به نام خداوند بخشنده مهربان

دانشگاه صنعتی شریف
مدرس: مهدی خرازی

تمرین گروهی دوم *
سیستم‌های عامل

دانشکده‌ی مهندسی کامپیوتر
پاییز ۱۳۹۷

۱. مقدمه

در این پروژه، باید به گسترش Pintos با هدف پشتیبانی از برنامه‌های کاربر بپردازید. شالوده کد حال حاضر این سیستم‌عامل قادر است تا برنامه‌های کاربر را بارگذاری و شروع نماید اما برنامه‌ها قادر به خواندن آرگومان‌های خط فرمان (command-line arguments) و یا ساختن فراخوان‌های سیستمی نیستند.

۱.۱ پاس دادن آرگومان

تابع `process_execute(char *file_name)` برای ساختن یک پردازه سطح کاربر جدید در Pintos استفاده می‌شود اما در حال حاضر از آرگومان‌های خط فرمان پشتیبانی نمی‌کند. شما باید قابلیت پاس دادن آرگومان را پیاده‌سازی کرده به طوری که فراخوانی تابع `process_execute("ls -ahl")` دو آرگومان `["ls", "-ahl"]` را برای برنامه سطح کاربر با استفاده از `argc` و `argv` فراهم آورد. تمام برنامه‌های آزمایشی ما در Pintos با چاپ کردن اسمشان (برای مثال `argv[0]`) شروع می‌شوند. از آنجا که پاس دادن آرگومان هنوز پیاده‌سازی نشده است، همه‌ی این برنامه‌ها هنگام تلاش برای دسترسی به `argv[0]` متوقف شده و به اصطلاح crash می‌کنند. تا زمانی که شما قابلیت پاس دادن آرگومان را پیاده سازی نکنید، هیچ کدام از برنامه‌های سطح کاربر قادر به اجرا نخواهند بود.

* این تمرین برگرفته از تمرین ارائه شده در دانشگاه برکلی، مربوط به درسِ CS۱۶۲ می‌باشد.
با تشکر از تیم دستیاران آموزشی

۲.۱. فراخوانی سیستمی کنترل پردازه

درحال حاضر Pintos تنها از فراخوانی exit پشتیبانی می‌کند و شما باید پشتیبانی از فراخوانی‌های سیستمی زیر را به آن اضافه کنید: halt, exec, wait, practice

هر کدام از این فراخوانی‌های سیستمی یک تابع متناظر در داخل کتابخانه سطح کاربر (می‌توانید در `lib/user/syscall.c` به آن دسترسی داشته باشید) دارد که آماده کردن آرگومان‌های فراخوانی سیستمی و مدیریت انتقال آن به سطح کرنل را انجام می‌دهد. رسیدگی به فراخوانی‌های سطح کرنل از طریق توابع موجود در `userprog/syscall.c` صورت می‌گیرد. فراخوانی سیستمی halt وظیفه خاموش کردن سیستم را بر عهده دارد. فراخوانی سیستمی exec اقدام به آغاز یک برنامه جدید با process_execute() می‌کند (در Pintos فراخوانی سیستمی fork وجود ندارد. فراخوانی سیستمی exec شبیه صدا زدن فراخوانی سیستمی fork در لینوکس و سپس بلافاصله صدا زدن فراخوان سیستمی execve در پردازه فرزند می‌باشد) فراخوانی سیستمی wait منتظر می‌ماند تا یک پردازه فرزند خاص با دستور exit به اتمام برسد. در نهایت فراخوان سیستمی practice فقط یک واحد به آرگومان اولش اضافه کرده و نتیجه را برمی‌گرداند (هدف از این فراخوانی سیستمی، آشنایی شما با نوشتن یک برنامه مدیریت فراخوانی سیستمی است).

برای پیاده‌سازی فراخوانی‌های سیستمی، ابتدا به یک راه امن و مطمئن برای خواندن از و نوشتن در حافظه موجود در فضای آدرس مجازی پردازه کاربر نیاز دارید. آرگومان‌های فراخوانی سیستمی در پشته پردازه کاربر قرار گرفته‌اند که این پشته دقیقا بالای `stack pointer` پردازه است. اگر هنگامی که برنامه کاربر یک فراخوانی سیستمی را صدا می‌زند مقدار `stack pointer` نامعتبر باشد، کرنل نمی‌تواند هنگام خواندن محتوای نشانگر نادرست و یا null به اصطلاح crash کند. به علاوه، بعضی از آرگومان‌های فراخوانی‌های سیستمی، نشانگر به بافرهای درون فضای آدرسی پردازه کاربر هستند که این نشانگرها نیز می‌توانند نامعتبر باشند.

شما باید مواردی که یک فراخوانی سیستمی، به دلیل خطا در دسترسی به حافظه، نمی‌تواند به طور کامل اجرا شود را مدیریت کنید. این گونه از خطاهای حافظه شامل نشانگرهای null، نشانگرهای نامعتبر (اشاره به قسمتی از حافظه می‌کنند که مقداردهی معتبر نشده‌اند) و یا نشانگرهایی که به فضای آدرس مجازی سطح کرنل اشاره می‌کنند، است. به عنوان نکته بدانید ممکن است که یک بخش ۴ بایتی حافظه (مثلا یک int ۳۲ بیتی) از ۲ بایت معتبر و ۲ بایت نامعتبر تشکیل شده باشد اگر آن بخش در مرز صفحه‌ها قرار گرفته باشد. شما باید این گونه موارد را با terminate کردن پردازه کاربر مدیریت کنید. توصیه می‌شود این بخش از کد خود را، پیش از پیاده‌سازی سایر توابع فراخوانی سیستمی خود آزمایش نمایید. برای اطلاعات بیشتر می‌توانید به بخش ۳.۱.۷ Accessing User Memory در مستند اصلی Pintos مراجعه کنید.

۳.۱. فراخوان سیستمی عملیات مربوط به فایل

علاوه بر فراخوانی‌های سیستمی کنترل پردازه، نیاز است تا فراخوانی‌های سیستمی مورد نیاز برای عملیات مربوط به فایل‌ها، که در ادامه آمده است را پیاده‌سازی کنید:

create, remove, close, tell, seek, write, read, filesize, open

سیستم‌عامل Pintos به طور پیش‌فرض دارای یک filesystem بسیار ساده است. پیاده‌سازی این فراخوانی‌های سیستمی توسط شما، به سادگی توابع مناسب را در کتابخانه filesystem صدا میزند و نیازی نیست که هیچ یک از این عملیات‌های فایل را خودتان پیاده‌سازی کنید.

filesystem پیش‌فرض Pintos یک فایل‌سیستم thread-safe نیست. شما باید اطمینان حاصل کنید که که فراخوانی‌های سیستمی مربوط به عملیات فایل‌ها، چندین توابع پیش‌فرض موجود در فایل‌سیستم Pintos را به صورت همزمان فراخوانی نکند. در تمرین گروهی بعدی، قابلیت همزمانی را به فایل‌سیستم Pintos اضافه خواهید کرد اما برای این تمرین، این اجازه داده می‌شود تا از قابلیت global lock بر روی عملیات‌های مربوط به فایل‌سیستم، برای اطمینان از thread-safe بودن فایل‌سیستم استفاده کنید. توصیه می‌شود از تغییر دایرکتوری `filesys` در این تمرین گروهی اجتناب کنید.

هنگامی که یک پردازه کاربر در حال اجرا است، باید اطمینان حاصل کنید که هیچ کس نمی‌تواند نسخه قابل اجرا آن را بر روی دیسک تغییر دهد. تست `"rox"` مطمئن می‌شود که شما از نوشتن بر روی فایل اجرایی پردازه‌ای که هم‌اکنون در حال اجرا است، جلوگیری می‌کنید. توابع `file_deny_write()` و `file_allow_write()` می‌توانند به شما برای انجام این وظیفه کمک کنند.

نکته: کد نهایی شما برای این تمرین گروهی، به عنوان نقطه شروع برای تمرین گروهی بعدی مورد استفاده قرار خواهد گرفت. تست‌های تمرین گروهی بعد وابسته به برخی از فراخوانی‌های سیستمی پیاده‌سازی شده در این تمرین گروهی هستند. لذا هنگام طراحی و پیاده‌سازی موارد مربوط به این تمرین گروهی، این نکته را نیز در ذهن داشته باشید.

۲. تحویل دادنی‌ها

نمره‌ی تمرین شما شامل ۴ بخش است:

- ٪۲۰ مستند طراحی و بازخورد طراحی
- ٪۵۵ کد و پیاده‌سازی
- ٪۱۵ تست های دانشجو
- ٪۱۰ گزارش نهایی و کیفیت کد نوشته شده

۱.۲. مستند طراحی و بازخورد طراحی

قبل از شروع به پیاده‌سازی، شما باید مستندی شامل مراحل پیاده‌سازی برای هر یک از ویژگی‌های یاد شده 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 اختصاص داده شده به شما بازخورد مناست بگیرید.

۲.۲. دستورالعمل مستند طراحی

مستند طراحی را در فایل doc/ghw2.md که قبلا در مخزن گیت گروه شما ساخته شده بنویسید. شما باید از مارکدان برای فرمت طراحی مستند طراحی خود استفاده کنید. شما می‌توانید پیش‌نمایش مستند طراحی خود را روی واسط وب گیت با رفتن به آدرس زیر مشاهده کنید: (بخش group1 را با شماره گروه خود جایگزاری کنید.)

https://tarasht.ce.sharif.ir/ce424-971-groups/ce424-971-group1/src/master/doc

برای هر یک از بخش‌های این تمرین شما باید چهار جنبه‌ی زیر از تمرینتان را با توجه به مستند طراحیتان توضیح دهید. توصیه‌ی ما این است که یک بخش برای هر یک از ۳ بخش تمرین ایجاد کرده و سپس زیربخش‌هایی را برای جنبه‌های مختلف تمرین خود بسازید.

۱. ساختار داده ها و توابع: تعریف ساختار داده های استفاده شده، متغیر های global و typedef ، static ها enumerations ها و ...که اضافه کرده و یا تغییر داده اید را باید در این بخش مستند کنید. این تعاریف باید با زبان برنامه نویسی C نوشته شوند و نه pseudocode و شامل یک توضیح مختصر از هدف تعریف و یا تغییر هر مورد باشند. توضیحات شما باید تا حد ممکن مختصر باشند. توضیحات کامل را باید در بخش بعدی مستند کنید.

۲. الگوریتم‌ها: در اینجا شما باید نحوه کار کردن کد خود را توضیح دهید. توضیحات شما باید در یک سطح پایین‌تری (low-level) از توضیحات کلی ارائه شده درباره نیازمندی ها، که در صورت پروژه آمده است باشد. لطفا از ذکر مشخصات پروژه که در صورت پروژه ذکر شده اند یا آوردن جملات این بخش با جمله‌بندی جدید به عنوان بخشی از توضیحات خود، خودداری نمائید. همچنین، از توضیح خط به خط کدی که قصد دارید بنویسید خودداری نمائید، به جای آن باید دستیار آموزشی پروژه را متقاعد کنید که طراحی شما، تمام نیازمندی های خواسته شده را برآورده می‌کند. این بخش از نظر فرمت و قالب باید شبیه به داکیومنت طراحی باشد که در تمرین گروهی قبل سابمیت کرده‌اید. همچنین انتظار میرود هنگام طراحی خود منابع مربوط به Pintos و سورس کد آن را مطالعه نموده و هر جا که در پیاده‌سازی خود از قسمتی از سورس کد استفاده نمودید، به طور دقیق به آن قسمت ارجاع دهید.

۳. همگام‌سازی: در این قسمت، باید همه منابعی که بین ریسه‌های مختلف به اشتراک گذاشته شده‌اند را لیست کنید. برای هر نمونه، باید ذکر کنید که دسترسی به منابع چگونه صورت می‌گیرد (برای مثال از درون scheduler ، از درون یک interrupt context و غیره). همچنین باید توضیح دهید که استراتژی شما برای اطمینان از اینکه این منابع به شکل امن اشتراک‌گذاری و تغییر می‌یابند چیست. برای هر نوع منبع، نشان دهید که طراحی شما از رفتار صحیح و جلوگیری از وقوع deadlock اطمینان حاصل می‌کند. به طور کلی، بهترین راهکارهای همگام‌سازی، ساده و به راحتی قابل اثبات هستند. اگر توضیح راهکار همگام‌سازی شما

دشوار باشد، به این معنی است که شما باید راهکار خود را ساده تر کنید. لطفا در مورد هزینه های زمان/حافظه مربوط به روش همگام‌سازی خود و اینکه آیا راهکار شما به طور موثر parallelism در کرنل را محدود می‌کند یا خیر، توضیح دهید. در مورد روش و طراحی پیشنهادی شما برای parallelism توضیح دهید و بنویسید که ریسه‌ها هر چند مدت یکبار بر روی منابع اشتراکی رقابت می‌کنند و همچنین محدودیت‌های موجود بر روی تعداد ریسه‌هایی که می‌توانند وارد نواحی بحرانی (critical-section) مستقل در یک لحظه شوند را بیان کنید.

۴. منطق: توضیح دهید چرا طراحی شما از آنچه که به عنوان جایگزین‌ها در نظر گرفته بودید بهتر است.، یا هر نوع کاستی که ممکن است طراحی شما داشته باشد را بیان کنید. شما باید درباره این فکر کنید که آیا طراحی شما می‌تواند به راحتی مفهوم سازی (conceptualize) گردد یا خیر، چه مقدار به کد زدن نیاز دارد، پیچیدگی‌های حافظه و زمان مربوط به الگوریتم‌های شما چقدر است و چقدر آسان یا دشوار خواهد بود تا طراحی شما برای جا دادن ویژگی اضافه توسعه داده شود.

۳.۲. سوالات افزوده برای سند طراحی

.علاوه به بر بخش‌های قبل، در سند طراحی خود باید پاسخ این سوالات را نیز بدهید

۱. وارد `test suit` پروژه در آدرس `pintos/src/tests/userprog` شوید. برخی از تست‌ها اشاره‌گرهای نامعتبری به عنوان آرگومان فراخوانی سیستمی دارند تا پیاده‌سازی شما را در مدیریت کردن فرآیند خواندن و نوشتن حافظه برنامه کاربر بسنجند. تستی را که هنگام اجرای فراخوانی سیستمی از یک `stack pointer ($esp)` نامعتبر استفاده کرده است بیابید. پاسخ شما باید دقیق بوده و نام تست و چگونگی کارکرد آن و همچنین شماره خطوط و نام متغیرهای تست را شامل شود.

۲. تستی را که هنگام اجرای فراخوانی سیستمی از یک `stack pointer` معتبر استفاده کرده ولی `stack pointer` آن‌قدر به مرز `page` نزدیک است که برخی از آرگومان‌های فراخوانی سیستمی در جای نامعتبر حافظه قرار گرفته‌اند، مشخص کنید. دقت کنید که پیاده‌سازی شما در این حالت باید برنامه سطح کاربر را ببندد. پاسخ شما باید دقیق بوده و نام تست و چگونگی کارکرد آن و همچنین شماره خطوط و نام متغیرهای تست را شامل شود.

۳. یک قسمت از خواسته‌های تمرین را که توسط `test suit` موجود تست نشده است، نام ببرید. سپس مشخص کنید تستی که این خواسته را پوشش بدهد چگونه باید باشد.

۴.۲. سوالات GDB

برای موفقیت در کار با Pintos آشنایی با GDB بسیار مفید است. بدین منظور سوالاتی از Pintos می‌پرسیم و انتظار داریم هر فرد در گروه نیز این قسمت را انجام دهد.

ابتدا وارد `src/userprog` شده و دستور `make check` را وارد کنید تا تمرین build شود. سپس دستور زیر را وارد کنید:

pintos –gdb –filesys-size=2 -p tests/userprog/args-none -a args-none – -q -f -v run args-none

سپس با دستور `pintos-gdb build/kernel.o` ، GDB را شروع کرده و آن را به پردازه Pintos متصل کنید (debugpoint) در صورت ابهام در این دستورات به بخش مربوط به GDB در تمرین یک مراجعه کنید.

سوالات:

۱. یک break point در `process_execute` ایجاد کرده و تا آن Point ادامه دهید. نام و آدرس ریسه‌ای که این تابع را اجرا می‌کند چیست؟ ریسه‌های فعال که اکنون در Pintos وجود دارند را بنویسید. (از دستور `dumplist &all_list thread allelem` استفاده کنید)

۲. backtrace ریسه جاری را بنویسید. می‌توانید نتیجه GDB را کپی کنید. همچنین خط‌های برنامه C مربوط به هر فراخوانی تابعی را بنویسید.

۳. یک break point در `start_process` ایجاد کرده و تا آن point ادامه دهید. نام و آدرس ریسه‌ای که این تابع را اجرا می‌کنند بنویسید. چه ریسه‌هایی اکنون در Pintos وجود دارند؟ `struct thread` مربوط به آن‌ها را بنویسید.

۴. ریسه‌ای که `start_process` را اجرا می‌کند کجا ساخته شده است؟ خط کد مربوط را بنویسید.

۵. دوباره continue را وارد کنید. برنامه کاربر باید یک page fault ایجاد کند که باعث می‌شود page fault handler اجرا شود. بدین صورت:

```
[Thread <main>] 1 stopped.
pintos-debug: a page fault exception occurred in user mode
pintos-debug: hit 'c' to continue, or 's' to step to intr_handler
0xc0021ab7 in intr0e_stub ()
```

خطی از کد برنامه کاربر که باعث page fault شده‌است را بیابید. دقت کنید که ممکن است این خط صرفا یک کد hex باشد. (از دستور `btpagefault` استفاده کنید)

۶. دلیل اینکه دستور `b tpagefault` یک آدرس hex برمی‌گرداند این است که دستور `pintos-gdb build/kernel.o` فقط نمادها (symbol) را از کرنل بارگذاری می‌کند. دستوری که باعث page fault شد در برنامه کاربر موجود است بنابرین باید نمادهای زیر را در GDB بارگذاری کنیم:

```
loadusersymbols build/tests/userprog/args-none
```

سپس دستور `btpagefault` را دوباره وارد کنید و نتیجه را بنویسید.

۷. چرا برنامه کاربر در این خط باعث page fault شد؟

۵.۲ بازخورد طراحی

شما در یک جلسه‌ی ۲۰ـ۲۵ دقیقه‌ای، طراحی خود را به TA پروژه ارائه می‌دهید. در آن جلسه باید آماده باشید تا به سوالات TA درمورد طراحی خود پاسخ دهید و از طراحی خود دفاع کنید.

۶.۲ نمره دهی

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

۳. پیاده‌سازی

نمره‌ی کد به صورت خودکار توسط جاج داده خواهد شد. Pintos در خود یک test suit دارد که شما می‌توانید برای تست به صورت محلی روی ماشین مجازی خود آن را اجرا کنید. در جاج نیز همان تست‌ها اجرا خواهند شد و نتیجه اجرا نمره شما در این بخش خواهد بود که می‌توانید نمره خود را در مخزن خود مشاهده کنید.

۱.۳ کد تست دانشجویان

همان‌طور که گفته‌شد test suit در Pintos از قبل تست‌هایی را برای اجرا دارد اما تمام خواسته‌های تمرین را پوشش نمی‌دهد. شما باید ۲ تست کیس جدید که خواسته‌های تمرین را در test suit پوشش نداده است بنویسید. انتخاب خواسته به عهده شما است. ویژگی یا خواسته‌ای را انتخاب کنید که با تست‌های جدید بهتر و بیشتر پوشش داده شود. مطمئن شوید که کد شما تست‌هایی را که نوشته‌اید با موفقیت اجرا می‌کند. توجه کنید که نام تست‌ها نباید از ۱۵ کاراکتر بیشتر باشد. پس از اتمام نوشتن تست‌ها مطمئن شوید که آن‌ها با دستور make check در pintos/src/userprog اجرا می‌شوند.

۴. کد تست دانشجویان

در کنار تست‌هایی که نوشته‌اید و به همراه کد شما بارگذاری می‌شوند باید گزارشی برای تست‌های خود بنویسید تا در نمره‌دهی استفاده شود. گزارش را در report/ghw2.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.
  - فرایند اجرا و توضیح دقیق خروجی تست
  - خروجی کرنل Pintos وقتی که تست‌های شما اجرا می‌شود. بدین منظور محتوای دو فایل زیر را وارد کنید:

    userprog/build/tests/userprog/YOUR-TEST.output
    userprog/build/tests/userprog/YOUR-TEST.result

  - 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؟

نمره‌دهی به این بخش بر اساس کامل بودن و درست بودن هریک از موارد خواسته شده است.

۵. گزارش نهایی و کیفیت کد

بعد از تمام شدن کد پروژه، باید گزارش نهایی خود را ارسال کنید. این گزارش را در ‘file/reports/project2.md’ در ریپوزیتوری گیت مربوط به گروه خود بنویسید. گزارش نهایی شما باید شامل این بخش‌ها باشد:

- تغییراتی که در سند طراحی اولیه خود داده‌اید، و توضیح اینکه چرا این تغییرات را داده‌اید. دوباره تکرار کردن مباحثی که در جلسه با TA ها بیان شدند اشکالی ندارد.
- تقسیم وظایف گروه، در مورد بخش‌های مختلف پروژه. چه بخش‌هایی از پروژه به خوبی پیش رفت، و کدام بخش‌ها می‌توانست بهبود یابد.
- گزارش تست‌هایی که خودتان نوشته‌اید. (برای جزئیات بیشتر به قسمت قبل مراجعه کنید)

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

- نوشتن کد به صورت پیوسته. همچنین کد شما باید با کد قبلی Pintos ترکیب شود. ایندنت‌ها، فاصله‌بندی و استانداردهای نام‌گذاری
- ساده و خوانا بودن کد
- کامنت گذاری در بخش‌های پیچیده‌ی کد

- نبود کد کامنت شده در ارسال نهایی
- عدم وجود duplicate code و استفاده مناسب از توابع
- دلایل استفاده از داده‌ساختارهای موجود، یا پیاده‌سازی داده ساختار جدید توسط خودتان را توضیح دهید.
- طولانی نبودن بیش از اندازه‌ی خط‌های کد (بیشتر از ۱۰۰ کاراکتر) کامیت‌های موجود در گیت شما. (آبجکت‌فایل ها و لاگ‌ها را کامیت نکنید مگر اینکه واقعا لازم باشند)