**به نام خدا**

**گزارش پروژه «5» آزمایشگاه سیستم عامل**

**استاد: دکتر کارگهی**

**گروه 11**

**امیرارسلان شهبازی 810101451**

**سید محمدحسین مظهری 810101520**

**محمدمهدی صمدی 810101465**

**لینک مخزن** <https://github.com/AMIRSH1383/OS-SMS_LAB5.git>

**مقدمه**

**1) راجع به مفهوم ناحیه مجازی در لینوکس به طور مختصر توضیح داده و آن را با xv6 مقایسه کنید.**

در لینوکس، هسته از نواحی حافظه مجازی با پیگیری memory mapping های پردازه استفاده می کند. برای مثال یک پردازه یک vma برای کد، یک vma برای هر نوع دیتا و یک vma برای هر memory mapping دارد. هر vma شامل تعدادی پیج است که هر کدام از این پیج ها یک ورودی به page table دارد اما xv6 از آدرس های مجازی 32 بیتی استفاده می کند که فضای آدرسی مجازی 4 گیگابایتی ایجاد می کند.  
همچنین xv6 از جدول دو سطحی استفاده می کند که مفهومی از حافظه مجازی ندارد.

**2) چرا ساختار سلسله مراتبی منجر به کاهش مصرف حافظه می گردد؟**

در ساختار سلسله مراتبی، پردازه ها و تسک ها به راحتی می توانند با به اشتراک گذاشتن کدها و داده ها توسط mapping بخش مناسب به صفحات فیزیکی از مصرف اضافی حافظه جلوگیری کنند.

**3) محتوای هر بیت یک مدخل در هر سطح چیست؟ چه تفاوتی میان آن ها وجود دارد؟**

در مدخل سطح page directory برای اشاره به سطح بعدی از 20 بیت استفاده شده است.  
همچنین 12 بیت برای سطح دسترسی نگهداری می شود.  
این 12 بیت در هردو سطح وجود دارد اما در سطح page table از 20 بیت برای آدرس صفحه فیزیکی استفاده می شود.

در بیت D(Dirty) با هم تفاوت دارند.  
در page directory این بیت به این معناست که صفحه باید در دیسک نوشته شود تا تغییرات اعمال شود. اما در page table این بیت معنایی ندارد.

**4) تابع kalloc چه نوع حافظه ای اختصاص می‌دهد؟(فیزیکی یا مجازی)**

این تابع یک حافظه فیزیکی به فضای 4096 بایتی اختصاص می‌دهد . پوینتری را return می‌کند که کرنل می‌تواند از آن استفاده کند و در صورتی که نتواند حافظه ای تخصیص دهد 0 را return می‌کند.

**5)تابع mappages چه کاربردی دارد ؟**

این تابع آدرس directory page و آدرس یک خانه حافظه مجازی و آدرس یک خانه حافظه فیزیکی و سایز را می‌گیرد و صفحه موجود در حافظه فیزیکی را با توجه به آدرس و سایزی که به آن داده ایم در آدرسی که در حافظه مجازی به آن داده ایم بارگذاری می‌کند.این کار برای دسترسی به متغیرهای پردازه در حال اجراست تا صفحه آن بتواند به درستی بارگذاری شود و تغییر داده شود . همچنین این تابع صفحه جدید را به pgdir اضافه می‌کند و کلا حافظه مجازی را به فیزیکی متصل می‌کند.

**6) این سوال وجود نداشت و شماره گذاری سوالات اشتباه شده بود.**

**7)راجع به تابع walkpgdir توضیح دهید. این تابع چه عمل سخت افزاری را شبیه سازی می‌کند؟**

تابع فوق آدرس directorypage و همچنین خانه ای از حافظه مجازی را گرفته و آدرس pagetable ای که در حافظه مجازی داریم را از directory page برمی‌گرداند و در صورت نیاز جدول مورد نظر را می‌سازد.

این تابع عمل سخت افزاری ترجمه آدرس مجازی به فیزیکی را شبیه سازی می‌کند.

**8) توابع allocuvm و mappages که در ارتباط با حافظه مجازی هستند را توضیح دهید.**

mappaggs: این تاﺑﻊ ﻣﺴﺌﻮل ﺑﺮﻗﺮاری ﯾﮏ ﻧﮕﺎﺷﺖ بین ﻣﺤﺪوده ای از آدرس ﻫﺎی ﻣﺠﺎزی و آدرس ﻫﺎی فیزﯾﮑﯽ اﺳﺖ. این تابع زﻣﺎﻧﯽ اﺳﺘﻔﺎده می ﺷﻮد ﮐﻪ ﺳﯿﺴﺘﻢ ﻋﺎﻣﻞ ﻧﯿﺎز ﺑﻪ اﯾﺠﺎد ﯾﮏ ارﺗﺒﺎط بین ﺣﺎﻓﻈﻪ ﻣﺠﺎزی ﮐﻪ ﻓﺮآﯾﻨﺪﻫﺎ می ﺑﯿﻨﻨﺪ و ﺣﺎﻓﻈﻪ فیزﯾﮑﯽ ﮐﻪ ﺳﺨﺖ اﻓﺰار اﺳﺘﻔﺎده می ﮐﻨﺪ دارد.

ﭘﺎراﻣﺘﺮﻫﺎ: این تابع ﯾﮏ دایرﮐﺘﻮری ﺻﻔﺤﻪ، ﻣﺤﺪوده ای از آدرس ﻫﺎی ﻣﺠﺎزی، آدرس ﻫﺎی ﻓlﺰﯾﮑﯽ ﻣ ﻨﺎﻇﺮ و flags permission را ﺑﻪ ﻋﻨﻮان ﭘﺎراﻣﺘﺮ می گیرد.

ﻧﮕﺎﺷﺖ: این تابع ﺑﺮ روی ﻣﺤﺪوده آدرس ﻫﺎی ﻣﺠﺎزی ﺣﻠﻘﻪ می زﻧﺪ. ﺑﺮای ﻫﺮ آدرس، آن را در دایرکتوری ﺻﻔﺤﻪ متناظر می ﯾﺎﺑﺪ.

ورودی ﺟﺪول ﺻﻔﺤﻪ :(PTE) اﮔﺮ ﯾﮏ Entry Table Page ﺑﺮای آدرس ﻣﺠﺎزی ﻓﻌﻠﯽ وﺟﻮد ﻧﺪاﺷته ﺑﺎﺷﺪ، mappages ﯾﮑﯽ اﯾﺠﺎد می ﮐﻨﺪ. ﺳﭙﺲ آدرس فیزﯾﮑﯽ PTE را ﺑﻪ آدرس ﻓیزﯾﮑﯽ متناظر تنظیم می ﮐﻨﺪ و ﬂag را اﻋﻤﺎل aمی ﮐﻨﺪ.

ﻣﺪیریتﺧﻄﺎ: اﮔﺮ mappages ﻫﻨﮕﺎم اﯾﺠﺎد ﯾﮏ PTE ﺑﺎ ﺧﻄﺎ ﻣﻮاﺟﻪ ﺷﻮد )ﺑﺮای مثال، اﮔﺮ ﺣﺎﻓﻈﻪ آزادی ﺑﺮای ﯾﮏ ﺟﺪول ﺻﻔﺤﻪ ﺟﺪﯾﺪ وﺟﻮد ﻧﺪاشته ﺑﺎﺷﺪ(، ﯾﮏ ﺧﻄﺎ ﺑﺮمی ﮔﺮداﻧﺪ.

در اﺻﻞ، mappages ﻣﺴﺌﻮل اﻃﻤﯿﻨﺎن از این اﺳﺖ ﮐﻪ وﻗﺘﯽ ﯾﮏ ﻓﺮآﯾﻨﺪ ﺑﻪ ﯾﮏ آدرس ﻣﺠﺎزی دﺳﺘﺮﺳﯽپیدا aﯽ ﮐﻨﺪ، ﺳﺨﺖ اﻓﺰار ﺑﻪ درﺳﺘﯽ آن دﺳﺘﺮﺳﯽ را ﺑﻪ آدرس فیزﯾﮑﯽ ﻣﻨﺎﺳﺐ ﻣﭗ می ﮐﻨﺪ.

allocuvm: این تابع مسئول افزایش حافظه مجازی کاربر در یک دایرکتوری صفحه خاص است. دو حالت وجود دارد که این تابع می تواند fail شود:

حالت 1: اگر تابع kalloc شکست بخورد. این تابع مسئول برگرداندن ادرس یک صفحه جدید و در حال حاضر استفاده نشده در رم است. اگر 0 برگرداند یعنی در حال حاضر صفحه استفاده نشده ای وجود ندارد.

حالت 2: اگر تابع mappages شکست بخورد. این تابع مسئول این است که صفحه تازه تخصیص داده شده را با استفاده از دایرکتوری صفحه داده شده برای فرایند که از آن استفاده می کند با مپ کردن آن صفحه با آدرس مجازی بعدی موجود در دایرکتوری صفحه، قابل دسترسی کند. اگر این تابع شکست بخورد احتمالا به این معنی است که دایرکتوری صفحه در حال حاضر پر است.

در هردو حالت allocuvm موفق به افزایش حافظه کاربر به اندازه درخواست شده، نشده است، بنابراین تمام تخصیص ها را تا نقطه شکست برگردانده و حافظه مجازی تغییر نکرده و خودش یک خطا برمی گرداند.

**9) شیوه بارگذاری برنامه در حافظه توسط فراخوانی سیستمی exec را شرح دهید.**

تابع exec برای جایگزین کردن فرآیند در حال اجرا با یک فرایند جدید استفاده می شود:

مراحل:

پاک کردن حالت فعلی حافظه: این تابع ابتدا حالت حافظه فرایند فراخواننده را پاک می کند.

یافتن فایل برنامه: سپس به سیستم فایل می رود تا فایل برنامه درخواستی را پیدا کند.

کپی کردن فایل برنامه: سپس این فایل را در حافظه برنامه کپی می کند.

مقدار دهی وضعیت اولیه وضعیت رجیسترها از جمله PC که به دستور بعدی اشاره می کند.

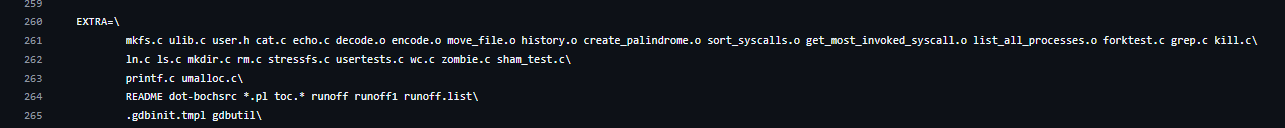
بارگذاری برنامه جدید: برنامه جدید در همان فضای آدرس، جایگزین قبلی می شود. این عمل همچنین به عنوان یک پوشش معرفی می شود.

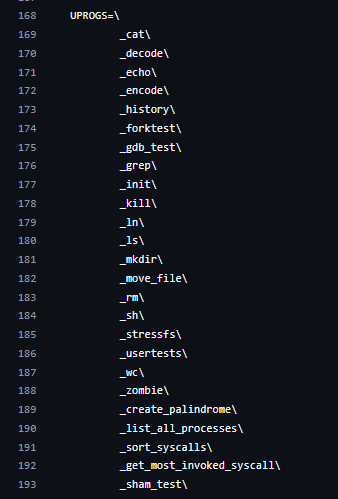
شناسه فرایند یا pid همان قبلی است: از آنجا که فرایند جدید ایجاد نمی شود، شناسه فرایند pid تغییر نمی کند. تابع exec مگر در صورت خطا ریترن نمی کند. این به این دلیل است که فرایند در حال اجرا کاملا توسط فرایند جدید جایگزین می شود.

**شرح پروژه**

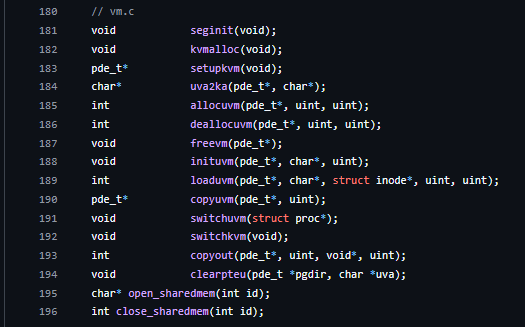
**توضیحات مربوط به پیاده سازی close shared memory و open shared memory**

Make File

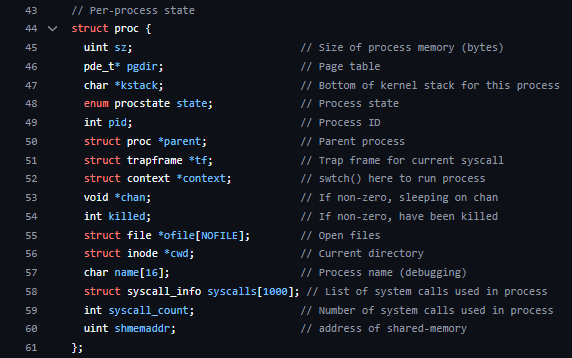




Defs.h

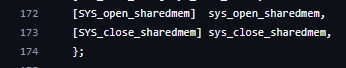


Proc.h



Syscall.c

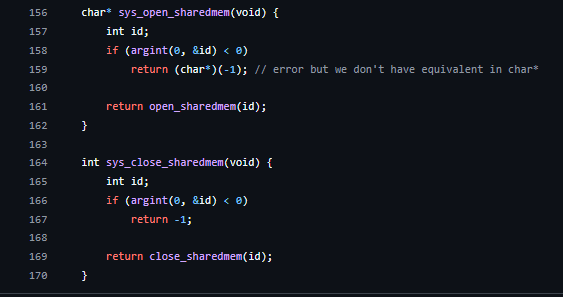




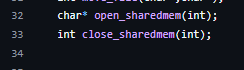
Syscall.h



Sysproc.c



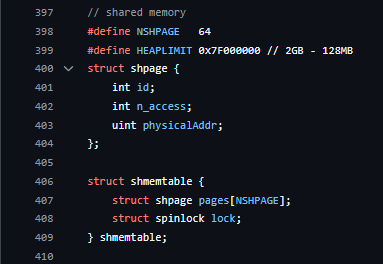
User.h



Usys.s



Vm.c

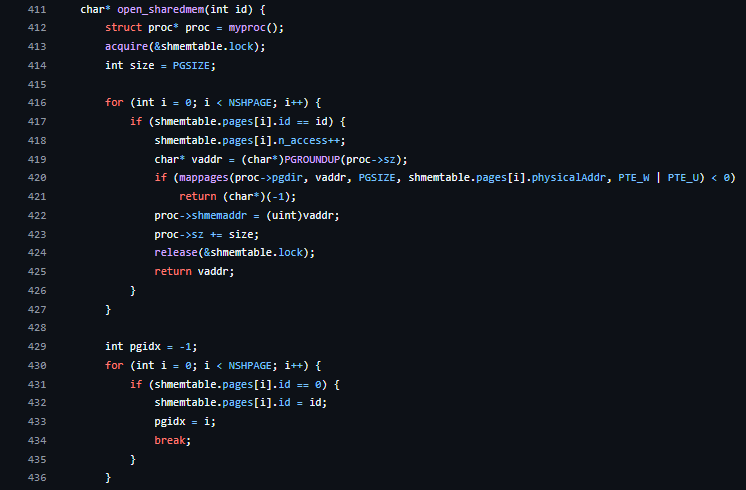


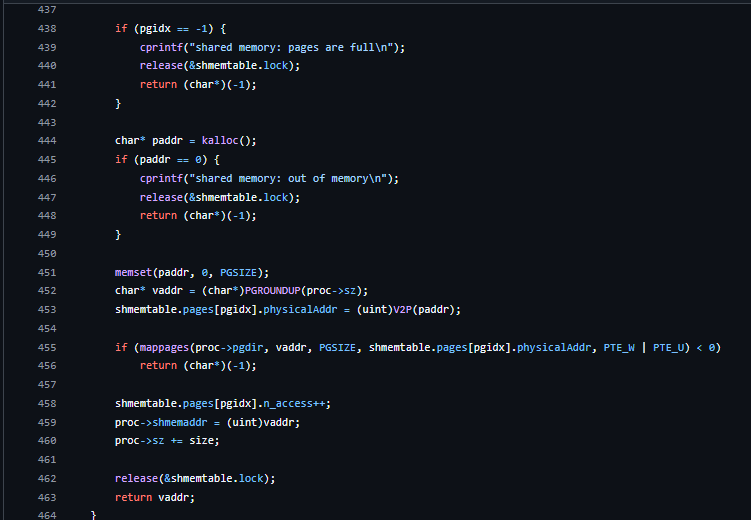
برای پیاده سازی حافظه اشتراکی 2 استراکت بالا را تعریف می کنیم .

یک استراکت برای نگهداری shared page که شامل شماره صفحه و تعداد دسترسی ها و آدرس فیزیکی صفحه می باشد.

و استراکتی دیگر جهت نگهداری shared memory table که شامل تعدادی صفحه و قفلی جهت نگهداری و دسترسی به table است .

**Open shared memory**

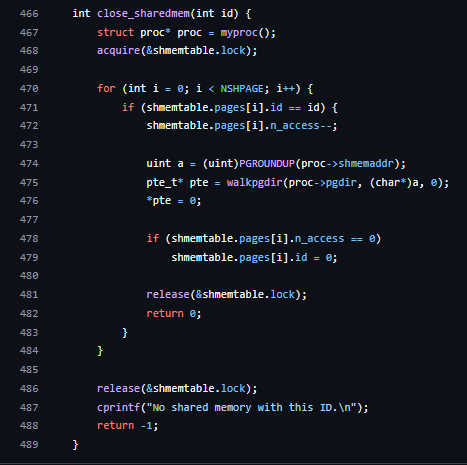




این تابع برای باز کردن یک بخش از حافظه مشترک است و دو حالت اصلی دارد:

1. **اگر حافظه مشترک با شناسه مشخص (id) وجود داشته باشد**:
   * حافظه پیدا شده و به فضای آدرس مجازی فرآیند فعلی متصل می‌شود.
   * تعداد دسترسی‌ها (n\_access) افزایش می‌یابد.
2. **اگر حافظه مشترک وجود نداشته باشد**:
   * یک صفحه جدید از حافظه فیزیکی با استفاده از kalloc تخصیص داده می‌شود.
   * حافظه به صفر مقداردهی شده و به فرآیند متصل می‌شود.
   * جدول حافظه مشترک با اطلاعات صفحه جدید به‌روزرسانی می‌شود.

**Close shared memory**



این تابع برای بستن یک بخش از حافظه مشترک است:

* ابتدا صفحه مرتبط با شناسه داده‌شده (id) را پیدا می‌کند.
* تعداد دسترسی‌ها (n\_access) کاهش می‌یابد.

اگر هیچ فرآیند دیگری از حافظه استفاده نکند (n\_access == 0)، شناسه حافظه پاک می‌شود و صفحه از جدول آزاد می‌شود.

**برنامه آزمون**

در بخش اخر ما فایلی برای تست کردن سیستم کال هایی که نوشتیم می نویسیم. فایل sham\_test.c

کد این فایل به صورت زیر است.



این فایل به این صورت است که یک متغیر مشترک در نظر می گیریم. سپس آن را با مقدار یک مقدار اولیه می دهیم. با توجه به اینکه فاکتوریل از ما خواسته شده بود، هرپردازه ای که می آید بسته به ایندکسی که دارد اجرا می شود(مثلا اینجا 5 تا پردازه داریم و 5 ایندکس) عددی را در مقدار مشترک قبلی ضرب کرده و همانجا ذخیره می کند.  
در هرگام نتیجه متغیر مشترک را چاپ می کنیم و به نتیجه زیر می رسیم.

همانطور که مشاهده می کنید نتیجه فاکتوریل ما درست است.

