به نام خداوند بخشنده مهربان

دانشگاه صنعتی شریف
مدرس: مهدی خرازی

تمرین گروهی سوم *
سیستم‌های عامل

دانشکده‌ی مهندسی کامپیوتر
پاییز ۱۳۹۶

آشنایی با فایل‌سیستم

۱. مقدمه

تا کنون سیستم‌عامل شما تا حدودی قدرت‌مند شده‌است و می‌تواند برنامه‌هایی که به‌وسیله‌ی فراخوانی‌های سیستمی[1] Pintos پیاده‌سازی شده‌اند را اجرا کند. شما می‌توانید یک Shell ،کامپایلر، Minesweeper یا هر برنامه‌ی کاربردی دیگری را که بخواهید در آن اجرا کنید. خوشبختانه، تعدادی برنامه‌ی کاربردی در مسیر *src/examples*[2] قرار دارد و نیازی به نوشتن آن‌ها نیست. لیست این برنامه‌ها عبارت‌است از:

- shell
- hex-dump
- rm
- cat
- insult
- mkdir
- cmp
- echo
- cp
- ls
- و...

یکی از بخش‌های مهم که پیاده‌سازی نشده‌است، یک فراخوانی سیستمی برای تخصیص[3] حافظه‌ی بیشتر به Heap و پشته است. به عنوان فعالیت تکمیلی می‌توانید این بخش را هم پیاده‌سازی کنید.

قبل از این که سراغ این تمرین برویم، با نحوه‌ی اجرای این برنامه‌ها آشنا می‌شویم. به این ترتیب می‌توانید به کمک آن‌ها، فایل‌سیستم خود را تست کنید.

---

* این تمرین برگرفته از تمرین ارائه شده در دانشگاه برکلی، مربوط به درس CS162 می‌باشد.

[1] system call
[2] directory
[3] allocate

برای شروع، با اجرای دستور make در مسیر *src/examples* و *src/userprog* تمام این برنامه‌ها را کامپایل کنید و سپس اسکریپت زیر را در مسیر *src/userprog/build* اجرا کنید (این اسکریپت را می‌توانید از اینجا دانلود کنید).

```
PROGRAMS="bubsort \
cat \
cmp \
cp \
echo \-
hexdump \
insult \
ls \
mkdir \
pwd \
rm \
shell"
EXAMPLES="../../examples"
CMDLINE="pintos ---filesyssize=100"
CMDLINE_END="-f -q run 'shell'"
for PROGRAM in $PROGRAMS; do
    CMDLINE+=" -p $EXAMPLES/$PROGRAM -a $PROGRAM"
done
CMDLINE+=" -- $CMDLINE_END"
echo $CMDLINE
eval $CMDLINE
```

run.sh

این اسکریپت باینری برنامه‌های مثال را در root قرار می دهد و Pintos را با shell ارائه شده اجرا می‌کند. بیشتر این برنامه‌ها تا زمانی که این تمرین را انجام ندهید، عملکرد صحیح نخواهند داشت. برخی از آن‌ها، مانند insult یا hex-dump، اگر تمرین گروهی دوم را صحیح پیاده‌سازی کرده‌باشید، به‌درستی کار خواهند کرد. برنامه‌های مختلف را اجرا کنید و سعی کنید بفهمید چه چیزهایی کار می‌کنند و چه چیزهایی کار نمی‌کنند و چرا.

۲. شرح تمرین

در این تمرین، شما باید ۳ قابلیت جدید را به فایل‌سیستم Pintos اضافه کنید. کارهایی که قراراست انجام دهید به‌طور مختصر در زیر شرح داده شده‌است.

نکته: این تمرین به پیاده‌سازی کامل و صحیح از تمرین گروهی دوم نیاز دارد. اگر تمرین دوم شما تمامی تست‌های مربوطه را پاس نمی‌کند، از دستیار های آموزشی درخواست کنید تا کدهای موردنیاز را در اختیار شما قرار دهند.

۱.۲. بخش ۱: Buffer Cache

توابع *inode_read_at()* و *inode_write_at()* در حال حاضر با هربار فراخوانی، به‌طور مستقیم به device block دسترسی پیدا می‌کنند.

شما باید یک cache به فایل‌سیستم اضافه کنید تا کارایی خواندن و نوشتن در دیسک را افزایش دهید. این cache باید بلوک‌های دیسک را به‌گونه‌ای در خود ذخیره کند که ۱) بتواند درخواست‌های خواندن را با داده‌های cache شده پاسخ دهد و ۲) عملیات نوشتن پشت‌سرهم را در یک عمل نوشتن دیسک ترکیب کند.

حداکثر ظرفیت cache شما باید ۶۴ بلوک دیسک باشد و سیاست جایگزینی[۴] آن، باید حداقل به‌خوبی الگوریتم ساعت[۵] باشد. همچنین در هنگام نوشتن باید **write-back** باشد نه write-through. اطمینان حاصل کنید که نه تنها دو تابع inode ذکرشده، بلکه تمام درخواست‌های دیسک، از cache شما عبور کنند.

۲.۲. بخش ۲: فایل‌های گسترش‌پذیر

سیستم‌عامل Pintos در حال حاضر نمی‌تواند حجم فایل‌ها را افزایش دهد، زیرا فایل‌سیستم آن فضای لازم را به‌صورت یک تکه از بلوک‌های پشت‌سرهم حافظه تخصیص می‌دهد. در این بخش، شما باید قابلیت گسترش فایل‌ها را به فایل‌سیستم Pintos اضافه کنید. مانند فایل‌سیستم Unix، این کار را با استفاده از ساختار inode با اشاره‌گرهای مستقیم، غیرمستقیم و غیرمستقیم دومرتبه‌ای[۶] انجام دهید. حداکثر حجم فایلی که باید پشتیبانی کنید، ۸ مگابایت (۲۲۳ بایت) است. در آخر یک فراخوانی سیستمی جدید به شکل `inumber(int fd)` بنویسید که با گرفتن توصیف‌گر فایل[۷]، شماره‌ی inode یکتای مربوط به آن را برگرداند.

۳.۲. بخش ۳: زیرپوشه‌ها

فایل‌سیستم فعلی Pintos پوشه‌ها را پشتیبانی می‌کند ولی برنامه های کاربردی راهی برای استفاده از آنها ندارند و همه‌ی فایل‌ها در root قرار می‌گیرند. شما باید فراخوانی‌های سیستمی chdir، mkdir، readdir و isdir را اضافه کنید تا کاربر قابلیت تغییر و استفاده از پوشه‌ها را داشته باشد. همچنین لازم است فراخوانی‌های سیستمی open، close، exec، remove و inumber را تغییر دهید تا با پوشه‌ها نیز کار کنند. شما باید از مسیرهای نسبی و مطلق در آدرس‌دادن فایل‌ها در فراخوانی های سیستمی پشتیبانی کنید. همچنین علائم "." و ".." که در آدرس دادن به کار می‌روند باید به درستی تعبیر شوند. در صورتی که آدرس با "/" یا "." شروع نشود فایل باید نسبت به مسیر جاری[۸] پردازه یافت شود. مثال‌هایی از فراخوانی‌های مجاز عبارت‌اند از:

[4] replacement policy
[5] clock algorithm
[6] doubly indirect
[7] file descriptor
[8] current working directory

`open("/my_files/notes.txt")` `open("my_files/notes.txt")`
`open("../logs/foo.txt")`

پردازه‌های فرزند، پوشه‌ی فعلی‌شان را از والد خود به ارث می‌برند. پوشه‌ی فعلی اولین پردازه، root است.

۴.۲. نیاز به همگام‌سازی (Synchronization)

کدهای تمرین شما، باید thread-safe باشند. اما در این تمرین نباید از یک قفل کلی برای فایل‌سیستم استفاده کنید. اطمینان حاصل کنید که دو عملیات که بر روی sector های متفاوت دیسک، فایل‌های متفاوت و پوشه‌های متفاوت کار می‌کنند، می‌توانند به‌طور هم‌زمان و موازی اجرا شوند. عملیاتی که روی sector های یکسان دیسک، فایل‌های یکسان یا پوشه‌های یکسان کار می‌کنند، می‌توانند متوالی اجرا شوند. برای مثال دو فراخوانی `remove("/my_files/notes.txt")` و `readdir(open("/my_files/"))` یک پوشه‌ی مشترک را در عملیات خود دخیل می‌کنند و اگر موازی اجرا شوند ممکن است، باعث انحصار متقابل[9] شوند. اما دو فراخوانی `read(open("/my_files/notes.txt"))` و `write(open("/my_files/test.c"))` بر روی دو فایل متفاوت عمل می‌کنند و داشتن پوشه‌ی مشترک در اینجا بی‌تاثیر است. بنابراین این دو می‌توانند موازی انجام شوند. ضرورتی که توضیح داده شد در پیاده‌سازی هر ۳ ویژگی باید رعایت شود.

نکته: اگر در تمرین گروهی دوم، یک قفل سراسری به فایل‌سیستم اضافه کرده‌اید، به یاد داشته‌باشید که آن را حذف کنید.

۳. تحویل‌دادنی‌ها

نمره‌ی تمرین شما شامل ۴ بخش است:

- ٪۲۰ مستند طراحی و بازخورد طراحی
- ٪۵۵ کد و پیاده‌سازی
- ٪۱۵ کد تست‌های دانشجویان
- ٪۱۰ گزارش نهایی و کیفیت کد نوشته‌شده

۱.۳. مستند طراحی و بازخورد طراحی

قبل از شروع به پیاده‌سازی، شما باید مستندی شامل مراحل پیاده‌سازی برای هریک از ویژگی‌های یادشده را آماده کنید و مطمئن شوید که طراحی شما مناسب و صحیح است. برای این تمرین، شما باید یک مستند طراحی را تحویل داده و از دستیار آموزشی اختصاص‌داده‌شده به شما بازخورد مناسب بگیرید.

[9] mutual exclusion

۱.۱.۳. دستورالعمل مستند طراحی

مستند طراحی را در فایل doc/project3.md که قبلا در مخزن گیت گروه شما ساخته شده است، بنویسید. شما باید از فرمت Markdown برای مستند طراحی خود استفاده کنید. می‌توانید پیش‌نمایش مستند طراحی خود را روی رابط وب گیت، با رفتن به آدرس زیر مشاهده کنید: (بخش group0 را با شماره گروه خود جای‌گذاری کنید.)

https://tarasht.ce.sharif.ir/ce424-961-groups/ce424-961-group1/src/master/doc/project3.md

برای هر یک از بخش‌های این تمرین، باید ۴ جنبه‌ی زیر از طراحی‌تان را توضیح دهید.

۱. **داده‌ساختارها و توابع** ـ تعریف تمامی struct ها، متغیرهای سراسری و ثابت، typedef ها یا enum هایی که اضافه می‌کنید و یا تغییر می‌دهید را بنویسید. تمامی تعاریف باید به **زبان C** و نه به صورت سودوکد نوشته شده باشد. هم‌چنین **به صورت خلاصه** درباره‌ی دلیل تغییر هر بخش توضیح دهید. توضیحات شما باید تا حد امکان خلاصه باشند. توضیحات ریز و دقیق خود را در بخش‌های دیگر بنویسید.

۲. **الگوریتم‌ها** ـ در این بخش شما باید توضیح دقیقی از نحوه‌ی کارکردن کد خود ارائه کنید. توضیحات شما باید با جزئیات بیشتری نسبت به صورت تمرین باشد (از بازنویسی توضیحات ارائه‌شده در صورت تمرین خودداری کنید.) هم‌چنین باید گویاتر از کدتان باشد (توضیح خط‌به‌خط درمورد کد نیز لازم نیست.) در نوشته‌ی خود سعی کنید تا ما را قانع کنید که کد شما تمامی خواسته‌های صورت تمرین **(حتی حالت‌های خاص نامتعارف)** را برآورده می‌کند. از شما انتظار می‌رود هنگام نوشتن مستند طراحی، سورس کد Pintos را بخوانید و در صورت نیاز به آن ارجاع دهید.

۳. **همگام‌سازی** ـ در این قسمت باید تمام منابعی که بین ریسه‌ها به اشتراک گذاشته می‌شوند را لیست کنید و برای هر مورد بررسی کنید که دسترسی به آنها چگونه صورت می‌گیرد. (برای مثال از داخل زمان‌بند یا هنگام رسیدگی به وقفه یا ...) سپس روش خود را برای اطمینان از اشتراک‌گذاری صحیح و ایمن این منابع توضیح دهید. برای هر منبع اثبات کنید که روش شما رفتار درست را تضمین می‌کند. به طور کلی بهترین استراتژی‌های همگام‌سازی ساده هستند و به راحتی می‌توان درستی‌شان را تحقیق کرد. در صورتی‌که توضیح استراتژی همگام‌سازی شما سخت است، نمایانگر این است که باید استراتژی‌تان را ساده‌تر کنید.
همچنین درباره‌ی موارد زیر توضیح دهید:

- هزینه‌ی زمانی و حافظه‌ای روش همگام‌سازی‌تان
- آیا با این روش شما، موازی‌سازی هسته به طرز قابل‌توجهی محدود می‌شود یا خیر؟ برای تشریح قابلیت توازی روش‌تان، توضیح‌دهید که ریسه‌ها با چه نرخ تکراری برای به‌دست آوردن منابع مشترک با هم رقابت می‌کنند. هم‌چنین محدودیت تعداد ریسه‌هایی که می‌توانند به ناحیه‌های بحرانی[۱۰] مستقل از هم وارد شوند را بیان کنید.

[10] critical section

۴. **منطق طراحی** ـ توضیح دهید چرا طراحی شما از دیگر روش‌هایی که بررسی کردید بهتر است و کاستی های آن را شرح دهید. در توضیحات خود به نکات زیر توجه داشته باشید:

- طراحی شما تا چه اندازه قابل درک است؟
- برنامه‌نویسی آن چقدر زمان بر است؟
- پیچیدگی الگوریتم‌های شما از نظر زمانی و حافظه چقدر است؟
- چقدر راحت می‌توان این طراحی را تغییر داد تا ویژگی‌های بیشتری در آن قرار گیرد؟

۲.۱.۳. موضوعات مستند طراحی

هر یک از مسائل زیر را در بخش الگوریتم‌ها و همگام‌سازی مستند طراحی‌تان بررسی کنید. نیازی نیست که این سوالات را به‌طور مستقیم پاسخ دهید، ولی مستند طراحی شما باید نشان دهد که طراحی شما هیچ‌یک از این مشکلات را ندارد.

- هنگامی که یک پردازه در حال خواندن یا نوشتن یک بلوک cache است، چگونه آن بلوک توسط پردازه‌های دیگر بیرون انداخته نمی‌شود؟
- هنگام بیرون انداختن یک بلوک از cache، چگونه از دسترسی پردازه‌های دیگر به آن بلوک جلوگیری می‌شود؟
- هنگام بارگذاری یک بلوک در cache، چگونه از بارگذاری همان بلوک در خانه‌ای دیگر از cache توسط بقیه‌ی پردازه‌ها، جلوگیری می‌شود؟
- چگونه فایل‌سیستم شما مسیرهای نسبی مانند *../my_files/notes.txt* را می‌گیرد و مسیر موردنظر را پیدا می‌کند؟ مسیرهای مطلق مانند */cs162/solutions.md* چگونه پیدا می‌شوند؟
- آیا یک پردازه می‌تواند پوشه‌ای که مسیر فعلی[11] یک پردازه است را حذف کند؟ تست‌ها هر دو جواب «بله» و «خیر» را می‌پذیرند، ولی در هر صورت باید اطمینان حاصل کنید که در پوشه‌ی پاک شده، امکان ایجاد فایل جدید وجود نداشته باشد.
- چگونه syscall handler های شما، از روی توصیف‌گر های فایل، پوشه یا فایل موردنظر را پیدا می‌کنند؟
- شما از پیش با اشباع حافظه در C، که با برگرداندن NULL از *malloc* محرض می‌شود، آشنا هستید. در این تمرین، شما باید اشباع دیسک را نیز بررسی کنید. هنگامی که فایل‌سیستم شما توانایی تخصیص بلوک‌های جدید را در دیسک نداشته باشد، شما باید یک استراتژی برای لغوکردن عملیات فعلی و برگشتن به آخرین وضعیت مناسب داشته باشید.

[11] current working directory

۳.۱.۳. سوالاتی دیگر برای مستند طراحی

سوال زیر را نیز در مستند طراحی خود پاسخ دهید:

۱. در قسمت cache این تمرین دو ویژگی اضافی را می‌توانید به‌طور اختیاری پیاده‌سازی کنید:

- **write-behind** ـ با این ویژگی cache شما به‌صورت متناوب بلوک های تغییریافته را در block device فایل‌سیستم می‌نویسد. بنابراین با قطع برق یا خاموش‌شدن ناگهانی سیستم مقدار زیادی از داده‌ها از دست نمی‌رود. بدون این قابلیت، یک cache از نوع write back تنها هنگامی داده‌ها را در دیسک می‌نویسد که ۱) داده تغییر داده شده باشد (بیت dirty آن ۱ باشد.) و در حال بیرون‌شدن از cache باشد یا ۲) سیستم در حال خاموش‌شدن باشد.
- **read-ahead** ـ با این ویژگی cache پیش‌بینی می‌کند چه بلوک‌هایی از داده احتمالا در آینده‌ی نزدیک موردنیاز هستند و در پس‌زمینه، آن‌ها را واکشی[۱۲] می کند. این قابلیت، در خواندن‌های متوالی از فایل یا الگوهای ساده‌ی دست‌یابی به فایل، کارایی را افزایش می‌دهد.

شما باید یک طراحی قابل پیاده‌سازی برای هر دو ویژگی بالا ارائه دهید. توجه داشته باشید که حتی اگر تصمیم بگیرید این بخش را پیاده‌سازی نکنید نیز، توضیحات مربوط به آن باید در مستند طراحی شما وجود داشته باشد.

۴.۱.۳. بازخورد طراحی

شما در یک جلسه‌ی ۲۰ ـ ۲۵ دقیقه ای، طراحی خود را به دستیارآموزشی ارائه میدهید. در آن جلسه باید آماده باشید تا به سوالات پاسخ دهید و از طراحی خود دفاع کنید.

۵.۱.۳. نمره‌دهی

مستند طراحی و بازخورد طراحی با هم نمره دهی میشوند. این بخش ۲۰ نمره دارد که بر اساس توضیحات شما از طراحیتان در مستند و پاسخ‌دهی شما به سوالات در جلسه‌ی بازخورد، نمره دهی میشود. باید حتما در جلسه‌ی بازخورد طراحی حضور داشته باشید تا نمره ای به شما تعلق بگیرد.

۲.۳. پیاده‌سازی

Pintos یک مجموعه تست دارد که می‌توانید با دستور `"make check"` در مسیر *pintos/src/filesys* آن‌ها را اجرا کنید. نمره‌ی پیاده‌سازی شما توسط نمره‌دهنده‌ی خودکار داده می‌شود. این نمره در فایلی به نام grade.txt در پوشه‌ی مربوط به هر تمرین قرار می گیرد.

[12] fetch

۱.۲.۳. کد تست دانشجویان

تست هایی که برای تمرین ۳ در Pintos قرار دارد قسمت cache را پوشش نمی‌دهد. برای این قسمت **دو مورد** از تست های زیر را پیاده‌سازی کنید:

- تاثیر cache را با اندازه گیری نرخ برخورد[۱۳] بیازمایید. ابتدا cache را راه‌اندازی مجدد کنید، سپس یک فایل را باز کنید و محتوای آن را پشت سر هم بخوانید. سپس فایل را ببندید و دوباره باز کنید و باز هم محتوای آن را پشت سر هم بخوانید.در سری دوم نرخ برخورد باید افزایش یافته باشد.

- قابلیت ترکیب‌کردن چندین عملیات نوشتن بر روی یک sector یکسان را بیازمایید. هر block device دو شمارنده‌ی *read_cnt* و *write_cnt* درون خود نگه می‌دارد. یک فایل بزرگ را بایت به بایت بنویسید. (سایز آن حداقل ۶۴ کیلوبایت یعنی دو برابر حداکثر ظرفیت cache شود.) سپس بایت به بایت از آن بخوانید. تعداد کل نوشتن ها در block_device باید در حدود ۱۲۸ باشد. (۶۴ کیلوبایت معادل ۱۲۸ بلوک است.)

- بررسی کنید که cache قابلیت نوشتن یک بلوک کامل را بدون نیاز به خواندن آن دارد. اگر برای مثال ۱۰۰ کیلوبایت (۲۰۰ بلوک) را درون فایل می‌نویسید، cache باید ۲۰۰ بار *block_write* و ۰ بار *block_read* را فراخوانی کند. البته عملیات خواندن برای فرادادهی[۱۴] inode ها مجاز است. همان‌طور که در بخش قبل گفته شد، تعداد عملیات خواندن و نوشتن در شمارنده‌های *read_cnt* و *write_cnt* وجود دارد و با استفاده از آن‌ها می‌توانید از وجود نداشتن عملیات خواندن اضافی اطمینان حاصل کنید.

در پیاده سازی کد تست خود به نکات زیر توجه کنید:

- تمرکز را بر روی تست ویژگی‌های کلی بگذارید نه قابلیت های خاص cache خودتان و حداقل پیش‌فرض‌ها را در مورد نحوه‌ی پیاده‌سازی cache درنظر بگیرید.

- تست خود را به گونه‌ای بنویسید که بدون نیاز به تغییر، بتواند بر روی پروژه‌ی سایر گروه‌ها اجرا شود.

- تست‌های شما باید با دستور `"make check"` در مسیر *pintos/src/filesys* اجرا شود.

۳.۳. گزارش نهایی و کیفیت کد

بعد از تمام شدن کد تمرین، باید گزارش نهایی خود را ارسال کنید. این گزارش را در *reports/project3.md* در مخزن گیت مربوط به گروه خود بنویسید. گزارش نهایی شما باید شامل این بخش ها باشد:

- تغییراتی که نسبت به مستند طراحی اولیه‌ی خود داده‌اید، و توضیح این‌که چرا این تغییرات را داده‌اید. می‌توانید مباحثی که در جلسه با دستیاران آموزشی بیان شدند را هم اضافه کنید.

[13] hit rate
[14] metadata

- تقسیم وظایف گروه، در بخش‌های مختلف پروژه. هر فرد چه کارهایی را انجام داده‌است؟ چه بخش‌هایی از پروژه به‌خوبی پیش رفت، و کدام بخش‌ها می‌توانست بهبود یابد؟
- کد تست دانشجویی شما (برای اطلاعات بیشتر بخش‌های قبل را مشاهده کنید.)

نمره‌ی شما بسته به کیفیت کدی است که پیاده‌سازی کرده‌اید و این مساله به موارد مختلفی بستگی دارد:

- مدارکی که نشان دهند کد شما فارق از مشکلات امنیتی حافظه (مخصوصا مشکلات مربوط به رشته در C)، نشتی حافظه، رسیدگی ضعیف به ارور یا race condition است.
- نوشتن کد به صورت یکپارچه. کد شما باید با کد قبلی Pintos ترکیب شود. تورفتگی‌ها، فاصله‌بندی و استانداردهای نام‌گذاری خود را بررسی کنید.
- ساده و خوانابودن کد
- گذاشتن comment در بخش‌های پیچیده‌ی کد
- نبودن کد comment شده در ارسال نهایی
- نبودن کد کپی شده و وجود توابع مناسب
- اینکه الگوریتم‌های مربوط به لیست پیوندی را دوباره پیاده‌سازی کرده‌اید یا اینکه از الگوریتم موجود استفاده کرده‌اید.
- طولانی نبودن بیش از اندازه‌ی خط‌های کد (بیش از ۱۰۰ کاراکتر)
- درست بودن commit ها در گیت. (object file ها و log ها را commit نکنید مگر اینکه واقعا لازم باشند)

۱.۳.۳. گزارش تست‌های دانشجویان

باید گزارشی برای تست‌هایی که نوشته‌اید تهیه کنید تا برای نمره‌دهی استفاده شود. گزارش را در reports/project3.md در کنار گزارش نهایی خود قرار دهید. مطمئن شوید موارد زیر در گزارش شما آمده است:

- برای هر یک از دو تست موارد زیر را بنویسید:
  - ویژگی یا خواسته‌ای که توسط تست پوشش داده شده است
  - نجوه‌ی کارکرد تست شما و خروجی مورد انتظار
  - خروجی کرنل Pintos وقتی که تست‌های شما اجرا می‌شود. بدین منظور محتوای دو فایل زیر را کپی کنید:

filesys/build/tests/filesys/extended/your-test-1.output

filesys/build/tests/filesys/extended/your-test-1.result

- برای هر تست، دو باگ احتمالی در کرنل را در نظر بگیرید و بگویید که چگونه می‌تواند روی تست شما تاثیر بگذارد. پاسخ شما می‌تواند به فرم زیر باشد:
  «اگر در کرنل اتفاق x به جای y رخ دهد، تست نتیجه z را می‌دهد.»

- تجربه شما هنگام نوشتن تست برای Pintos چگونه بود؟ چه چیزهایی در سیستم تست Pintos می‌تواند بهبود پیدا کند؟ هنگام نوشتن تست‌ها چه چیزهایی یاد گرفتید؟

نمره‌دهی به این بخش بر اساس کامل‌بودن و درست‌بودن هریک از موارد خواسته‌شده است. اگر تمامی موارد بالا در گزارش شما موجود باشند و تست‌های قابل‌قبولی نوشته باشید، از این بخش نمره‌ی کامل دریافت می‌کنید.