به نام خدا

درس سیستم‌های عامل

نیم‌سال دوم ۰۰-۹۹

دانشکدهٔ مهندسی کامپیوتر

دانشگاه صنعتی شریف

مدرس **مهدی خرازی**

تمرین گروهی **سه**

موضوع **فایل سیستم**

موعد تحویل مستند طراحی **ساعت ۲۳:۵۹ دوشنبه ۲۰ اردیبهشت ۱۴۰۰**

موعد تحویل کد و گزارش نهایی **ساعت ۲۳:۵۹ شنبه ۱ خرداد ۱۴۰۰**

با سپاس از دستیاران آموزشی **حسین احمدزاده، حسین ذاکری‌نیا، علی احتشامی، ارشیا مقیمی، مجید گروسی و محمد امیدوار**

اقتباس شده از **CS162 در بهار ۲۰۲۰ در دانشگاه کالیفرنیا، برکلی**

فهرست مطالب

۰ سرآغاز

شما در این تمرین سامانه‌ی مدیریت فایل‌های [1] Pintos را بهبود خواهید داد. در این سند، این ویژگی‌ها به طور مختصر توضیح داده شده‌اند و برای توضیحات بیش‌تر می‌توانید به قسمت منابع در این مستند [2] مراجعه نمایید.

۱ وظیفه‌ی شما

شما در این پروژه سه ویژگی جدید به Pintos می‌افزایید. در این سند این ویژگی‌ها به طور مختصر توضیح داده شده‌اند و برای توضیحات بیش‌تر می‌توانید به قسمت منابع مراجعه نمایید.

نکته: این پروژه نیاز به پیاده‌سازی درست پروژه اول دارد و باید کد پروژه اول را ادامه بدهید. پیشنهاد می‌نماییم در صورتی که مشکلی در آن پروژه داشتید، آن را برطرف نمایید و کد خودتان را ادامه دهید. در صورتی که نمی‌توانستید این کار را انجام دهید، به یکی از دستیاران آموزشی درس اطلاع دهید.

۱.۱ وظیفه ۱: حافظه نهان بافر

در حال حاضر توابع inode_read_at() و inode_write_at() در هر بار فراخوانی به صورت مستقیم به Block Device سیستم پرونده‌ها دسترسی می‌یابند. شما باید یک حافظه نهان بافر [3] برای سیستم پرونده‌ها به Pintos اضافه نمایید تا کارایی خواندن و نوشتن افزایش یابد. حافظه نهان شما هر قطعه از دیسک را به صورت جدا نگه می‌دارد در نتیجه: (۱) شما می‌توانید به درخواست‌های خواندن با داده‌ی نگه‌داری شده زودتر پاسخ دهید و (۲) می‌توانید چند عملیات نوشتن بر روی دیسک را یکی نمایید. حجم بیشینه حافظه نهان بافر شما باید به اندازه ۶۴ قطعه دیسک باشد. انتخاب سیاست جایگذاری قطعه‌ها را شما انجام می‌دهید. اما سیاستی که انتخاب می‌نمایید باید تخمینی از MIN بر فرض محل باشد. برای مثال استفاده از LRU ، NRU (ساعت)، ساعت با شانس n ام و یا لیست‌های شانس دوباره قابل قبول است. اما استفاده از FIFO ، RANDOM و یا MRU قابل قبول نیست. حافظه نهان شما باید به صورت write-back باشد، نه به صورت write-through . علاوه بر این دو تابع، باید تمام عملیات‌های با دیسک از این حافظه نهان استفاده نمایند.

۲.۱ وظیفه ۲: پرونده‌های توسعه پذیر

در حال حاضر در Pintos نمی‌توانید اندازه پرونده‌ها را افزایش دهید چون سیستم پرونده‌های Pintos هر پرونده را به صورت یک تکه در قالب چند قطعه مجاور در دیسک ذخیره می‌کند. وظیفه شما است که سیستم پرونده‌ها را تغییر دهید تا توسعه دادن پرونده‌ها را پشتیبانی نماید. طراحی شما باید دسترسی سریع تصادفی به فایل‌ها را در نظر بگیرد، در نتیجه باید از طراحی مانند FAT خودداری نمایید. یکی از راه حل‌ها می‌تواند استفاده از یک inode فهرست شده باشد که دارای پوینترهای مستقیم، غیر مستقیم و غیر مستقیم دو مرحله‌ای باشد (مانند سیستم پرونده UNIX FFS). بیشینه اندازه پرونده‌ای که باید از آن پشتیبانی نمایید 8 MiB است (2^{23} بایت). همچنین باید از فراخوانی سیستمی inumber(int fd) نیز پشتیبانی نمایید. این فراخوانی باید شماره inode پرونده‌ای که مربوط به این توصیف کننده پرونده [4] است را برگرداند. در انتها برای پیاده‌سازی به حالات خاص توجه نمایید و مطمئن شوید که کد شما در حالات خاص باعث خرابی نشود. در زمان اتمام فضای حافظه و یا اتمام فضای دیسک اطمینان حاصل نمایید که در وضعیت پایداری قرار بگیرید (به خصوص در زمان افزایش اندازه‌ی یک پرونده) و دچار نشت حافظه [5] نشوید.

۳.۱ وظیفه ۳: پوشه

هم‌اکنون سیستم پرونده Pintos از پوشه‌ها پشتیبانی می‌کند اما برنامه‌های کاربر راهی برای استفاده از آن ندارند (پرونده‌ها در ریشه ذخیره می‌شوند). شما باید فراخوانی‌های سیستمی chdir ، mkdir ، readdir و isdir را پیاده سازی نمایید. همچنین باید فراخوانی‌های سیستمی open ، close ، exec ، remove و inumber را نیز ویرایش نمایید تا با پوشه‌ها نیز کار نمایند. پشتیبانی از آدرس‌دهی نسبی نیز

1) File System
2) https://inst.eecs.berkeley.edu/~cs162/sp20/static/projects/proj3.pdf
3) Buffer Cache
4) File Descriptor
5) Memory Leak

در هر فراخوانی سیستمی‌ای که آدرس به عنوان ورودی می‌گیرد نیاز است. برای مثال: اگر در برنامه ای تابع chdir(”my_files/”) صدا زده شده باشد و سپس open(”notes.txt”) را صدا بزنیم، باید در پوشه کار فعلی به دنبال notes.txt بگردیم و پرونده my_files/notes.txt را باز نماییم. همچنین از آدرس‌دهی مطلق مانند open(”/my_files/notes.txt”) نیز باید پشتیبانی نمایید. در آدرس‌دهی‌ها کارکرد علائم خاص ”.” و ”..” را نیز باید پیاده‌سازی نمایید، مانند open(”../logs/foo.txt”) . پردازه‌های فرزند باید پوشه کار پدر را به ارث ببرند. پوشه کار پردازه اول نیز ریشه است.

۴.۱ الزامات هماهنگ سازی

پروژه شما باید همواره thread-safe باشد. اما در این پروژه دیگر اجازه استفاده از یک قفل سراسری [۶] دور سیستم پرونده‌ها را ندارید. در صورتی که این قفل سراسری وجود داشته باشد، دیگر نمی‌توان عملیات خواندن و نوشتن sector های مختلف دیسک را انجام داد. شما باید به روشی قابلیت‌های سیستم پرونده را پیاده‌سازی نمایید که عملیات‌های بر روی دیسک که از هم مستقل هستند (عملیات‌های بر روی بخش‌های مختلف دیسک) را بتوان بصورت همروند انجام داد و یکی برای دیگری صبر ننماید.

معنای مستقل بودن عملیات‌ها چیست؟ برای این پروژه، عملیات‌ها را مستقل در نظر می‌گیریم در صورتی که بر روی بخش‌های مختلف دیسک انجام شوند. این عملیات‌ها را می‌توان بصورت همزمان انجام داد. اگر دو عملیات بر روی یک بخش انجام می‌شوند یا یک پرونده را توسعه می‌دهند، دیگر مستقل نیستند و باید به صورت سری انجام شوند تا ثبات داده حفظ شود. خواندن به صورت همروند ضروری نیست.

چند مثال، فرض نمایید دستورات int notes = open(”/my_files/notes.txt”); و int test = open(”/my_files/test.c”); را اجرا نمودیم:

۱. read(notes) و write(test) را باید بتوان به صورت همروند اجرا نمود. زیرا بر روی بخش‌های مختلفی از دیسک کار می‌نمایند.

۲. read(notes) و write(notes) را نباید بتوان به صورت همروند اجرا نمود. زیرا بر روی بخش‌های یکسانی از دیسک کار می‌نمایند (open پرونده را از ابتدایش باز می‌نماید در نتیجه این دو تابع از بخش ۰ پرونده شروع به کار می‌نمایند) .

۳. read(notes) و read(notes) را می‌توان به صورت همروند اجرا نمود اما الزامی نیست. زیرا بر روی بخش‌های یکسانی از دیسک کار می‌نمایند.

یادداشت: در صورتی که در پروژه ۱ دور سیستم پرونده یک قفل سراسری قرار داده‌اید، حذف کردن آن را فراموش ننمایید.

۲ تحویل‌دادنی‌ها

نمره شما بر اساس معیارهای زیر تعیین می‌گردد:

- ۱۵ درصد سند طراحی و جلسه بررسی طراحی
- ۶۰ درصد پیاده‌سازی و کد (لازم به ذکراست این نمره براساس نمره نمره‌دهنده خودکار و همچنین نظر TA محاسبه خواهد شد.)
- ۱۵ درصد تست‌های دانشجویان
- ۱۰درصد گزارش نهایی و کیفیت کد

۱.۲ سند طراحی و جلسه بررسی طراحی

قبل از این که شروع به کد زدن کنید، بایستی برای پیاده‌سازی خود یک نقشه‌ی راه داشته باشید و بدانید که قصد دارید هر ویژگی را چطور پیاده‌سازی کنید و همچنین باید بتوانید خودتان را قانع کنید که طراحی را به درستی انجام داده‌اید و اشکالی در آن نیست. برای این تمرین گروهی، بایستی که یک مستند طراحی تحویل بدهید و در جلسه‌ی مرور طراحی، شرکت کنید. در این جلسه، دستیاران آموزشی با شما در مورد طراحی مد نظر شما مشورت خواهند کرد و از شما سوالاتی خواهند پرسید و بایستی بتوانید از طراحی خود دفاع کنید.

6) global

۱.۱.۲ بررسی اجمالی طراحی

قالب مستند طراحی این پروژه در آدرس `design/project3.md` قرار دارد. شما باید این مستند را کامل نمایید و در همان آدرس قراردهید. مستند طراحی در فرمت Markdown است. برای مشاهده آن می‌توانید در طرشت به آدرس پرونده بروید و آن را در قالب نهایی مشاهده نمایید. برای هریک از ۲ بخش پروژه شما باید طراحی خود را از چهار جنبه (که در ادامه آورده می‌شود) توضیح دهید.

۱. **داده ساختارها و توابع:** هر داده ساختار، متغیرهای global و یا static و typedef ها و یا enum هایی را که به کد اضافه کردید یا تغییر دادید، به صورت کد C (نه شبه‌کد) بیان کنید. همراه کدها توضیحی حداقلی در مورد هدف این تکه‌کد دهید (توضیحات مفصل‌تر در بخش‌های بعدی مطلوب است).

۲. **الگوریتم‌ها:** در این بخش توضیح می‌دهید که چرا کد شما کار می‌کند! توضیحات شما باید مفصل‌تر از توضیحاتی باشد که در سند تمرین آمده است (سند تمرین موجود است و تکرار آن بی‌مورد است).از طرفی در نظر داشته باشید که توضیحات این بخش باید از سطح کد بالاتر باشد و لازم نیست که خط‌به‌خط کد توضیح داده شود. صرفا باید ما را قانع کنید که کد شما نیازمندی مطرح شده را برطرف کرده است. لازم به ذکر است که در این جا باید شرایطی که استثنا و حالت خاص به حساب می‌آیند توضیح داده شوند. این قسمت باید در قالب و فرمت مشابه مستند طراحی پروژه‌های ۱ و ۲ باشد. ما انتظار داریم که قبل از نوشتن سند طراحی مقدار خوبی از کد Pintos را خوانده باشید. بدون مطالعه‌ی کد نمی‌توانید الگوریتم‌های خود را به درستی توضیح دهید.

۳. **به‌هنگام‌سازی:** این قسمت باید شامل همه منابعی که بین ریسه‌ها به اشتراک گذاشته می‌شوند باشد. برای هر حالت بررسی کنید که چطور منابع قابل دسترسی اند. (به طور مثال از داخل زمان‌بند[7]، داخل interrupt context یا ...) و استراتژی خود برای اطمینان از این‌که این منابع به صورت امن به اشتراک گذاشته می‌شوند یا تغییر داده می‌شوند را توضیح دهید. برای هر منبع نشان‌دهید که طراحی شما رفتار درستی را تضمین می‌کند و از بن‌بست [8] اجتناب می‌کند. به صورت کلی بهترین استراتژی‌های به‌هنگام‌سازی ساده و به سادگی قابل تایید هستند. اگر به سختی می‌توانید استراتژی خود را توضیح دهید این یک نشانه خوبی است که بهتر است استراتژی خود را ساده‌تر کنید.
همچنین درباره هزینه‌ی روش خود از نظر زمان یا حافظه و این‌که این روش تا چه میزان موازات[9] را در هسته کاهش می‌دهد بحث کنید. درباره موازات این‌که هر چند وقت یک‌بار ریسه‌ها روی منابع مشترک رقابت می‌کنند و محدودیت تعداد ریسه‌هایی که می‌توانند همزمان وارد بخش‌های بحرانی مستقل شوند را توضیح‌دهید.

۴. **منطق:** توضیح دهید چرا طراحی شما از دیگر روش‌هایی که بررسی کردید بهتر است و کاستی‌های آن را شرح دهید. مثلاً، به این نکات توجه داشته باشید: چقدر طراحی قابل درک است؟ تا چه اندازه برنامه‌نویسی آن زمان‌بر است؟ پیچیدگی الگوریتم‌های شما از نظر زمانی و حافظه چقدر است؟ آیا می‌توان با هدف افزودن ویژگی‌های بیشتر به این طراحی، به راحتی این طراحی را تغییر داد؟

۲.۱.۲ نکاتی برای مستند طراحی

در بخش الگوریتم و به‌هنگام‌سازی نکات زیر را بررسی کنید. لازم نیست به این سوالات به صورت مستقیم پاسخ دهید ولی مستند طراحی شما باید به وضوح نشان دهد که این مشکلات در طراحی وجود ندارد:

۱. وقتی که یک پردازه فعالانه در حال خواندن یا نوشتن در یک بلوک حافظه‌نهان بافر است، چگونه از این که پردازه‌های دیگر این بلوک را خارج کنند جلوگیری می‌شود؟

۲. در زمان خارج کردن یک بلوک از حافظه نهان چگونه از تلاش باقی پردازه‌ها برای دسترسی به این بلوک جلوگیری می‌شود؟

۳. اگر یک بلوک در حافظه نهان بارگیری شده باشد، چگونه از بارگیری مجدد این بلوک در حافظه نهان جلوگیری می‌شود؟ چگونه از دسترسی دیگر پردازه‌ها به بلوک قبل از کامل شدن بارگیری جلوگیری می‌شود؟

۴. چگونه فایل سیستم شما با گرفتن یک مسیر نسبی مانند `../my_files/notes.txt` پوشه متناظر را پیدا می‌کند؟ برای مسیرهای مطلق مانند `/my_files/solutions.md` چطور؟

7) scheduler
8) deadlock
9) parallelism

۵. آیا یک پردازه کاربر اجازه پاک کردن یک پوشه که cwd مربوط به پردازه در حال اجرا است را دارد؟ در تست‌ها هر دو پاسخ بله و خیر قبول می‌شوند ولی شما باید مطمئن شوید که فایل جدید در یک پوشه پاک شده ساخته نشود.

۶. چگونه کنترل‌کننده فراخوانی‌های سیستمی [10] با استفاده از توصیف کننده فایل [11]، فایل و یا پوشه مربوط را پیدا می‌کند؟ (در واقع توصیف کنید که مثلا با گرفتن fd ۳، چگونه فایل را پیدا می‌کنید.)

۷. شما با رسیدگی به memory exhaustion در C با چک کردن این‌که خروجی تابع malloc مقدار NULL دارد یا نه آشنا هستید. در این پروژه شما باید به فرسودگی فضای دیسک نیز رسیدگی کنید. زمانی که فایل سیستم شما توانایی تخصیص بلوک‌های جدید دیسک را نداشته باشد، شما باید توانایی این را داشته باشید که عملیات در حال انجام را متوقف کنید و به وضعیت خوب قبلی برگردید.

۳.۱.۲ سوال‌های افزون بر طراحی

شما باید به این سوال در مستند طراحی خود پاسخ دهید:

۱. برای این پروژه ۲ ویژگی اختیاری درباره حافظه نهان بافر وجود دارد: write-behind یا read-ahead. یک حافظه نهان بافر با write-behind به صورت متناوب بلوک‌های تغییر داده شده را در بلوک‌های فایل سیستم دستگاه می‌نویسد تا در صورت قطعی برق سیستم اطلاعات زیادی از دست ندهد. بدون این ویژگی حافظه نهان write-back فقط زمانی که یک داده کثیف شده و در حال خارج شدن از حافظه نهان است یا سیستم در حال خاموش شدن است داده را در دیسک می‌نویسد. یک حافظه نهان با read-ahead پیش‌بینی می‌کند که چه داده‌ای را سیستم نیاز خواهد داشت و در پس‌زمینه داده را واکشی [12] می‌کند و می‌تواند به خوبی کارایی را در خواندن فایل‌های متوالی یا خواندن فایل‌ها با الگوهایی با قابلیت پیش‌بینی آسان افزایش دهد.درباره یک استراتژی ممکن برای پیاده سازی این دو ویژگی بحث کنید. شما باید به این سوال جدا از این که قصد پیاده‌سازی این ویژگی‌ها را دارید یا خیر پاسخ دهید.

۴.۱.۲ بازخورد طراحی

شما در یک جلسه ی ۲۵-۳۰ دقیقه ای، طراحی خود را به دستیاران آموزشی پروژه ارائه می‌دهید. در آن جلسه باید آماده باشید تا به سوالات دستیار آموزشی در مورد طراحی خود پاسخ دهید و از طراحی خود دفاع کنید.

۵.۱.۲ نمره‌دهی

مستند طراحی و بازخورد طراحی با هم نمره‌دهی می‌شوند. این بخش ۱۵ نمره دارد که بر اساس توضیحات شما از طراحی در مستند طراحی و پاسخ‌دهی به سوالات در جلسه‌ی بازخورد طراحی نمره‌دهی می‌شود. باید حتما در جلسه بازخورد طراحی حضور داشته باشید تا نمره‌ای به شما تعلق گیرد.

۲.۲ پیاده‌سازی

نمره‌ی پیاده‌سازی شما توسط نمره‌دهنده‌ی خودکار داده می‌شود. Pintos یک مجموعه تست دارد که می‌توانید خودتان آن را اجرا کنید. دقیقا همین تست‌ها برای نمره‌دهی شما استفاده می‌گردد لازم به ذکر است با تغییر دادن تست‌ها تغییری در تست‌هایی که سامانه‌داوری اجرا می‌کند ایجاد نمی‌شود و نمره‌ای که از آن بدست می‌آید، ملاک است.

۱.۲.۲ تست‌های دانشجویان

در حال حاضر Pintos تست هایی برای پروژه ۳ام دارد اما این تست‌ها بخش حافظه نهان بافر را پوشش نمی‌دهند. شما باید ۲ تست از تست های توصیف شده در زیر را پیاده کنید.

10) System call handler
11) file descriptor
12) fetch

- در این سناریو اثربخشی حافظه نهان بافر خود را با استفاده از حساب کردن hit rate می‌سنجید. ابتدا حافظه نهان بافر را خالی کنید سپس یک فایل را باز کنید و به صورت ترتیبی آن را بخوانید.تا مقدار hit rate را برای یک حافظه نهان بافر خالی بدست آورید.پس از آن فایل را ببندید و دوباره آن را باز کرده و به همان صورت بخوانید تا مطمئن شوید hit rate بهبود یافته است.

- در این سناریو توانایی حافظه نهان بافر خود را در ادغام و یکی کردن تغییرات بر روی یک sector را ارزیابی می‌کنید. به این صورت که هر block حافظه دو شمارش‌گر read_cnt و write_cnt را نگه‌داری می‌کند. حال شروع به تولید و نوشتن یک فایل بزرگ به صورت بایت به بایت کنید(حجم فایل تولید شده بیش از ۶۴ کیلوبایت باشد که دوبرابر اندازه بیشینه حافظه نهان بافر است). سپس فایل را به صورت بایت به بایت بخوانید در صورت صحت کارکرد حافظه نهان بافر تعداد نوشتن بر روی دیسک باید در حدود ۱۲۸ مورد باشد(بدلیل اینکه ۶۴ کیلوبایت دارای ۱۲۸ block است.)

- در این سناریو توانایی حافظه نهان بافر خود را در نوشتن یک block کامل بدون اینکه نیاز باشد آن block را بخوانیم می‌آزماییم. به عنوان مثال اگر شما ۱۰۰ کیلوبایت(۲۰۰ block) را در یک فایل می‌نویسید حافظه نهان بافر شما باید ۲۰۰ بار block_write را صدا بزند اما هیچگاه تقاضایی برای خواندن از حافظه نداشته باشد(البته درخواست برای خواندن اطلاعاتی در مورد خود فایل ها قابل قبول است.)

در نظر داشته باشید که تست هایی که می‌نویسید حداقل وابستگی به نحوه پیاده‌سازی شما داشته باشد اما در نظر داشته باشید در نظر گرفتن فرض‌های پایه‌ای در مورد حافظه نهان بافر قابل قبول است.به طور خلاصه تست های خود را طوری پیاده کنید که اگر قرار شد این تست کد گروه دیگری را تست کند نیازی به تغییرات عمده در کد نباشد.
پس از اتمام نوشتن کد تست‌ها اطمینان حاصل کنید که تست‌های شما نیز با اجرای دستور "make check" در شاخه pintos/src/filesys/ اجرا می‌شوند.

۳.۲ گزارش نهایی

نمره‌دهی گزارش شما بر مبنای دو چیز است: **اول**، بایستی برای هر commit، پیام دقیقی نوشته باشید. بدین منظور پس از مشخص کردن پرونده‌هایی که قصد دارید آنها را commit کنید، فرمان زیر را اجرا کنید.

```
git commit
```

بعد از این فرمان، برای شما ویرایشگری باز خواهد شد که در آن پیام خود را بنویسید. پیام شما باید به گونه‌ای شفاف باشد که هم‌گروهی شما با خواندن فقط همین پیام، متوجه وضعیت کنونی پروژه شود. تلاش کنید طوری این پیام‌ها را بنویسید که حتی بدون نیاز به دیدار حضوری با یکدیگر، کار گروهی خود را انجام دهید و هماهنگ بمانید (متاسفانه در حال حاضر واقعا هم امکان دیدن یک‌دیگر را نداریم : (پس سعی کنید در بستر Git هماهنگ باشید).
برای نمونه، می‌توانید اسلوب نوشتن چنین پیام‌هایی را در changelog های هسته‌ی سیستم عامل Linux ببینید. بدیهی است که انتظار نوشتن پیام‌هایی به این تفصیل وجود ندارد اما پیام شما باید حداقل اطلاعات زیر را داشته باشد:

```
Add some feature/Fix some bugs(some should be explained)

Test 27 passed but test 28 and 31 that related to that feature has some issues.
In line ... of file ... this pointer has invalid value that caused that problem(that
    should be explained)
```

به طور خاص، بایستی دقیق بودن پیامهای خود را هنگام تلفیق کردن انشعابهای غیراصلی در انشعاب master رعایت کنید.
دوم ،بعد از اتمام کد پروژه باید یک گزارش از پیاده‌سازی خود آماده کنید. گزارش خود را در مسیر reports/project3.md قرار دهید. موارد زیر در گزارش شما مطلوب است:

- تغییراتی که نسبت به سند طراحی اولیه داشتید و دلایلی را که به خاطر آنها، این تغییرات را اعمال کردید، بیان کنید (در صورت لزوم آوردن بحث‌های خود با دستیار آموزشی مانعی ندارد).

- بیان کنید که هر فرد گروه دقیقا چه بخشی را انجام داد؟ آیا این کار را به صورت مناسب انجام دادید و چه کارهایی برای بهبود عملکردتان می‌توانید انجام دهید.

- گزارشی از نحوه پیاده‌سازی تست ها

کد شما بر اساس کیفیت کد نیز نمره‌دهی خواهد شد. موارد بررسی از این دست می‌باشند:

- آیا کد شما مشکل بزرگ امنیتی در بخش حافظه دارد (به صورت خاص رشته‌ها در زبان C)؟ memory leak و نحوه مدیریت ضعیف خطاها نیز بررسی خواهد شد.
- آیا از یک Code Style واحد استفاده کردید؟ آیا style مورد استفاده توسط شما با Pintos هم‌خوانی دارد؟ (از نظر فرورفتگی و نحوه نام‌گذاری)
- آیا کد شما ساده و قابل درک است؟
- آیا کد پیچیده‌ای در بخشی از کدهای خود دارید؟ در صورت وجود آیا با قرار دادن توضیحات مناسب آن را قابل فهم کردید؟
- آیا کد Comment شده‌ای در کد نهایی خود دارید؟
- آیا کدی دارید که کپی کرده باشید؟
- آیا الگوریتم‌های linked list را خودتان پیاده‌سازی کردید یا از پیاده‌سازی موجود استفاده کردید؟
- آیا طول خط کدهای شما بیش از حد زیاد است؟ (۱۰۰ کاراکتر)
- آیا در مخزن Git شما، پرونده‌های دودویی حضور دارند؟ (پرونده‌های دودویی و پرونده‌های log را commit نکنید مگر این که واقعا لازم باشند.)

۱.۳.۲ گزارش تست‌های پیاده‌سازی شده

نیاز است برای نمره‌دهی به بخش تست پیاده‌سازی شده توسط شما گزارشی از آن را در گزارش نهایی خود بیاورید. گزارش شما باید برای هر ۲ تست بخش های زیر را پوشش دهد

- توضیح دهید تست شما دقیقا چه چیزی را تست می کند.
- توضیح دهید چگونه این امر را تست می‌کنید و همینطور به صورت کیفی بیان کنید که خروجی مورد نظر باید چگونه باشد تا تست پاس شود.
- خروجی خود هسته وقتی تست اجرا می‌شود و همچنین خروجی تست را در این بخش بیاورید.برای اینکار محتوای دو فایل your-test-1.output و your-test-1.result که در مسیر filesys/build/tests/filesys/extended یافت می‌شود را کپی کنید.
- دو ایراد بالقوه و غیر بدیهی هسته را بیان کنید و نشان دهید که در صورت بروز این خطا تست شما چه خروجی خواهد داشت.گزارش شما باید به این صورت باشد:

  "اگر هسته X را به جای Y انجام دهد آنگاه خروجی تست Z خواهد بود."

  شما باید برای هر تست دو ایراد مجزا پیدا کنید اما ایراد های دو تست می‌تواند یکسان باشد یا اشتراک داشته باشند.لازم به ذکر است که ایراد بیان شده باید مربوط به سناریو تست شما باشد به عنوان مثال سناریو زیر قابل قبول نیست: اگر هسته خطای نحوی [۱۳] داشته باشد آنگاه تست اجرا نمی‌شود.

در آخر تجربه خود از نوشتن تست برای Pintos را برای ما بیان کنید. نمره این بخش براساس تلاش شما که در نوشتن گزارش منعکس می‌شود داده می‌شود و اگر بخش‌های بالا تکمیل باشد نگرانی از جهت نمره برای این بخش نداشته باشید.

13) Syntax error