به نام خدا

پروژه پنجم آزمایشگاه سیستم‌عامل

(مدیریت حافظه در xv6)

طراحان: آرین فیروزی، بابک حسینی محتشم

در این پروژه شیوه مدیریت حافظه در سیستم‌عامل xv6 بررسی شده و قابلیت‌هایی به آن افزوده خواهد شد. در ادامه ابتدا مدیریت حافظه به طور کلی در xv6 معرفی شده و در نهایت صورت آزمایش شرح داده خواهد شد.

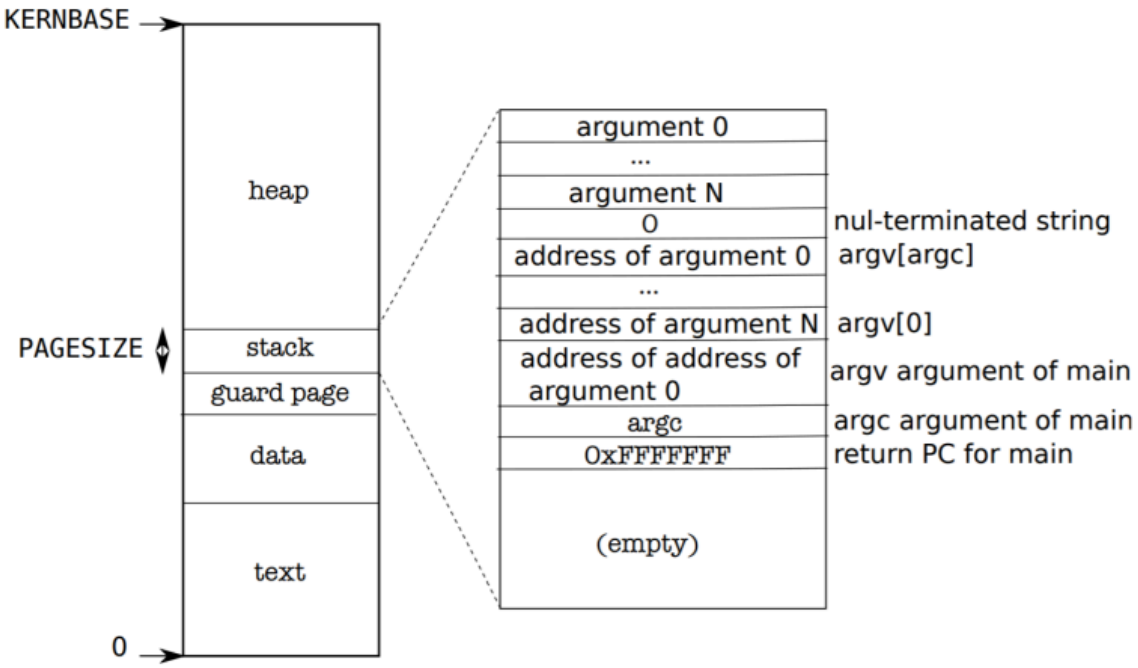
مقدمه

یک برنامه، حین اجرا تعامل‌های متعددی با حافظه دارد. دسترسی به متغیرهای ذخیره شده و فراخوانی توابع موجود در نقاط مختلف حافظه مواردی از این ارتباط‌ها می‌باشد. معمولاً کد منبع دارای آدرس نبوده و از نمادها برای ارجاع به متغیرها و توابع استفاده می‌شود. این نمادها توسط کامپایلر و پیونددهنده[1] به آدرس تبدیل خواهد شد. حافظه یک برنامه سطح کاربر شامل بخش‌های مختلفی مانند کد، پشته[2] و هیپ[3] است. این ساختار برای یک برنامه در xv6 در شکل زیر نشان داده شده است:

[1] Linker

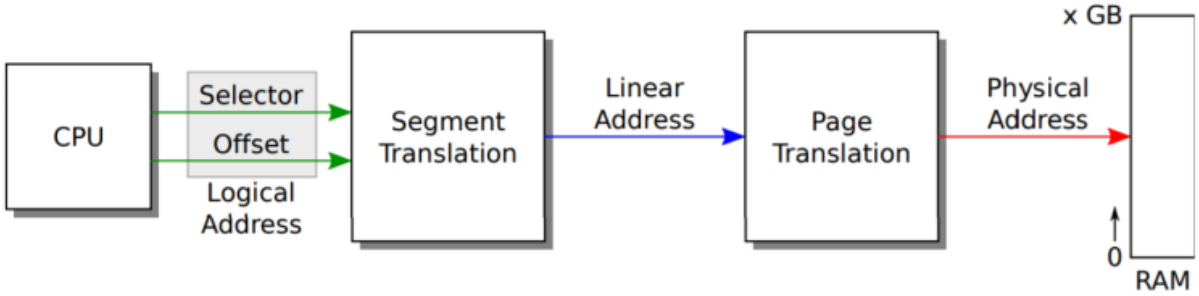
[2] Stack

[3] Heap



۱) راجع به مفهوم ناحیه مجازی[4] در لینوکس به طور مختصر توضیح داده و آن را با xv6 مقایسه کنید. صفحه های محافظ[5] به چه منظور استفاده میشوند؟

همان‌طور که در آزمایش یک ذکر شد، در مد محافظت‌شده[6] در معماری x86 هیچ کدی (اعم از کد هسته یا کد برنامه سطح کاربر) دسترسی مستقیم به حافظه فیزیکی[7] نداشته و تمامی آدرس‌های برنامه از خطی[8] به مجازی[9] و سپس به فیزیکی تبدیل می‌شوند. این نگاشت در شکل زیر نشان داده شده است:



[4] Virtual Memory Area
[5] Guard Page
[6] Protected Mode
[7] Physical Memory
[8] Linear
[9] Virtual

به همین منظور، هر برنامه یک جدول اختصاصی موسوم به جدول صفحه [10] داشته که در حین فرآیند تعویض متن[11] بارگذاری شده و تمامی دسترسی‌های حافظه (اعم از دسترسی به هسته یا سطح کاربر) توسط آن برنامه توسط این جدول مدیریت می‌شود.

به علیت عدم استفاده صریح از قطعه‌بندی در بسیاری از سیستم‌عامل‌های مبتنی بر این معماری، می‌توان فرض کرد برنامه‌ها از صفحه‌بندی [12]و لذا آدرس مجازی استفاده می‌کنند. علت استفاده از این روش مدیریت حافظه در درس تشریح شده است. به طور مختصر می‌توان سه علت عمده را برشمرد:

۱) **ایزوله‌سازی پردازه‌ها از یکدیگر و هسته از پردازه‌ها:** با اجرای پردازه‌ها در فضاهای آدرس[13] مجزا، امکان دسترسی یک برنامه مخرب به حافظه برنامه‌های دیگر وجود ندارد. ضمن این که با اختصاص بخش مجزا و ممتازی از هر فضای آدرس به هسته امکان دسترسی محافظت‌نشده پردازه‌ها به هسته سلب می‌گردد.

۲) **ساده‌سازی ABI سیستم‌عامل:** هر پردازه می‌تواند از یک فضای آدرس پیوسته (از آدرس مجازی صفر تا چهار گیگابایت در معماری x86) به طور اختصاصی استفاده نماید. به عنوان مثال کد یک برنامه در سیستم‌عامل لینوکس در معماری x86 همواره (در صورت عدم استفاده از تصادفی‌سازی چینش فضای آدرس[14]) از آدرس 0x08048000 آغاز شده و نیاز به تغییر در آدرس‌های برنامه‌ها متناسب با وضعیت جاری تخصیص حافظه فیزیکی نمی‌باشد.

۳) **استفاده از جابه‌جایی حافظه:** با علامت‌گذاری برخی از صفحه‌ها کم استفاده (در جدول صفحه) و انتقال آن‌ها به دیسک، حافظه بیشتری در دسترس خواهد بود. به این عمل جابه‌جایی حافظه[15] اطلاق می‌شود.

---

[10] Page Table

[11] Context Switch

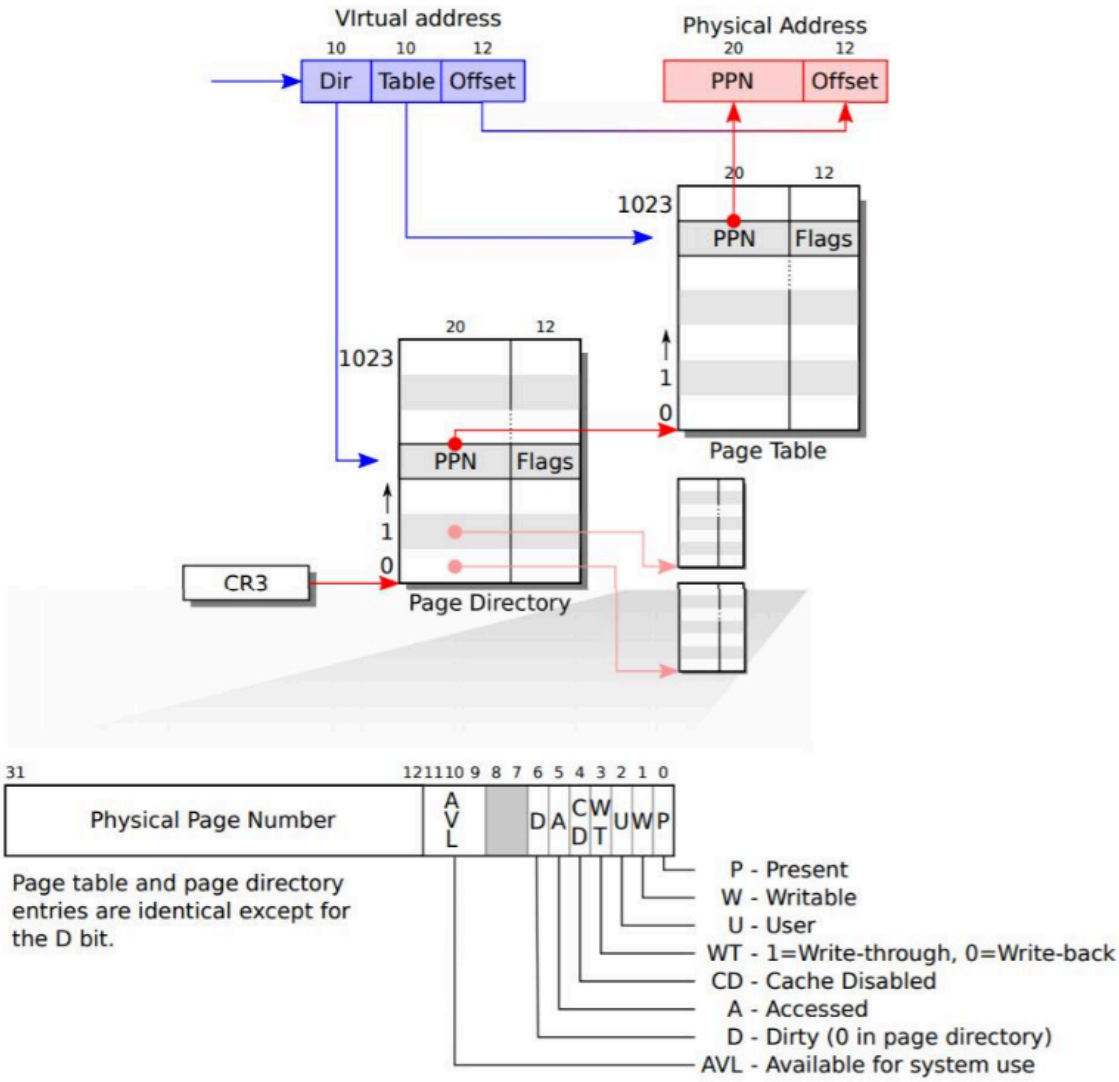
[12] Paging

[13] Address Spaces

[14] Address Space Layout Randomization

[15] Memory Swapping

ساختار جدول صفحه در معماری x86 (در حالت بدون گسترش آدرس فیزیکی[16] (PAE) و گسترش اندازه صفحه[17] (PSE)) در شکل زیر نشان داده شده است.



هر آدرس مجازی توسط اطلاعات این جدول به آدرس فیزیکی تبدیل می‌شود. این فرآیند، سخت‌افزاری بوده و سیستم‌عامل به طور غیرمستقیم با پر کردن جدول، نگاشت را صورت می‌دهد. جدول صفحه دارای سلسله مراتب دو سطحی بوده که به ترتیب Page Directory و Page Table نام دارند. هدف از ساختار سلسله‌مراتبی کاهش مصرف حافظه است.

۲) چرا ساختار سلسله‌مراتبی منجر به کاهش مصرف حافظه می‌گردد؟

۳) مفهوم هر بیت یک مدخل (۳۲ بیتی) در هر سطح چیست؟ چه تفاوتی میان آن‌ها وجود دارد؟

---

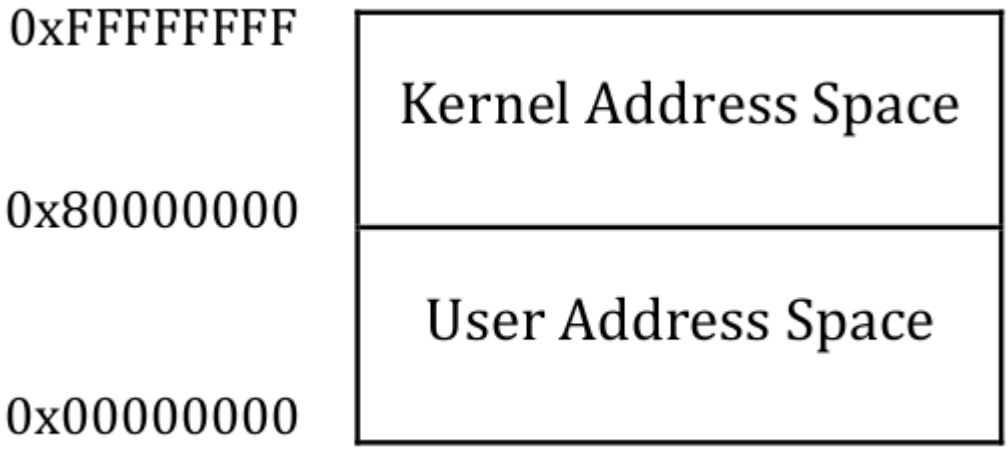
[16] Physical Address Extension

[17] Page Size Extension

مدیریت حافظه در xv6

ساختار فضای آدرس در xv6

در xv6 نیز مد اصلی اجرای پردازنده، مد محافظت‌شده و ساز و کار اصلی مدیریت حافظه صفحه‌بندی است. به این ترتیب نیاز خواهد بود که پیش از اجرای هر کد، جدول صفحه آن در دسترس پردازنده قرار گیرد. کدهای اجرایی در xv6 شامل کد پردازه‌ها)کد سطح کاربر(و ریسه هسته متناظر با آن‌ها و کدی است که در آزمایش یک، کد مدیریت‌کننده نام‌گذاری شد.[18] آدرس‌های کد پردازه‌ها و ریسه هسته آن‌ها توسط جدول صفحه‌ای که اشاره‌گر به ابتدای Page Directory آن در فیلد pgdir از ساختار (proc خط ۲۳۳۹) قرار دارد، نگاشت داده می‌شود. نمای کلی ساختار حافظه مجازی متناظر با جدول این دسته در شکل زیر نشان داده شده است:



دو گیگابایت پایین جدول صفحه مربوط به اجزای مختلف حافظه سطح کاربر پردازه است. دو گیگابایت بالای جدول صفحه مربوط به اجزای ریسه هسته پردازه بوده و در تمامی پردازه‌ها یکسان است. آدرس تمامی متغیرهایی که در هسته تخصیص داده می‌شوند در این بازه قرار می‌گیرد. جدول صفحه کد مدیریت‌کننده هسته، دو گیگابایت پایینی را نداشته (نگاشتی در این بازه ندارد) و دو گیگابایت بالای آن دقیقا شبیه به پردازه‌ها خواهد بود. زیرا این کد، همواره در هسته اجرا شده و پس از بوت غالبا، در اوقات بیکاری سیستم اجرا می‌شود.

[18] بحث مربوط به پس از اتمام فرآیند بوت است. به عنوان مثال، در بخشی از بوت، از صفحات چهار مگابایتی استفاده شد که از آن صرف نظر شده است.

کد مربوط به ایجاد فضاهای آدرس در xv6

فضای آدرس کد مدیریت‌کننده هسته در حین بوت، در تابع main ایجاد می‌شود. به این ترتیب که تابع kvmalloc فراخوانی شده (خط ۱۲۲۰) و به دنبال آن تابع setupkvm متغیر kpgdir را مقداردهی می‌نماید (خط ۱۸۴۲). به طور کلی هر زمان نیاز به مقداردهی ساختار فضای آدرس هسته باشد، از setupkvm استفاده خواهد شد. با بررسی تابع setupkvm (خط ۱۸۱۸) می‌توان دریافت که در این تابع، ساختار فضای آدرس هسته بر اساس محتوای آرایه (kmap خط ۱۸۰۹) دیده می‌شود.

۴) تابع kalloc چه نوع حافظه‌ای تخصیص می‌دهد؟ (فیزیکی یا مجازی)

۵) تابع mappages چه کاربردی دارد؟

۶) آیا تمام قسمت های حافظه مجازی از آدرس صفر تا proc->sz از طریق کاربر قابل استفاده اند؟

فضای آدرس مجازی نخستین برنامه سطح کاربر (initcode) نیز در تابع main ایجاد می‌گردد. به طور دقیق‌تر تابع userinit (خط ۱۲۳۵) فراخوانی شده و توسط آن ابتدا نیمه هسته فضای آدرس با اجرای setupkvm (خط ۲۵۲۸) مقداردهی خواهد شد. نیمه سطح کاربر نیز توسط تابع inituvm ایجاد شده تا کد برنامه نگاشت داده شود. فضای آدرس باقی پردازه‌ها در ادامه اجرای سیستم توسط توابع fork یا exec مقداردهی می‌شوند. به این ترتیب که هنگان ایجاد پردازه فرزند توسط fork با فراخوانی تابع copyuvm (خط ۲۵۹۲) فضای آدرس نیمه هسته ایجاد شده (خط ۲۰۴۲) و سپس فضای آدرس نیمه کاربر از والد کپی می‌شود. این کپی با کمک تابع walkpgdir (خط ۲۰۴۵) صورت می‌پذیرد.

۷) راجع به تابع walkpgdir توضیح دهید. این تابع چه عمل سخت‌افزاری را شبیه‌سازی می‌کند؟

وظیفه تابع exec اجرای یک برنامه جدید در ساختار بلوک کنترلی پردازه[19] یک پردازه موجود است. معمولا پس از ایجاد فرزند توسط fork فراخوانده شده و کد، داده‌های ایستا، پشته و هیپ برنامه جدید را در فضای آدرس فرزند ایجاد می‌نماید. بدین ترتیب با اعمال تغییراتی در فضای آدرس موجود، امکان اجرای یک برنامه جدید فراهم می‌شود. روش متداول Shell در سیستم‌های‌های مبتنی بر یونیکس از جمله xv6 برای اجرای برنامه‌های جدید مبتنی بر exec است. Shell پس از دریافت ورودی و فراخوانی fork1 تابع runcmd را برای اجرای دستور ورودی، فراخوانی می‌کند (خط ۸۷۲۴). این تابع نیز در نهایت تابع exec را فراخوانی می‌کند (خط ۸۶۲۶). چنانچه در آزمایش یک مشاهده شد، خود Shell نیز در حین بوت با فراخوانی سیستمی sys_exec (خط ۸۴۱۴) و به دنبال آن exec ایجاد شده و فضای آدرسش به جای فضای آدرس نخستین پردازه initcode چیده می‌شود. در پیاده‌سازی exec مشابه قبل setupkvm فراخوانی شده (خط ۶۶۳۷) تا فضای‌آدرس هسته تعیین گردد. سپس با فراخوانی allocuvm فضای مورد نیاز برای کد و داده‌های برنامه جدید (خط ۶۶۵۱) و صفحه محافظ و پشته (خط ۶۶۶۵) تخصیص داده می‌شود. دقت شود تا این مرحله تنها تخصیص صفحه صورت گرفته و باید این فضاها در ادامه توسط توابع مناسب با داده‌های مورد نظر پر شود (به ترتیب خطوط ۶۶۵۵ و ۶۶۸۶).

---

[19] Process Control Block

۸) توابع allocuvm و mappages که در ارتباط با حافظه‌ی مجازی هستند را توضیح دهید.

۹) شیوه‌ی بارگذاری [20] برنامه در حافظه توسط فراخوانی سیستمی exec را شرح دهید.

شرح پروژه

هدف پروژه:

شما در درس با صفحه‌بندی[21] و الگوریتم‌های جایگزینی[22] صفحه آشنا شده اید. در این تمرین می‌خواهیم یک مکانیسم صفحه بندی ساده برای xv6 بسازیم و چند الگوریتم جایگزینی را با برنامه‌های سطح کاربر آزمایش کنیم.

الزامات پیاده‌سازی:

۱. ساختار داده جدول صفحه[23]

یک جدول مرکزی برای نگهداری اطلاعات مربوط به صفحه های مجازی تعریف و پیاده‌سازی کنید. هر ورودی این جدول شامل اطلاعات زیر است:

- **اشاره‌گر به آدرس فیزیکی شروع حافظه**
- **شماره‌ی صفحه مجازی**
- **شماره پردازه مالک صفحه**
- **معتبر بودن یا نبودن صفحه**
- **پارامترهای مورد نیاز برای الگوریتم‌های جایگزینی و مقایسه آن‌ها**

توجه کنید که این جدول باید از تعداد ۴ صفحه پشتیبانی کند و صفحه‌های موجود در آن باید در ابتدا با استفاده از kalloc ایجاد شده و هنگام جایگزینی، با کپی کردن محتوای صفحه مورد نظر از حافظه پر شوند. همچنین برای پیاده سازی الگوریتم‌های جایگزینی می‌توانید چند پارامتر کمکی در این ساختار قرار دهید. برای دسترسی راحت‌تر به معیارهای عملکرد هر الگوریتم بهتر است فیلدهایی در این جدول قرار دهید. معیارهای مقایسه در بخش‌های بعدی توضیح داده شده اند.

---

[20] Load

[21] Paging

[22] Page Replacement

[23] Page Table

توجه داشته باشید که این جدول باید با قفل پیاده شود تا خواندن و نوشتن در آن به درستی انجام شود. برای این کار، لازم است دو فراخوانی سیستمی تعریف کنید که فراخوانی اول، با دریافت پوینتر به آدرس مورد نظر عدد داده شده را در صفحه موجود در جدول می‌نویسد و فراخوانی سیستمی دوم با دریافت پوینتر به آدرس، عدد آن آدرس را برمی‌گرداند.

توجه کنید که در صورتی که صفحه مورد نظر در جدول موجود نباشد، باید طبق الگوریتم جایگزینی، محتوای یکی از صفحات جدول در حافظه اصلی نوشته شود و محتوای صفحه مورد نظر از صفحه اصلی به جای آن نوشته شود.

بیت اعتبار در جدول در ابتدا باید برای تمام صفحات جدول صفر باشد تا نشان دهد که محتوای صفحات اعتبار ندارد و نباید آن‌ها را در حافظه کپی کرد ولی پس از کپی کردن صفحات حافظه به جدول، اعتبار صفحه کپی شده باید یک شود. همچنین در صورت خارج شدن یک پردازه، بیت اعتبار هر یک از صفحات آن در جدول باید غیرفعال شوند.

۲. الگوریتم‌های جایگزینی صفحات

برای این تمرین شما باید چهار الگوریتم زیر را پیاده کنید:

- **FIFO (First in First Out)**
- **LRU (Least Recently Used)**
- **LFU (Least Frequently Used)**
- **Clock Algorithm (Second-Chance Algorithm)**

جایگزینی صفحات جدول باید تنها در صورتی انجام شود که صفحه مورد نظر در جدول نباشد و تمام صفحات موجود در جدول اعتبار داشته باشند. برای مطالعه بیشتر در مورد این الگوریتم‌ها می‌توانید به بخش ۴ فصل ۱۰ از کتاب مراجعه کنید.

۱۰) در مورد الگوریتم های مورد استفاده در سیستم عامل های ویندوز و لینوکس تحقیق کنید و به طور خلاصه کاربرد های هر کدام را ذکر کنید.

۳. معیارهای ارزیابی الگوریتم‌ها

برای ارزیابی الگوریتم‌ها شما در هر بار اجرای برنامه‌ها باید معیارهای زیر را به ازای هر یک از الگوریتم‌های جایگزینی به دست بیاورید و در گزارش جدولی برای مقایسه آن ارائه کنید:

- **مقدار hit count یعنی تعداد دفعاتی که صفحه مورد نظر در جدول وجود داشتند**
- **مقدار hit ratio یعنی درصد صفحاتی که مورد نظر که در جدول وجود داشتند**
- **زمان اجرای برنامه**

برای این کار، یک فراخوانی سیستمی بنویسید که معیارهای ذکر شده را در طول اجرای سیستم‌عامل چاپ کند و این فراخوانی سیستمی را در انتهای برنامه‌های خود صدا بزنید.

همچنین در گزارش مطابقت معیارها را با آنچه در مورد الگوریتم‌ها می‌دانید مقایسه کنید و برای هر معیار توجیه مناسبی ارائه کنید.

۴. برنامه سطح کاربر

در این بخش شما برنامه‌های سطح کاربری طراحی می‌کنید که دسترسی به حافظه زیادی نیاز دارند. **هر برنامه** را با استفاده از **هر الگوریتم** اجرا کنید و نتایج را ذخیره کنید. برای اطمینان از کارکرد درست: *الف)* تعداد صفحات جدول را تغییر دهید و *ب)* برنامه‌ها را به صورت چندپردازه‌ای بنویسید و اجرا کنید. مقدار به دست آمده در برنامه‌ها مهم نیست و می‌توانید بدون مقداردهی اولیه از garbage value ها استفاده کنید، فقط توجه کنید که خطاهای ممکن را به درستی هندل کنید.

برای اطمینان از درستی کارکرد بعد از هر جایگزینی صفحه که اجرا می‌شود **شماره صفحات** موجود در جدول را چاپ کنید و در گزارش خود نشان دهید.

۱.۴. برنامه ۱:

برنامه‌ای بنویسید که با استفاده از حلقه‌ای صفحات را پیمایش می کند. کافیست درون حلقه به میزانی در حافظه پیش بروید که حداقل ۱۰ صفحه پیمایش شده باشد. نیازی به پیمایش تمام عناصر صفحه نیست، هدف دسترسی به حداقل یک عنصر در صفحه است.

۲.۴. برنامه ۲:

برنامه‌ای مشابه به برنامه ۱ بنویسید که در آن به جای ۱۰ صفحه از ۵ صفحه اطلاعات خوانده می‌شود و این کار ۴ بار اجرا می‌شود. ترتیب خواندن صفحات به صورت حلقوی بوده و کل برنامه در دو حلقه تودرتو اجرا می‌شود.

۳.۴. برنامه ۳:

شبه‌کد این برنامه به این صورت است (مقدار دهی اولیه نیاز نیست، صرفا برای درک آسان‌تر مسئله صفحات تعریف شده اند):

```
Initialize array HotPages[3]
Initialize array ColdPages[5]
```

```
for_ = 1 to 50:
    for i = 1 to 100:
        access HotPages[0]
        access HotPages[1]
        access HotPages[2]

    for j = 1 to 5:
        access ColdPages[j]
```

۴.۴. برنامه ۴:

شبه‌کد این برنامه به این صورت است:

```
Initialize array Pages[5]

for _ = 1 to 100:
    access Pages[0]
    access Pages[1]
    access Pages[2]
    access Pages[3]


    access Pages[0]
    access Pages[1]

    access Pages[4]
```

۱۱) مقایسه ی برنامه ها را طبق معیار هایی که خواسته شده انجام دهید و بنویسید آیا طبق آنچه که از الگوریتم ها میدانید این رفتار قابل پیش بینی است یا خیر؟

نکات پیاده سازی

- پروژه خود را در Github یا Gitlab پیش برده و در نهایت یک نفر از اعضای گروه کدها را به همراه پوشه git و گزارش زیپ کرده و در سامانه با فرمت OS-Lab5-<SID1>-<SID2>-<SID3>.zip آپلود نمایید.
- رعایت نکردن مورد فوق کسر نمره را به همراه خواهد داشت.
- استفاده از هوش مصنوعی مجاز است فقط توجه کنید که در تحویل دانش شما در مورد پروژه معیار قرار خواهد گرفت بنابراین سعی کنید بر روی کد تحویل داده شده تسلط کافی داشته باشید.
- بخش خوبی از نمرهٔ شما را پاسخ دهی به سوالات مطرح شده، تشکیل میدهد که به شما در درک نحوه کارکرد xv6 و پیاده سازی قسمت سوالات کمک میکند.
- پاسخ سوالها را تا حد ممکن کوتاه بنویسید.
- همه افراد می بایست به پروژه مسلط باشند و نمره تمامی اعضای گروه لزوما یکسان نیست.
- تمامی مواردی که در جلسه توجیهی گروه تیمز و فروم درس مطرح می شوند، جزئی از پروژه خواهند بود در صورت وجود هرگونه سوال یا ابهام میتوانید با ایمیل دستیاران مربوطه یا گروه اسکایپی درس در ارتباط باشید.
- این تمرین صرفا برای یادگیری شما طرح شده است.
- در صورت محرز شدن تقلب در تمرین مطابق با قوانین درس برخورد خواهد شد.

موفق باشید