR	همه فرآیندها از اولویت یکسانی برخوردارند و به ترتیب نوبتی اجرا می‌شوند، که باعث می‌شود زمان‌بندی مناسب برای سیستم‌های ساده و بی‌درنگ باشد، اما در مدیریت پیچیده‌تر اولویت‌ها و تسک‌ها دچار چالش است. زمان‌بند سیستم ناهمگونی پردازنده را در نظر نمی‌گیرد و همه هسته‌ها معادل فرض شده‌اند.

[12] Efficiency cores
[13] Completely Fair Scheduler
[14] Energy Aware Scheduling
[15] virtual runtime
[16] Load balancing
[17] Quality of Service classes

3. شرح پروژه

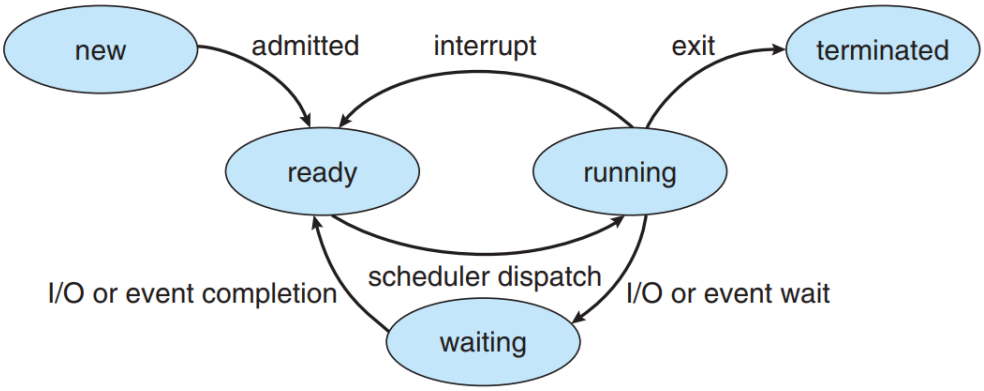
هدف:

آشنایی عمیق با مفاهیم پایهٔ زمان‌بندی و نحوهٔ تحقق آن‌ها در سیستم‌عامل xv6 قبل از شروع پیاده‌سازی.

توضیحات:

قبل از پیاده‌سازی زمان‌بند مورد نیاز در این تمرین، لازم است ابتدا با مفاهیم پایهٔ زمان‌بندی و نحوهٔ تحقق این مفاهیم در سیستم‌عامل xv6 آشنا شوید. این بخش به بررسی همین مفاهیم اختصاص دارد.

به‌طور کلی هر پردازه در چرخهٔ حیات خود، از وضعیت‌های مختلفی عبور می‌کند. شکل ۱ (برگرفته از فصل ۳ منبع درس) این چرخهٔ وضعیت‌ها را نشان می‌دهد و مسیر گذار پردازه میان حالت‌های مختلف را توصیف می‌کند.



شکل ۱. چرخهٔ وضعیت یک پردازه.

xv6 نیز از این قاعده مستثنا نیست و به منظور زمان‌بندی و مدیریت پردازه‌ها، وضعیت هر کدام از پردازه‌ها را در PCB[18] آن پردازه نگه می‌دارد. عمدهٔ عملیات زمان‌بندی و اقدامات مربوط به آن و همچنین داده‌ساختارهای مرتبط، در فایل‌های proc.c و proc.h قابل مشاهده هستند.

[18] Process Control Block

نتیجه آموزشی:

درک کامل از چرخهٔ حیات پردازه‌ها، ساختار PCB، و نحوهٔ مدیریت وضعیت‌های مختلف پردازه در سیستم‌عامل xv6.

۱) ساختار PCB و همچنین وضعیت‌های تعریف شده برای هر پردازه را در xv6 پیدا کرده و گزارش کنید. آیا شباهتی میان داده‌های موجود در این ساختار و ساختار به تصویر کشیده شده در شکل 3.3 منبع درس وجود دارد؟

۲) برای هر وضعیت پردازه در xv6 معادل وضعیت مربوطه در شکل ۱ را مشخص کنید.

1.3 تغییر وضعیت در xv6

هدف:

درک نحوهٔ گذار پردازه‌ها بین وضعیت‌های مختلف و شناسایی توابع مسئول این گذارها در xv6.

توضیحات:

همان‌طور که در شکل ۱ نیز مشاهده می‌کنید، هر پردازه در هر وضعیتی که باشد، توسط رویدادهای مختلف به وضعیت‌های دیگر گذار[19] می‌کند.

در این قسمت با گذارهای نمایش داده شده در شکل ۱ و معادل آن‌ها در xv6 آشنا می‌شوید.

نتیجه آموزشی:

توانایی شناسایی و تحلیل گذارهای وضعیت پردازه‌ها و درک نقش هر تابع در این فرآیند.

[19] Transition

Admitted 2.3

هدف:

درک فرآیند ایجاد و پذیرش پردازه‌های جدید در سیستم و شناسایی توابع مسئول این فرآیند.

توضیحات:

پس از اجرای سیستم‌عامل، اولین پردازه (initcode) توسط تابع userinit ساخته خواهد شد (main:36). در این تابع، مقداردهی‌های اولیه انجام می‌شود تا پردازهٔ مربوط به برنامهٔ سطحِ کاربرِ init.c ایجاد شود. در این برنامه، پردازهٔ init، عملیات fork را انجام می‌دهد تا پردازهٔ جدیدی تحت رابطهٔ فرزندی با آن ایجاد شود (init.c:24). این پردازهٔ جدید در واقع بستر اجرای بقیهٔ برنامه‌های سطح کاربر در xv6 یا همان shell می‌باشد که با آن کم و بیش برخورد داشته‌اید. در واقع تمامی برنامه‌های سطح کاربر fork و exec ای از پردازهٔ sh یا همان shell می‌باشند.

نتیجه آموزشی:

درک کامل از فرآیند ایجاد پردازه‌های جدید و نحوهٔ گذار آن‌ها از حالت new به ready.

۳) با توجه به توضیحات چرخهٔ وضعیت پردازه‌ها و ساختار xv6، مشخص کنید کدام توابع موجود در proc.c باعث گذار پردازه از حالت new به حالت ready در شکل ۱ می‌شوند. وضعیت یک پردازه در xv6 در این گذار از چه حالتی/حالت‌هایی به چه حالت/حالت‌هایی تغییر می‌کند؟

Scheduler Dispatch 3.3

هدف:

درک عمیق از مکانیزم زمان‌بندی و نحوهٔ انتخاب و اجرای پردازه‌ها توسط زمان‌بند در سیستم‌عامل xv6.

توضیحات:

با قرار گرفتن یک پردازه در حالت ready و پس از گرفتن نوبت توسط زمان‌بند، پردازه اجرا خواهد شد. تابع scheduler وظیفهٔ این نوبت‌دهی را بر عهده دارد (proc.c:323).

همانطور که پیش‌تر اشاره شد، هر پردازنده در xv6، یک اجرای مستقل از زمان‌بند خود را دارد. اما نکتهٔ حائز اهمیت این است که در xv6 تمامی اطلاعات پردازه‌های موجود، در یک جدول مشترک قرار دارند (proc.c:10). به منظور جلوگیری از رقابت[20] میانِ دو پردازنده در این جدول و ایجاد تغییرات همزمان؛ صف پردازه‌ها با استفاده از یک قفل محافظت شده است. قفل و مکانیزم قفل‌گذاری در ادامهٔ درس و همچنین آزمایشگاه بعد بررسی خواهد شد.

۴) سقف تعداد پردازه‌های ممکن در xv6 چه عددی است؟ در صورتی که یک پردازه تعداد زیادی پردازهٔ فرزند ایجاد کند و از این سقف عبور کند، کرنل چه واکنشی نشان داده و برنامهٔ سطح کاربر چه بازخوردی دریافت می‌کند؟

تابع scheduler در یک حلقه بی‌نهایت اجرا می‌شود و الگوریتم زمان‌بندی را اجرا می‌کند. ابتدا وقفه‌ها را فعال می‌کند و سپس با قفل کردن جدول پردازه‌ها (ptable)، پردازهٔ آماده اجرا (RUNNABLE) مطابق الگوریتم زمان‌بندی انتخاب می‌شود.

پس از انتخاب پردازه توسط زمان‌بند، به پردازنده‌ای که زمان‌بندِ آن فعال شده است واگذار و سپس عملیات تعویض متن انجام می‌شود و وضعیت پردازه به حالت RUNNING تغییر پیدا می‌کند. با پایان یافتن اجرای پردازه، سیستم به فضای کرنل برمی‌گردد و از ادامهٔ جایی که ابتدا تعویض متن رخ داده بود، اجرای زمان‌بندِ آن پردازنده، ادامه پیدا می‌کند (proc.c:346,347). پس، تابع scheduler یک بار تا انتهای صف می‌رود و هر پردازه را به شیوه‌ای که گفته شد اجرا می‌کند. بعد از بررسی آخرین پردازه در صف، قفل جدول پردازه‌ها را آزاد کرده و حلقهٔ بی‌نهایت دوباره از ابتدای صف شروع می‌شود و همین روند تکرار می‌شود.

نتیجه آموزشی:

درک کامل از الگوریتم زمان‌بندی نوبت گردشی، اهمیت قفل‌گذاری در سیستم‌های چندپردازنده‌ای، و نحوهٔ تعامل زمان‌بند با پردازه‌ها.

۵) چرا نیاز است در ابتدای هر حلقه تابع scheduler، جدول پردازه‌ها قفل شود؟ آیا در سیستم‌هایی که تنها یک پردازنده (یک هستهٔ پردازشی) دارند هم نیاز است این کار صورت بگیرد؟

۶) با فرض اینکه xv6 در حالت چند پردازنده‌ای (چند هسته‌ای) در حال اجراست، اگر یک پردازه آماده به اجرا شود، و صف پردازه‌ها در پردازنده دیگری در حال طی شدن باشد (proc:335)، در مکانیزم

[20] Race/Contention

زمان‌بندی xv6 نسبت به موقعیت پردازه در صف، در چه iteration ای امکان schedule پیدا می‌کند؟ (در همان iteration یا در iteration بعدی)

Context Switch 4.3

هدف:

درک عمیق از فرآیند تعویض متن و نحوهٔ ذخیره و بازیابی وضعیت پردازه‌ها در xv6.

توضیحات:

تعویض متن به فرایندی گفته می‌شود که در آن سیستم عامل یک پردازه در حال اجرا را متوقف کرده و یک پردازه جدید را اجرا می‌کند. همچنین در صورت وقوع یک وقفه[21] سیستم عامل نیاز دارد تا پردازه جاری را متوقف کرده و به وقفه رسیدگی کند.

در xv6، تعویض متن با استفاده از تابع swtch انجام می‌گیرد که وظیفهٔ ذخیره و بازیابی وضعیت را دارد:

```
void swtch(struct context *old, struct context *new);
```

این تابع وضعیت پردازه جاری را در old ذخیره می‌کند و وضعیت پردازه جدید را از new بارگذاری می‌کند. ساختار context وظیفهٔ نگهداری آخرین وضعیت پردازه را دارد.

نتیجه آموزشی:

درک کامل از مکانیزم تعویض متن، ساختار context، و نحوهٔ حفظ و بازیابی وضعیت رجیسترها در تعویض بین پردازه‌ها.

۷) رجیسترهای موجود در ساختار context را نام ببرید.

۸) همانطور که می‌دانید یکی از مهم‌ترین رجیسترها قبل از هر تعویض متن Program Counter است که نشان می‌دهد روند اجرای برنامه تا کجا پیش رفته است. با ذخیره‌سازی این رجیستر می‌توان محل

[21] interrupt

ادامهٔ برنامه را بازیابی کرد. این رجیستر در ساختار context چه نام دارد؟ شیوه ذخیره و بازنشانی این رجیستر را توضیح دهید.

Interrupt 5.3

هدف:

درک نقش وقفه‌ها در زمان‌بندی و نحوهٔ مدیریت آن‌ها در سیستم‌عامل xv6.

توضیحات:

تابع trap (trap.c:37) در سیستم‌عامل xv6 برای مدیریت وقفه‌ها به‌کار می‌رود. در صورتی که پردازه در حالت RUNNING باشد و وقفه تایمر (یعنی T_IRQ0+IRQ_TIMER) ایجاد شود، منجر به دو رویداد می‌شود. ابتدا زمان سیستم به اندازهٔ یک واحد افزایش پیدا می‌کند. که یکی از پردازنده‌ها مسئولیت این موضوع را بر عهده دارد (trap:50). رویداد دوم در صورتی که پردازه‌ای در حالت اجرا باشد، تابع yield برای آن فراخوانی شده و آن پردازه، پردازنده را رها می‌کند (trap:106).

۹) همانطور که در قسمت قبل مشاهده کردید، ابتدای تابع scheduler، وقفه به کمک تابع sti، فعال می‌شود. با توجه به توضیحات این قسمت، اگر وقفه‌ها فعال نمی‌شد چه مشکلی به‌وجود می‌آمد؟ آیا وقفه‌ای در این حالت قابلیت اجرا دارد؟ توضیح دهید.

۱۰) وقفهٔ تایمر هر چه مدت یک بار صادر می‌شود؟ (راهنمایی: می‌توانید با اضافه کردن یک cprintf پس از *ticks++* و یا با مراجعه به تابع lapicinit واقع در lapic.c این موضوع را مشاهده کنید.)

تابع yield (proc.c:386) وظیفه تغییر حالت پردازه از RUNNING به RUNNABLE را دارد؛ یعنی پردازه از اجرا خارج شده و پردازنده را رها می‌کند تا زمان‌بند در خصوص اجرای پردازه‌ها مجددا تصمیم بگیرد. فرایند آن به این شکل است:
ابتدا با قفل کردن جدول پردازه‌ها[22]، پردازه فعلی در جدول به حالت RUNNABLE تغییر وضعیت می‌دهد. سپس تابع sched فراخوانی می‌شود تا زمان‌بند، امکان انتخاب پردازه بعدی را داشته باشد. در

[22] Process Table (ptable)

پایان، قفل جدول پردازه‌ها آزاد می‌شود و کنترل به زمان‌بند برمی‌گردد. تابع sched (proc.c:366) در نهایت، یک تعویض متن از پردازهٔ کنونی، به تابع scheduler مربوط به همین پردازنده انجام می‌دهد.

نتیجه آموزشی:

درک کامل از نقش وقفه‌های تایمر در زمان‌بندی، مکانیزم yield و sched، و نحوهٔ محاسبه کوانتوم زمانی در الگوریتم نوبت گردشی.

۱۱) با توجه به توضیحات داده شده، چه تابعی منجر به انجام شدنِ گذار interrupt در شکل ۱ خواهد شد؟

۱۲) با توجه به توضیحات قسمت scheduler dispatch می‌دانیم زمان‌بندی در xv6 به شکل نوبت گردشی است. حال با توجه مشاهدات خود در این قسمت، استدلال کنید، کوانتوم زمانی این پیاده‌سازی از زمان‌بندیِ نوبت گردشی چند میلی ثانیه است؟

Wait 6.3

هدف:

درک مکانیزم انتظار پردازه‌ها برای رویدادهای مختلف و نحوهٔ آگاه‌سازی آن‌ها از وقوع این رویدادها.

توضیحات:

در صورتی که پردازه نیازمندِ به‌وقوع پیوستنِ اتفاقی برای ادامه انجام پردازش خود باشد، مانند عملیات I/O یا منتظر ماندن برای یک رویداد، برای آن صبر می‌کند. در این حالت، پس از به‌وقوع پیوستن اتفاق مورد نظر، سیستم عامل وظیفه دارد تا پردازه را از این اتفاق آگاه سازد.

یکی از این رویدادها، منتظر ماندن برای اتمام پردازه فرزند از طرف والد می‌باشد که به کمک تابع wait انجام می‌شود. این تابع به والد این امکان را می‌دهد تا منتظرِ خاتمهٔ کارِ یکی از فرزندان خود باشد و زمانی که یکی از آن‌ها به پایان رسید، والد را از اتمام آن آگاه و از سیستم پاک‌سازی کند.

نتیجه آموزشی:

درک کامل از مکانیزم همگام‌سازی بین پردازه‌ها، نحوهٔ انتظار برای رویدادها، و توابع مسئول آگاه‌سازی پردازه‌ها.

۱۳) تابع wait در نهایت از چه تابعی برای منتظر ماندن برای اتمام کار یک پردازه استفاده می‌کند؟

۱۴) با توجه به پاسخ سوال قبل، استفاده(های) دیگر این تابع چیست؟ (ذکر یک نمونه کافی است.)

۱۵) الف) چه تابعی در سطح کرنل، منجر به آگاه‌سازی پردازه از رویدادی است که برای آن منتظر بوده است؟ ب) این تابع منجر به گذار از چه وضعیتی به چه وضعیتی در شکل ۱ خواهد شد؟ ج) آیا توابع دیگری وجود دارند که منجر به انجام این گذار شود؟ نام ببرید.

Exit 7.3

هدف:

درک فرآیند خاتمه پردازه‌ها و مدیریت منابع آن‌ها پس از اتمام اجرا.

توضیحات:

در انواع حالت‌های پردازه‌ها، حالت ZOMBIE نشان‌دهندهٔ پردازه‌ای است که اجرای آن به پایان رسیده، اما هنوز منابع آن به‌طور کامل آزاد نشده و منتظر است تا والد از طریق wait آن را جمع‌آوری کند. با فراخوانی تابع exit در xv6، وضعیت پردازه به حالت ZOMBIE می‌رود و والدِ پردازه نیز به حالت RUNNABLE تغییر می‌کند تا در فرصت بعدی که اجرا شد، منابع فرزندِ خاتمه‌یافتهٔ خود را جمع‌آوری کند.

نتیجه آموزشی:

درک کامل از چرخهٔ حیات پردازه‌ها، وضعیت ZOMBIE، و رویکرد سیستم‌عامل در مدیریت پردازه‌های یتیم.

۱۶) در بخش ۳.۳.۲ منبع درس با پردازه‌های Orphan آشنا شدید، رویکرد xv6 در رابطه با این گونه پردازه‌ها چیست؟

4. زمان‌بندی هسته‌های ناهمگون:

هدف:

یادگیری اینکه CPUها یکسان نیستند.

خواسته:

فرض کنید در سیستم شما دو نوع هسته CPU وجود دارد. هسته‌های E-core که شماره CPUID آن‌ها عددی زوج است، هسته‌های با توان محاسبات کمتر ولی از لحاظ مصرف انرژی به صرفه‌ترند در حالی که هسته‌های P-core با CPUID فرد، هسته‌هایی قدرتمند ولی با مصرف بالای انرژی هستند. در ساختار CPU فیلد جدیدی اضافه کنید و نوع هسته را در آن نگه دارید.

نتیجه آموزشی:

سیستم‌عامل باید بداند سخت‌افزار چه ساختاری دارد.

۱۷) ساختار مربوط به CPU در xv6 پیدا کرده و با بررسی فیلدهای آن، گزارش کنید.

1.4 هسته‌های E با شماره CPUID زوج: زمان‌بند نوبت گردشی با کوانتوم زمانی

هدف:

پیاده‌سازی الگوریتم زمان‌بندی نوبت گردشی با کوانتوم زمانی مشخص برای هسته‌های کم‌مصرف.

خواسته:

برای شروع، ابتدا بدون تغییر فرکانس وقفهٔ تایمر، زمان‌بندی xv6 را به حالتی تغییر دهید که کوانتوم زمانی نوبت گردشی آن ۳۰ میلی ثانیه (بر حسب تیک زمانی سیستم) باشد. برای اطمینان از صحت پیاده‌سازی خود، xv6 را در حالتی اجرا کنید که تعداد پردازنده‌های مورد استفاده 1 عدد باشد. برای این کار می‌توانید در هنگام اجرای دستور make پرچم CPUS را برابر با 1 قرار دهید و یا در Makefile این متغیر را تغییر دهید.

توجه: در صورتی که نسخه QEMU شما از 6.2[23] بالاتر باشد، می‌بایست تغییراتی را در Makefile ایجاد کنید تا تنظیمات تعداد پردازنده‌ها به درستی اعمال شود. برای این کار تغییرات انجام شده در این لینک را اعمال کنید.

در قدم بعدی در فایل trap.c از cprintf استفاده کنید تا pid پردازه در هر تیک از اجرا و یا pid پردازه همراه با مدت زمان اجرای آن بر حسب تیک زمانی سیستم چاپ کنید. در نهایت برنامهٔ سطح کاربری بنویسید که درون خود چند پردازه ایجاد کند و هر پردازه مشغول انجام عملیاتی شود که به اندازه کافی طول می‌کشد. تصویری از خروجی اجرای سیستم عامل در این حالت را در گزارش خود قرار دهید.

راهنمایی: به دلیل وجود صف‌های دیگر در زمان‌بندی و اجرا از آنان، احتمالا نیاز داشته باشید تا اندیس آخرین پردازهٔ اجرا شده برای صف نوبت‌گردشی را نگه دارید تا در دفعهٔ بعد، از ادامهٔ صف، زمان‌بندی را از پی بگیرید. (به دلیل نیاز به تغییر شرایطی که در قسمت Scheduler Dispatch توضیح داده شد.)

راهنمایی پیاده‌سازی: در بعضی مواقع ممکن است پردازهٔ قابل اجرایی در سیستم حضور نداشته باشد. در این شرایط ممکن است (با توجه به پیاده‌سازی) در صورت عدم مقدار دهیِ p *struct proc واقع در تابع scheduler در هر iteration از حلقه بی‌نهایت (for(;;))، مقدار این متغیر، مقدار قدیمی خود را نگه دارد. این مسئله منجر به این می‌شود که پردازه‌ای که شرایط اجرا ندارد برای اجرا صادر شود که می‌تواند مشکلات زیادی را به وجود بیاورد. پیشنهاد می‌شود در ابتدای حلقه، مقدار p را به مقداری پیش‌فرض مانند NULL مقداردهی کنید و در انتهای منطق scheduler خود اگر پردازه این شرایط را نداشت (NULL نبود) برای اجرا صادر شود.

نتیجه آموزشی:

توانایی پیاده‌سازی الگوریتم زمان‌بندی نوبت گردشی با کوانتوم زمانی قابل تنظیم و درک اهمیت مدیریت صحیح متغیرها در زمان‌بند.

2.4 جدا کردن صف هر هسته (امتیازی)

هدف:

بهینه‌سازی زمان‌بندی در سیستم‌های چندپردازنده‌ای با ایجاد صف‌های جداگانه برای هر هسته و کاهش رقابت بر سر منابع مشترک.

[23] برای یافتن نسخه QEMU می‌توانید از دستور qemu-system-x86_64 --version استفاده کنید.

خواسته:

در سیستم‌عامل xv6 تنها یک صف وجود دارد که بین تمام هسته‌ها مشترک است. به همین دلیل دسترسی به این صف مشترک باعث ایجاد شرایط رقابتی[24] می‌شود و باید با استفاده از قفلی از دسترسی هم‌زمان به صف جلوگیری شود. با افزایش تعداد هسته‌ها دسترسی به صف می‌تواند گلوگاه شود. به همین دلیل در سیستم‌عامل‌های چند پردازشی، به ازای هر هسته صف جداگانه‌ای وجود دارد. با ایجاد تغییرات در xv6 به ازای هر هسته یک صف پردازه جداگانه تشکیل دهید که هر پردازنده تنها پردازه‌های موجود در صف خود را برای زمان‌بندی انتخاب می‌کند. بدین ترتیب در تمام مکان‌هایی که از جدول پردازه‌ها استفاده می‌شود نیز باید تغییر کنند تا از صف پردازنده‌ها استفاده کنند.

نتیجه آموزشی:

درک اهمیت معماری صف‌های جداگانه در سیستم‌های چندپردازنده‌ای و توانایی پیاده‌سازی آن برای کاهش رقابت و بهبود کارایی.

3.4 هسته‌های P با شماره CPUID فرد: زمان‌بند اولین ورود-اولین رسیدگی[25]

هدف:

پیاده‌سازی الگوریتم زمان‌بندی FCFS تغییر یافته برای هسته‌های قدرتمند که بر اساس زمان ایجاد پردازه عمل می‌کند.

خواسته:

با این الگوریتم زمان‌بندی در درس آشنا شده‌اید. در این روش زمان‌بندی، اولین پردازه‌ای که در صف آمده باشد، سریع‌تر انتخاب می‌شود. الگوریتم زمان‌بندی تا اتمام این پردازه به سراغ پردازه‌ی بعدی نخواهد رفت.

در این پروژه الگوریتم پیاده‌سازی متفاوت است بدین صورت که پردازه‌ای که زودتر ایجاد شده است را به عنوان پردازه اول در نظر می‌گیریم. بدین ترتیب اگر پردازه‌ای وارد صف شد که از پردازه در حال اجرای فعلی زودتر ایجاد شده بود، به جای آن اجرا می‌شود.

[24] Race Condition
[25] First Come First Served (FCFS)

راهنمایی: برای پیاده‌سازی این الگوریتم نیاز است زمان (tick) ایجاد هر پردازه را ذخیره کنید و با استفاده از آن، پردازه‌ای که از بقیه پردازه‌های این صف زودتر ایجاد شده است را پیدا کرده و پردازنده را برای اجرا به آن اختصاص دهید. توجه کنید که در اینجا الگوریتم بر اساس **زمان ایجاد پردازه** کار می‌کند نه **زمان ورود پردازه** به این صف.

نتیجه آموزشی:

توانایی پیاده‌سازی الگوریتم FCFS تغییر یافته و درک تفاوت بین زمان ایجاد و زمان ورود پردازه به صف در زمان‌بندی.

4.4 جابه‌جایی پردازه میان صف‌ها

هدف:

پیاده‌سازی مکانیزم توازن بار برای توزیع یکنواخت پردازه‌ها بین هسته‌های مختلف و بهینه‌سازی استفاده از منابع.

خواسته:

در سیستم‌های چندپردازنده‌ای، ممکن است برخی هسته‌ها تعداد زیادی پردازه در صف خود داشته باشند، در حالی که هسته‌های دیگر تقریباً بیکار باشند. این وضعیت باعث کاهش کارایی و افزایش زمان انتظار پردازه‌ها می‌شود. برای حل این مشکل، از روش‌های توازن بار[26] استفاده می‌شود. یکی از این روش‌ها هل دادن پردازه[27] است که هر CPU که بار بیشتری دارد، بخشی از پردازه‌های خود را به CPUهای سبک‌تر منتقل می‌کند.

البته این روش معایبی هم دارد از جمله آنکه شامل سربار جابه‌جایی پردازه‌ها است و برخلاف وابستگی پردازه‌ای[28] است. همچنین در پردازنده‌های ناهمگون هل دادن پردازه از هسته E به هسته P باعث مصرف بیشتر انرژی می‌شود. بنابراین معمولاً تنها زمانی انجام می‌شود که اختلاف بار قابل‌توجه باشد.

[26] Load Balancing
[27] Process Pushing
[28] Processor Affinity

در این پروژه هر پردازه‌ای که ایجاد می‌شود باید در صف هسته E که کمترین تعداد پردازه را دارد، اضافه شود. پردازه‌های init و sh باید همیشه در صف هسته‌های E قرار گیرند تا اجرای آن‌ها طبق الگوریتم نوبت گردشی تضمین شود.

هر هسته E هر 50 میلی‌ثانیه، تعداد پردازه‌های هسته‌های P را بررسی می‌کند. اگر تعداد پردازه‌های موجود در صف آن حداقل سه پردازه بیشتر از سبک‌ترین هسته P باشد، اولین پردازه اضافه شده به صف خود (به جز init و sh) را به صف آن هسته P هل می‌دهد.

نتیجه آموزشی:

درک اهمیت توازن بار در سیستم‌های چندپردازنده‌ای، آشنایی با الگوریتم‌های توازن بار، و توانایی پیاده‌سازی آن‌ها با در نظر گیری ملاحظات انرژی و کارایی.

5.4 ارزیابی الگوریتم‌های زمان‌بندی

هدف:

یادگیری نحوهٔ اندازه‌گیری و ارزیابی عملکرد الگوریتم‌های زمان‌بندی با استفاده از معیارهای استاندارد.

خواسته:

برای مقایسه‌ی الگوریتم‌های زمان‌بندی، معیارهای گوناگونی در درس معرفی شده‌اند. در این پروژه باید معیار گذردهی[29] را اندازه‌گیری کنید.

برای این منظور، باید دو فراخوانی سیستمی جدید پیاده‌سازی شود. این فراخوانی‌ها به برنامه‌های سطح کاربر اجازه می‌دهند که شروع و پایان بازهٔ اندازه‌گیری را مشخص کنند. با صدا زدن فراخوانی سیستمی شروع اندازه‌گیری در ابتدای برنامه، سیستم باید شمارنده‌های مورد نیاز را مقداردهی اولیه کند. با صدا زدن فراخوانی سیستمی دوم، اندازه‌گیری متوقف شده و معیار گذردهی حساب شده چاپ می‌شود.

نتیجه آموزشی:

توانایی طراحی و پیاده‌سازی فراخوانی‌های سیستمی برای اندازه‌گیری عملکرد، و درک نحوهٔ محاسبه و تحلیل معیارهای ارزیابی زمان‌بندی.

[29] Throughput

5.5 فراخوانی‌های سیستمی مورد نیاز:

هدف:

طراحی و پیاده‌سازی رابط‌های برنامه‌نویسی کاربردی برای دسترسی به اطلاعات زمان‌بندی و ارزیابی عملکرد سیستم.

خواسته:

فراخوانی سیستمی	توضیح
آغاز اندازه‌گیری گذردهی	این فراخوانی سیستمی در ابتدای برنامه سطح کاربر باید صدا زده شود تا زمان آغاز پردازه‌ها را برای اندازه‌گیری گذردهی مشخص کند.
پایان اندازه‌گیری گذردهی	این فراخوانی سیستمی در انتهای برنامه سطح کاربر باید صدا زده شود تا پایان کار پردازه‌ها را برای اندازه‌گیری گذردهی مشخص کند و این معیار را به صورت مناسب چاپ کند.
چاپ اطلاعات	با صدا زدن این فراخوانی سیستمی تمامی اطلاعات مربوط به پردازه از قبیل نام، شماره پردازه، وضعیت، صف پردازه، الگوریتم زمان‌بندی و طول عمر پردازه چاپ می‌شود.

6.5 برنامه‌های سطح کاربر

هدف:

ایجاد برنامه‌های تست برای اعتبارسنجی پیاده‌سازی الگوریتم‌های زمان‌بندی و نمایش صحت عملکرد آن‌ها.

خواسته:

در این بخش باید برنامه‌هایی در سطح کاربر بنویسید تا صحت پیاده‌سازی الگوریتم‌های زمان‌بندی شما را نشان دهند. هدف از این برنامه‌ها ایجاد شرایطی است که تغییرات انجام‌شده در زمان‌بند به‌خوبی قابل مشاهده باشند.

توجه کنید حتی الامکان از sleep استفاده نکنید و از حلقه‌های تو در تو و محاسبات متنوع استفاده کنید تا پردازه‌ها در صف برای اجرا قرار بگیرند و بتوانید از صحت پیاده‌سازی الگوریتم‌ها اطمینان حاصل کنید.

در برنامه سطح کاربر شما می‌بایست پردازه‌های مختلف ایجاد شوند. با استفاده از فراخوانی‌های سیستمی چاپ اطلاعات و آغاز و پایان اندازه‌گیری معیارها، سناریوهای مختلفی رقم بخورند و اطلاعات مورد نیاز در هر مقطع نمایش داده شده و در نهایت هم گذردهی سیستم‌عامل را گزارش شوند.

نتیجه آموزشی:

توانایی طراحی و پیاده‌سازی برنامه‌های تست جامع برای اعتبارسنجی سیستم‌عامل و درک اهمیت تست‌های عملی در توسعهٔ سیستم‌عامل.

سایر نکات

- پروژه خود را در یک مخزن خصوص در Github یا Gitlab پیش برده و در نهایت یک نفر از اعضای گروه کدها را به همراه پوشهٔ .git[30] زیپ کرده و در سامانه با فرمت OS-Lab3-<SID1>-<SID2>-<SID3>.zip آپلود نمایید.

[30] برای اطمینان حاصل کردن از وجود پوشه .git. می‌توانید گزینهٔ نمایش فایل‌های مخفی (Show Hidden Files) را فعال و یا از دستور ls -a استفاده کنید.

- رعایت نکردن مورد فوق کسر نمره را به همراه خواهد داشت.
- بخش خوبی از نمرهٔ شما را پاسخ دهی به سوالات مطرح شده تشکیل می‌دهد که به شما در درک نحوهٔ کارکرد xv6 و پیاده‌سازی قسمت زمان‌بندی کمک می‌کند.
- پاسخ سوال‌ها را مختصر و مفید بنویسید.
- در پیاده‌سازی خود در خصوص مواردی که ذکر نشده‌اند می‌توانید فرض منطقی و دلخواه خود را داشته باشید.
- زمان‌بندی شما می‌بایست هم در حالت تک‌پردازنده‌ای (CPUS=1) و هم در حالت چند پردازنده‌ای (CPUS >= 2) معتبر بوده و به درستی کار کند.
- تابع printf در سطح برنامه‌های کاربر و تابع cprintf در سطح کرنل و همچنین ابزار gdb، امکانات خوبی در هنگام عیب‌یابی می‌باشند، در صورت نیاز آن‌ها را به کار بگیرید.
- همه افراد می‌بایست به پروژه مسلط باشند و نمره تمامی اعضای گروه لزوما یکسان نخواهد بود.
- فصل پنجم کتاب xv6 بسیار مفید خواهد بود و مطالعهٔ آن توصیه می‌شود.
- تمامی مواردی که در جلسه توجیهی، گروه اسکایپ و فروم درس مطرح می‌شوند، جزئی از پروژه خواهند بود. در صورت وجود هرگونه سوال یا ابهام میتوانید با ایمیل دستیاران مربوطه یا گروه اسکایپی درس در ارتباط باشید.
- این تمرین صرفا برای یادگیری شما طرح شده است. در صورت محرز شدن تقلب در تمرین، مطابق با قوانین درس برخورد خواهد شد.
- در صورت استفاده از چت‌بات‌ها برای پاسخ به سوالات، لینک صفحات چت را در انتهای گزارش قرار دهید. توجه کنید که به نحوه پرسیدن سوال توسط شما نمره داده خواهد شد و پیشنهاد می‌شود از روش‌های مهندسی پرسش[31] استفاده کنید.

موفق باشید!

[31] Prompt engineering