



فهرست

- ۱ دستورکار شماره دو: ۱.....
- ۲ اهداف آموزشی: ۱.....
- ۳ توضیح پروژه: ۲.....
- ۱-۳ پیش نیازها و ابزار مورد نیاز: ۲.....
- ۲-۳ ساختار کلی پروژه Pintos: ۲.....
- ۳-۳ مراحل انجام آزمایش: ۲.....
- ۱-۳-۳ فاز اول - آماده سازی و شناخت ساختار: ۲.....
- ۲-۳-۳ فاز دوم - طراحی: ۲.....
- ۳-۳-۳ فاز سوم - ساعت زنگدار (Alarm Clock): ۲.....
- ۴-۳-۳ فاز چهارم - زمان بند اولویت محور (Priority Scheduler): ۳.....
- ۵-۳-۳ فاز پنجم - زمان بند پیشرفته (Advanced Scheduler - MLFQS): ۳.....
- ۶-۳-۳ فاز ششم - تست و ارزیابی: ۳.....
- ۷-۳-۳ جمع بندی: ۳.....
- ۴-۳ نکات: ۴.....
- ۴ سؤالات تحلیلی پایانی: ۴.....

۱ دستورکار شماره دو:

آشنایی با مفاهیم همزمانی، زمانبندی و مدیریت نخها در سطح کرنل

۲ اهداف آموزشی:

- آشنایی با ساختار داخلی Pintos
- اضافه کردن قابلیت توسعه سیستم زمانبندی و همزمانی Pintos.

۳ توضیح پروژه:

این پروژه از چند بخش تشکیل شده است که در ادامه توضیح داده می‌شود.

۱-۳ پیش‌نیازها و ابزار موردنیاز:

- آشنایی با زبان C و ساختارهای داده‌ی لیست و صف
- مفاهیم پایه سیستم‌عامل: نخ (Thread)، زمان‌بند (Schedule)، وقفه (Interrupt).
- محیط نصب‌شده Pintos و شبیه‌ساز QEMU
- ابزارهای کامپایل (make, gcc) و اشکال‌زدایی (gdb)

۲-۳ ساختار کلی پروژه Pintos:

- کد پروژه در مسیر `pintos/src` قرار دارد.
- پوشه `threads` شامل کدهای مربوط به کرنل چندنخی است که در این آزمایش تغییر داده می‌شود.
- پوشه `devices` شامل درایورهای سخت‌افزاری از جمله `timer` است.
- کامپایل با دستور `make` و اجرای تست‌ها با `run test_name -v -- qemu -v -- pintos` انجام می‌شود.

۳-۳ مراحل انجام آزمایش:

۱-۳-۳ فاز اول – آماده‌سازی و شناخت ساختار

- اجرای پروژه‌ی اولیه Pintos
- مرور فایل‌های `thread.c`، `synch.c`، `timer.c`
- بررسی روند `switch_threads` و `schedule` با استفاده از `gdb`
- توجه به محدودیت ۴ KB حافظه پشته هر نخ

۲-۳-۳ فاز دوم – طراحی:

- تکمیل فایل `DESIGNDOC` در پوشه `threads`
- مستندسازی ساختارهای داده، الگوریتم‌ها و روش‌های هم‌زمانی مورد استفاده

۳-۳-۳ فاز سوم – ساعت زنگ‌دار (Alarm Clock):

- بازنویسی تابع `timer_sleep` برای جلوگیری از `busy-wait`
- نگهداری لیستی از نخ‌های در حال خواب و زمان بیداری آنها

- بیدارسازی نخ‌ها در وقفه‌ی تایمر (timer_interrupt)

۳-۳-۴ فاز چهارم – زمان‌بند اولویت‌محور (Priority Scheduler) :

- تعریف بازه‌ی اولویت از PRI_MIN تا PRI_MAX
- پیاده‌سازی preemption هنگام آماده شدن نخ با اولویت بالاتر
- بیدار شدن نخ با بالاترین اولویت هنگام آزاد کردن قفل‌ها یا semaphore.
- افزودن priority donation برای حل مشکل وارونگی اولویت
- اصلاح **قفل‌ها** برای پشتیبانی از donation های تودرتو
- پیاده‌سازی thread_set_priority و thread_get_priority

۳-۳-۵ فاز پنجم – زمان‌بند پیشرفته (Advanced Scheduler - MLFQS) :

- پیاده‌سازی متغیرهای nice، recent_cpu و load_avg
- به‌روزرسانی اولویت‌ها بر اساس فرمول BSD Scheduler
- غیرفعال شدن priority دستی در حالت MLFQS
- استفاده از حساب اعشاری ثابت (Fixed-Point Arithmetic) برای محاسبات

۳-۳-۶ فاز ششم – تست و ارزیابی:

- اجرای تست‌ها با دستور make check
- بررسی نتایج تست‌ها در مسیر tests/threads
- اطمینان از عبور تست‌های alarm، priority و mlfqs

۳-۳-۷ جمع‌بندی:

به طور خلاصه برخی از فایل‌هایی که تغییر می‌یابند در جدول ۱ ذکر شده‌اند.

جدول ۱- برخی از فایل‌هایی که می‌باید تغییر یابند

فایل	نقش
devices/timer.c	بازنویسی timer_sleep
threads/thread.c	اضافه کردن زمان‌بندها و priority
threads/thread.h	تعریف ساختار thread و اولویت‌ها
threads/synch.c	تغییر برای donation
threads/fixed-point.h	پیاده‌سازی محاسبات اعشاری ثابت

همچنین جدول ۲ جمع‌بندی فازها را نشان می‌دهد.

جدول ۲- خلاصه وظایف فازها

فاز	عنوان	فایل‌های کلیدی	خروجی مورد انتظار
۱	آماده‌سازی و شناخت ساختار	thread.c , synch.c	درک ساختار پایه
۲	طراحی	DESIGNDOC	طراحی مستند سیستم
۳	Alarm Clock	timer.c	حذف busy waiting، خواب صحیح نخ‌ها
۴	Priority Scheduler	thread.c , synch.c	زمانبندی بر اساس اولویت و donation
۵	زمانبند پیشرفته (MLFQS)	thread.c, fixed-point.h	پیاده‌سازی زمانبند پویا
۶	تست و ارزیابی	-	انجام تست‌ها و مستندسازی

۳-۴ نکات:

- از تعریف آرایه‌های بزرگ به صورت محلی در پشته خودداری شود.
- تغییرات فقط در فایل‌های threads و devices اعمال شود.
- کد تمیز، دارای توضیح و هماهنگ با سبک Pintos باشد.

۴ سؤالات تحلیلی پایانی:

- (۱) تفاوت بین روش busy-wait و روش block در پیاده‌سازی timer_sleep چیست؟
- (۲) در شرایطی که دو نخ با اولویت متفاوت برای یک قفل رقابت می‌کنند، donation چگونه عمل می‌کند؟
- (۳) در سیستم MLFQS، پارامتر nice چه نقشی در کاهش یا افزایش اولویت دارد؟
- (۴) تفاوت اصلی بین زمان‌بند ساده و MLFQS در پاسخ‌گویی به بار سیستم چیست؟
- (۵) چرا استفاده از fixed-point arithmetic در کرنل نسبت به floating-point ترجیح داده می‌شود؟

تحويل پروژه:

- گزارشکار به همراه سورس کدهای خود را پوشه‌ای با نام osLab_P1_stdID ارسال کنید.
- مهلت ارسال یکشنبه ۱۴۰۴/۰۸/۲۵ ساعت ۱۳ می‌باشد.
- ارائه حضوری یکشنبه ۱۴۰۴/۰۸/۲۵ در زمان برگزاری کلاس خواهد بود.

موفق باشید - آهوز