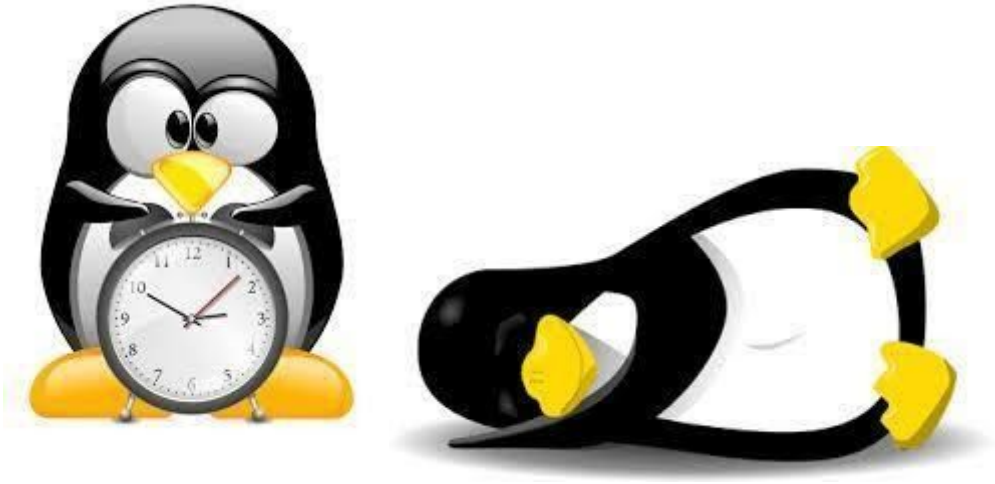
به نام خدا

آزمایشگاه سیستم‌عامل

پروژه سوم: زمان‌بندی پردازه‌ها

طراحان: فاطمه حقیقی، بهاران خاتمی



در این پروژه با زمان‌بندی در سیستم‌عامل‌ها آشنا خواهید شد. در این راستا الگوریتم زمان‌بندی xv6 بررسی شده و با ایجاد تغییرهایی در آن الگوریتم زمان‌بندی صف بازخوردی چندسطحی[1] (MFQ) پیاده‌سازی می‌گردد. هم‌چنین استفاده از فاکتور زمان در این سیستم‌عامل بررسی می‌گردد. در انتها توسط فراخوانی‌های سیستمی پیاده‌سازی شده، از صحت عملکرد زمان‌بند اطمینان حاصل خواهد شد.

مقدمه

همان‌طور که در پروژه یک اشاره شد، یکی از مهم‌ترین وظایف سیستم‌عامل، تخصیص منابع سخت‌افزاری به برنامه‌های سطح کاربر است. پردازنده مهم‌ترین این منابع بوده که توسط

[1] Multilevel Feedback Queue Scheduling

زمان‌بند[2] سیستم‌عامل به پردازه‌ها تخصیص داده می‌شود. این جزء سیستم‌عامل، در سطح هسته اجرا شده و به بیان دقیق‌تر، زمان‌بند، ریسه‌های هسته[3] را زمان‌بندی می‌کند.[4] دقت شود وظیفه زمان‌بند، زمان‌بندی پردازه‌ها (نه همه کدهای سیستم) از طریق زمان‌بندی ریسه‌های هسته متناظر آن‌ها است. کدهای مربوط به وقفه سخت‌افزاری، تحت کنترل زمان‌بند قرار نمی‌گیرند. اغلب زمان‌بندهای سیستم‌عامل‌ها از نوع کوتاه‌مدت[5] هستند. زمان‌بندی بر اساس الگوریتم‌های متنوعی صورت می‌پذیرد که در درس با آن‌ها آشنا شده‌اید. یکی از ساده‌ترین الگوریتم‌های زمان‌بندی که در xv6 به کار می‌رود، الگوریتم زمان‌بندی نوبت‌گردشی[6] (RR) است. الگوریتم زمان‌بندی صف بازخوردی چندسطحی با توجه به انعطاف‌پذیری بالا در بسیاری از سیستم‌عامل‌ها مورد استفاده قرار می‌گیرد. این الگوریتم در هسته لینوکس نیز تا مدتی مورد استفاده بود. زمان‌بند کنونی لینوکس، زمان‌بند کاملاً منصف[7] (CFS) نامیده می‌شود. در این الگوریتم پردازه‌ها دارای اولویت‌های مختلف بوده و به طور کلی تلاش می‌شود تا جای امکان پردازه‌ها با توجه به اولویتشان سهم متناسبی از پردازنده را در اختیار بگیرند. به طور ساده می‌توان آن را به نوعی نوبت‌گردشی تصور نمود. هر پردازه یک زمان اجرای مجازی[8] داشته که در هر بار زمان‌بندی، پردازه دارای کمترین زمان اجرای مجازی، اجرا خواهد شد. هر چه اولویت پردازه بالاتر باشد زمان اجرای مجازی آن کندتر افزایش می‌یابد. در جدول زیر الگوریتم‌های زمان‌بندی سیستم‌عامل‌های مختلف نشان داده شده است [2].

سیستم‌عامل	الگوریتم زمان‌بندی	توضیحات
Windows NT/Vista/7	MFQ	۳۲ صف ۰ تا ۱۵ اولویت عادی

[2] Scheduler

[3] Kernel Threads

[4] ریسه‌های هسته کدهای قابل زمان‌بندی سطح هسته هستند که در نتیجه درخواست برنامه سطح کاربر (در متن پردازه) ایجاد شده و به آن پاسخ می‌دهند. در بسیاری از سیستم‌عامل‌ها از جمله xv6 تناظر یک‌به‌یک میان پردازه‌ها و ریسه‌های هسته وجود دارد.

[5] Short Term

[6] Round Robin

[7] Completely Fair Scheduler

[8] Virtual Runtime

		۱۶ تا ۳۱ اولویت بی‌درنگ نرم
Mac OS X	MFQ	چندین صف با ۴ اولویت عادی، پر اولویت سیستمی، فقط مد هسته، ریسه‌های بی‌درنگ
FreeBSD/NetBSD	MFQ	بیش از ۲۰۰ صف
Solaris	MFQ	۱۷۰ صف
Linux < 2.4	MFQ	-
Linux < 2.6 ≥ 2.4	EPOCH-based	سربار بالا
Linux < 2.6.23 ≥ 2.6	(Scheduler O(1	پیچیده و سربار پایین
Linux ≥ 2.6.23	CFS	-
xv6	RR	-

زمان‌بندی در xv6

هسته xv6 از نوع با ورود مجدد[9] و غیرقبضه‌ای[10] است. به این ترتیب اجرای زمان‌بند تنها در نقاط محدودی از اجرا صورت می‌گیرد. به عنوان مثال، چنان‌چه در آزمایش دوم مشاهده شد وقفه‌های قابل چشم‌پوشی[11] قادر به وقفه دادن به یکدیگر نبوده و تنها امکان توقف تله‌های غیروقفه را دارند. هم‌چنین تله‌های غیروقفه نیز قادر به توقف یکدیگر نیستند. به طور دقیق‌تر زمان‌بندی تنها در زمان‌های محدودی ممکن است: ۱) هنگام وقفه تایمر و ۲) هنگام رهاسازی داوطلبانه شامل به خواب رفتن یا خروج توسط فراخوانی exit(). به خواب رفتن و فراخواندن

[9] Reentrant

[10] Nonpreemptive

[11] Maskable Interrupts

exit() می‌تواند دلایل مختلفی داشته باشد. مثلاً یک پردازه می‌تواند به طور داوطلبانه از طریق فراخوانی سیستمی sys_exit()، تابع exit() را فراخوانی نماید. هم‌چنین پردازه بدرفتار، هنگام مدیریت تله به طور داوطلبانه! مجبور به فراخوانی exit() خواهد شد (خط ۳۴۶۹). همه این حالات در نهایت منجر به فراخوانی تابع ۲۸۰۷) ()sched) و به دنبال آن اجرای تابع زمان‌بندی یا scheduler() می‌گردند (خط ۲۷۵۷).

۱) چرا فراخوانی sched() منجر به فراخوانی scheduler() می‌شود؟ (منظور، توضیح شیوه اجرای فرایند است.)

زمان‌بندی

همان‌طور که پیش‌تر ذکر شد، زمان‌بند xv6 از نوع نوبت‌گردشی است. به عبارت دیگر هر پردازه دارای یک برش زمانی[21] بوده که حداکثر زمانی است که قادر به نگه‌داری پردازنده در یک اجرای پیوسته می‌باشد. این زمان برابر یک تیک تایمر (حدود ۱۰ میلی‌ثانیه) می‌باشد.[31] با وقوع وقفه تایمر که در هر تیک رخ می‌دهد تابع yield() فراخوانی شده (خط ۳۴۷۵) و از اتمام برش زمانی پردازه جاری خبر خواهد داد.

زمان‌بندی توسط تابع scheduler() صورت می‌پذیرد. این تابع از یک حلقه تشکیل شده که در هر بار اجرا با مراجعه به صف پردازه‌ها یکی از آن‌ها که قابل اجرا است را انتخاب نموده و پردازنده را جهت اجرا در اختیار آن قرار می‌دهد (خط ۲۷۸۱).

[12] Time Slice

[13] تنظیمات تایمر هنگام بوت صورت می‌پذیرد.

۲) صف پردازه‌هایی که تنها منبعی که برای اجرا کم دارند پردازنده است، صف آماده[41] یا صف اجرا[51] نام دارد. در xv6 صف آماده مجزا وجود نداشته و از صف پردازه‌ها بدین منظور استفاده می‌گردد. در زمان‌بند کاملاً منصف در لینوکس، صف اجرا چه ساختاری دارد؟

۳) همان‌طور که در پروژه یک مشاهده شد، هر هسته پردازنده در xv6 یک زمان‌بند دارد. در لینوکس نیز به همین‌گونه است. این دو سیستم‌عامل را از منظر مشترک یا مجزا بودن صف‌های زمان‌بندی بررسی نمایید.

۴) در هر اجرای حلقه ابتدا برای مدتی وقفه فعال می‌گردد. علت چیست؟ آیا در سیستم تک‌هسته‌ای به آن نیاز است؟

۵) تابع معادل scheduler() را در هسته لینوکس بیابید. جهت حفظ اعتبار اطلاعات جدول پردازه‌ها، از قفل‌گذاری استفاده می‌شود. این قفل در لینوکس چه نام دارد؟

۶) وصله[61] PREEMPT_RT در سیستم‌عامل لینوکس چگونه پیش‌بینی‌پذیری[71] اجرا را افزایش می‌دهد؟ آیا سیستم کاملاً پیش‌بینی‌پذیر می‌شود؟ چرا؟

تعویض متن

پس از انتخاب یک پردازه جهت اجرا، توابع switchuvm() و switchkvm() حالت حافظه پردازه را به حالت جاری حافظه سیستم تبدیل می‌کنند. در میان این دو عمل، حالت پردازنده نیز توسط تابع swtch() از حالت (محتوای ساختار context (خط ۲۳۲۶) که ساختار اجرایی در هسته است) مربوط به زمان‌بند (کد مدیریت‌کننده سیستم در آزمایش یک که خود به نوعی ریسه هسته بدون پردازه متناظر در سطح کاربر است) به حالت پردازه برگزیده، تغییر می‌کند. تابع swtch() (خط ۳۰۵۸) دارای دو پارامتر old و new می‌باشد. ساختار بخش مرتبط پشته هنگام فراخوانی این تابع در شکل زیر نشان داده شده است.

esp + 8	new

[14] Ready Queue

[15] Run Queue

[16] Patch

[17] Predictibility

esp + 4	old
esp	ret addr

بخش مرتبط ساختار پشته پیش و پس از تغییر اشارهگر پشته (خط ۳۰۷۱) به ترتیب در نیمه چپ و راست شکل زیر نشان داده شده است.

	new
	old
	ret addr
	ebp
	ebx
	esi
esp	edi
	new'
	old'
	ret addr’
	ebp’
	ebx’
	esi’
esp'	edi'

اشارهگر به اشارهگر به متن ریسه هسته قبلی در old، متن ریسه هسته قبلی در پنج ثبات بالای پشته سمت چپ و اشارهگر به متن ریسه هسته جدید در new قرار دارد. اشارهگر به اشارهگر به متن ریسه هسته جدید در old’، متن ریسه هسته جدید در پنج ثبات بالای پشته سمت راست و اشارهگر به متن ریسه هستهای که قبلاً این ریسه هسته جدید به آن تعویض‌متن کرده بود، در new’ قرار دارد. متن ریسه هسته جدید از پشته سمت راست به پردازنده منتقل شده (خطوط ۳۰۷۴ تا ۳۰۷۸) و نهایتاً پردازه سطح کاربر اجرا خواهد شد.

زمان‌بندی بازخوردی چندسطحی

در این زمان‌بند، پردازه ها با توجه به اولویتی که دارند در سطوح مختلف قرار می‌گیرند که در این پروژه فرض شده است که سه سطح و متعاقباً سه اولویت وجود دارد. شما برای آزمودن زمان‌بند خود باید فراخوانی سیستمی را پیادهسازی کنید که بتواند پردازه را بین سطوح مختلف جابجا کرده تا

قادر به اعمال الگوریتم های مختلف هر صف باشید. همانطور که گفته شد زمان‌بندی که توسط شما پیاده سازی می‌شود دارای سه سطح می‌باشد که لازم است در سطح یک الگوریتم زمان‌بندی نوبت گردشی[81]، در سطح دوم الگوریتم زمان‌بندی بخت آزمایی[91] و در سطح سوم الگوریتم زمان‌بندی ابتدا بهترین کار[02] (BJF) را اعمال کنید. لازم به ذکر است که میان سطوح، اولویت وجود دارد. به این صورت که ابتدا تمام پردازه های سطح اول، سپس در صورت خالی بودن سطح اول، تمام پردازه‌های سطح دوم و در آخر در صورت خالی بودن هر دو سطح قبل، تمام پردازه‌های سطح سوم اجرا خواهند شد و شما با فراخوانی سیستمی که پیاده سازی می کنید می‌توانید سطح پردازه‌ها را تغییر دهید.

همچنین زمان‌بند پیادەسازی شده توسط شما باید دارای قابلیت افزایش سن[12] بوده و اگر پردازه‌ای بیشتر از زمانی معین اجرا نشود، آن پردازه را به سطح اول منتقل کند.

زمان‌بند نوبت گردشی

در این زمان‌بند یک واحد زمانی کوچکی به نام برش زمانی یا کوانتوم زمانی[22] داریم. در این زمان‌بند صف پردازه‌های آماده اجرا را به صورت یک صف حلقوی در نظر می‌گیریم، پردازه‌ها به صورت چرخشی، پردازنده را برای بازه حداکثر، یک کوانتوم زمانی در اختیار می‌گیرند.

به عبارت دیگر زمان‌بند، پردازه موجود در ابتدای صف را انتخاب نموده و یک تایمر برای پردازنده تنظیم می‌کند که پس از یک کوانتوم زمانی، پردازنده در اختیار پردازه دیگر قرار گیرد. پردازه‌ها در این نوع زمان‌بند به دو صورت عمل می‌کنند. حالت اول زمانی است که زمان مورد نیاز پردازه کمتر یا مساوی یک کوانتوم زمانی است، در این حالت پردازه به‌صورت داوطلبانه پردازنده را رها می کند. پس از آن پردازنده، پردازه بعدی که در ابتدای صف قرار دارد را انتخاب می‌نماید. حالت دوم، حالتی که زمان مورد نیاز پردازه بیشتر از یک کوانتوم زمانی است. در این حالت تایمر خاموش شده و منجر به وقفه در اجرا می‌گردد. سپس تعویض‌متن رخ داده و پردازه در انتهای صف اجرا قرار می گیرد. پس از آن پردازنده، پردازه ابتدای صف اجرا را انتخاب می‌کند.

[18] Round Robin

[19] Lottery

[20] Best Job First

[21] Aging

[22] Time Quantum

نکته‌ای که باید در پیاده‌سازی این الگوریتم رعایت شود این است که پردازه‌ها به ترتیب ورود به صف، اجرا خواهند شد و پردازه جدید، به انتهای زنجیره پردازه‌های در حال انتظار افزوده می‌شود.

زمان‌بند بخت‌آزمایی

این زمان‌بند بر پایه تخصیص منابع به پردازه‌ها به صورت تصادفی می‌باشد. ولی هر پردازه با توجه به تعداد بلیت شانش که دارد احتمال انتخاب شدنش به عنوان پردازه بعدی برای اجرا مشخص می‌شود. انتخاب پردازه برای اجرا توسط زمان‌بند پردازنده به این صورت می‌باشد که هر پردازه تعدادی بلیت شانس دارد و پردازنده به صورت تصادفی یک بلیت را انتخاب نموده و پردازه صاحب آن بلیت، اجرا خواهد شد. هنگامی که اجرای این پردازه توسط عواملی چون اتمام برش زمانی، مسدود شدن جهت عملیات ورودی/خروجی و ... به پایان رسید، روند مذکور تکرار خواهد شد.

هر بلیت معادل یک عدد طبیعی بوده و هر پردازه می‌تواند بازه‌ای از اعداد را به عنوان بلیت‌های شانس خود داشته باشد. زمان‌بند پردازه‌ها با تولید عددی تصادفی در بازه کل این اعداد، یک بلیت و متناظر با آن یک پردازه را برای اجرا انتخاب می‌کند. به عنوان مثال دو پردازه A و B داریم و A دارای ۶۰ بلیت شانس (بلیت‌های شماره ۱ تا ۶۰) و B دارای ۴۰ بلیت شانس (بلیت‌های شماره ۶۱ تا ۱۰۰) می‌باشد. زمان‌بند در هر مرحله، عددی تصادفی بین ۱ تا ۱۰۰ را انتخاب نموده و اگر عدد انتخاب شده بین ۱ تا ۶۰ باشد، پردازه A و در غیر این‌صورت پردازه B انتخاب می‌گردد. در شکل زیر مثالی از ۱۰ مرحله انتخابی توسط زمان‌بند پردازنده نشان داده شده است.

```
Ticket number -  73 82 23 45 32 87 49 39 12 09.
Resulting Schedule -  B  B  A  A  A  B  A  A  A  A.
```

زمان‌بند اول بهترین کار

در این بخش تقریبی از الگوریتم BJF پیاده‌سازی خواهد شد [1]. در این حالت لازم است برای پردازه زمان ورود و تعداد سیکل اجرا را مشخص نمایید. برای محاسبه زمان ورود می‌توانید از زمان سیستم هنگام ایجاد پردازه استفاده نموده و برای محاسبه تعداد سیکل اجرا، باید یک مشخصه برای پردازه خود با همین نام در نظر بگیرید. مقدار اولیه تعداد سیکل اجرا را هنگام ساخته شدن پردازه برابر با صفر در نظر بگیرید. به ازای هر بار اجرای پردازه، ۰.۱ واحد به تعداد سیکل

اجرایی آن بیفزایید. ابتدا معیاری را تحت عنوان رتبه برای هر پردازه تعریف می کنیم. این معیار برای هر پردازه به صورت زیر قابل تعریف است:

$$rank = (Priority \times Priority\_ratio) + (Arrival\ Time \times Arrival\ Time\_ratio) + (ExecutedCycle \times ExecutedCycle\_ratio)$$

در این فرمول، با داشتن اطلاعات در مورد اولویت، زمان ورود پردازه به صف و تعداد سیکل‌های اجرا شده هر برنامه می توانیم رتبه هر پردازه را داشته باشیم. در هر پردازه اولویت معادل با معکوس تعداد بلیت‌های آن پردازه می باشد.[32] سه ضریب معادله فوق توسط فراخوانی‌های سیستمی مربوطه مقداردهی می‌شود. زمان‌بندی بر اساس رتبه پردازه‌ها صورت می‌گیرد و اولویت اجرا با پردازه‌ای است که رتبه کمتری دارد.

نکته: پارامترهای جدیدی که برای الگوریتم‌های مختلف زمان‌بندی به پردازه اضافه می‌کنید و هنگام ایجاد پردازه، آن‌ها را مقداردهی می‌کنید باید به گونه‌ای مقداردهی شوند که به پردازه هایی که exec می‌شوند مانند پردازه‌هایی که توسط پوسته[42] ساخته و اجرا می‌شوند به سایر پردازه‌ها که تنها fork می‌شوند و exec نمی‌شوند اولویت دادهشود تا پوسته شما قفل نشود.

مکانیزم افزایش سن

همانطور که در کلاس درس فرا گرفتید، برای جلوگیری از گرسنگی[52]، اولویت پردازه‌هایی که مدت زیادی منتظر بوده و پردازنده به آن‌ها اختصاص نیافته به مرور افزایش می‌یابد. در زمان‌بندی که پیادسازی می‌کنید پردازه‌ها را به طور پیش‌فرض در صف دوم قرار دهید و در صورتی که پردازه‌ای ۱۰۰۰۰ سیکل منتظر مانده باشد آن را به صف اول منتقل کنید. در صورت بازانتقال این پردازه به صف های دیگر، رصد کردن تعداد سیکل اجرا نشده پردازه را از ابتدا از سر بگیرید.

فراخوانی‌های سیستمی مورد نیاز

[23] عدد اولویت پایین‌تر، بیانگر اولویت بیشتر است.

[24] Shell
[25] Starvation

۱)تغییر صف پردازه: پس از ایجاد پردازه‌ها (به تعداد لازم) باید با نوشتن یک فراخوانی سیستمی مناسب مشخص کنید که هر پردازه مربوط به کدام صف از سه صفی که پیادسازی کرده‌اید، تعلق دارد. همچنین باید بتوان یک پردازه را از یک صف به صف دیگری انتقال داد. این فراخوانی سیستمی، PID پردازه و شماره صف مقصد را به عنوان ورودی دریافت می‌کند.

۲) مقداردهی بلیت بخت‌آزمایی: باید به هرکدام از پردازه‌هایی که در صف اول قرار دارند تعدادی بلیت اختصاص دهید تا الگوریتم بخت‌آزمایی قابل اجرا باشد. بنابراین باید یک فراخوان سیستمی پیادسازی کنید که به پردازه‌های صف اول، بلیت مربوطه را تخصیص دهد. ورودی، PID پردازه مورد نظر و مقدار بلیت آن خواهد بود.

۳) مقداردهی پارامترهای BJF در سطح پردازه: طی این فراخوانی سیستمی، باید بتوان ضرایب موثر در محاسبه رتبه یک پردازه را تغییر داد. ورودی این فراخوانی سیستمی، PID پردازه مورد نظر و سه مقدار برای سه ضریب معادله می باشد.

۴) مقداردهی پارامترهای BJF در سطح سیستم: طی این فراخوانی سیستمی، باید بتوان ضرایب موثر در محاسبه رتبه را تغییر داد. ورودی این فراخوانی سیستمی سه مقدار برای سه ضریب معادله می باشد. این فراخوانی سیستمی مقدار ضرایب را برای تمام پردازه‌ها تغییر می‌دهد.

۵) چاپ اطلاعات: برای اینکه برنامه شما قابل تست باشد باید یک فراخوانی سیستمی پیادسازی کنید که لیست تمام پردازه‌ها را چاپ نموده و در هر سطر این لیست باید نام پردازه، شماره پردازه، وضعیت، شماره صف، تعداد بلیت بخت‌آزمایی، مقدار ضریب موثر، مقدار رتبه و تعداد سیکلی که پردازنده به آن پردازه اختصاص یافته است در آن گنجانده شود. یک مثال **نیمه‌کامل** در شکل زیر نشان داده شده است. توجه کنید **در صورتی که تمامی مقادیر فوق چاپ نشود نمره از شما کسر میگردد.**

```
name    pid     state           ticket
-----------------------------------------
init    1       SLEEPING        10
sh      2       SLEEPING        10
ps      48      RUNNING         1000
foo     15      SLLEPING        1000
foo     16      RUNNING         10
```

جهت حصول اطمینان از زمان‌بند خود، یک برنامه سطح کاربر با نام foo بنویسید که تعدادی پردازه در آن ساخته شده و پردازه‌ها عملیات پردازشی[62] انجام دهند تا فرصت بررسی عملکرد زمان‌بند وجود داشته باشد. می‌توان این برنامه را با اجرای دستور foo& در پس‌زمینه اجرا نموده و در این حین، توسط فراخوانی سیستمی چاپ اطلاعات از نحوه عملکرد آن مطلع شد. توجه کنید که در برنامه foo فراخوانی سیستمی صدا نمی‌شود. **فراخوانی‌های سیستمی فوق را به صورت برنامه سطح کاربری در بیاورید که بتوان آن را به صورت مستقیم از پوسته فراخوانی کرده** و آرگومان ها را به آن ارسال نمود.

سایر نکات

- **آدرس مخزن** و **شناسه آخرین تغییر** خود را در محل بارگذاری در سایت درس، بارگذاری نمایید.
- پاسخ تمامی سؤالات را در **کوتاه‌ترین اندازه ممکن** در گزارش خود بیاورید.
- همه افراد باید به پروژه مسلط باشند و نمره تمامی اعضای گروه لزوماً یکسان نخواهد بود.
- در صورت مشاهده هرگونه شباهت بین کدها یا گزارش دو گروه، به هر دو گروه نمره 0 تعلق می‌گیرد.
- فصل ۵ کتاب xv6 می‌تواند مفید باشد.
- هر گونه سؤال در مورد پروژه را فقط از طریق فروم درس مطرح نمایید.

موفق باشید

مراجع

[1] Mohammed A F Al-Husainy. 2007. Best-job-first CPU scheduling algorithm. *Inf. Technol. J.* 6, 2 (2007), 288–293.

[26] CPU Intensive

Retrieved from
https://scialert.net/fulltext/?doi=itj.2007.288.293

[2] Donald H. Pinkston. 2014. Caltech Operating Systems Slides.