به نام خدا

گزارش پروژه پنجم

امیرحسین علی خانی و صدرا مدائنی اول و هستی ابوالحسنی

(1

VMAبازه‌ای پیوسته از آدرس‌های مجازی است که همه ویژگی‌ها و دسترسی‌های یکسان (خواندن، نوشتن، اجرا) دارند. هسته لینوکس برای هر فرایند فهرستی ازVMA ها (در قالب یک لیست پیوندی و درخت قرمز–سیاه) نگه می‌دارد. هر بخش از برنامه مانند کد، داده، پشته، حافظه heap و فایل‌های map در VMA جداگانه‌ای قرار می‌گیرد؛ دسترسی به آدرسی خارج از این نواحی منجر به خطای Segmentation Fault می‌شود.

xv6حافظه کاربر را به‌صورت یک بلوک تقریباً پیوسته از آدرس 0 تا KERNBASE مدیریت می‌کند و ساختارVMA های جداگانه‌ای ندارد. در xv6 ترتیب ثابت متن–کد، داده و پشته وجود دارد و این بخش‌ها فقط با جداول صفحه تمیز می‌شوند. لینوکس می‌تواند آدرس‌های پراکنده با «حفره‌ها» داشته باشد و ویژگی‌های پیشرفتهmap ، کتابخانه‌های مشترک و نگهداریVMA های جداگانه را پشتیبانی می‌کند؛ در حالی که xv6 تمام کد و داده‌ٔ برنامه را به‌صورت خصوصی در حافظه‌اش بارگذاری می‌کند.

(2

صفحات محافظ برای حفاظت از مرزهای حافظه و جلوگیری از خطاهای سرریز پشته استفاده می‌شوند.در چیدمان حافظه نشان‌داده‌شده، یک صفحه محافظ بلافاصله زیر پشته کاربر قرار می‌گیرد. این صفحه عمداً بدون نگاشت (علامت‌گذاری به عنوان نامعتبر) می‌ماند. اگر پشته برنامه بیش از حد بزرگ شود و از اندازه اختصاص داده‌شده عبور کند، سعی می‌کند به این صفحه محافظ بنویسد. چون صفحه نامعتبر است، سخت‌افزار یک خطای صفحه (page fault) ایجاد می‌کند. این امکان به سیستم‌عامل می‌دهد تا سرریز را تشخیص داده و فرآیند را به‌صورت ایمن خاتمه دهد، به‌جای آنکه پشته به‌صورت ساکن داده یا کدهای مجاور را بازنویسی کند.

3)

یک ساختار صفحه‌بندی دو–سطحی حافظه را به‌صورت مؤثری در فضای آدرس‌های پراکنده مدیریت می‌کند.در یک جدول صفحه تک‌سطحی برای معماری ۳۲–بیتی با صفحه‌های ۴KB ، جدول باید 2^{20} ورودی داشته باشد. چون جدول باید به‌صورت پیوسته باشد، برای هر فرای ۴ MB RAM مصرف می‌شود، حتی اگر فرایند کوچک باشد. در ساختار دو–سطحی (دایرکتوری صفحه و جدول صفحه) فقط دایرکتوری بالایی) ۴ (KB در ابتدا تخصیص می‌شود. جداول صفحه ثانویه فقط برایواحی حافظه مجازی که فرایند واقعاً استفاده می‌کند، ساخته می‌شوند. چون بیشتر فرایندها تنها بخشی از فضای آدرس ۴ GB را استفاده می‌کنند و بین بخش‌های استک و کد/هیپ فواصل خالی دارند، سیستم‌عامل از تخصیص حافظه برای جدول‌های صفحه مربوط به این «حفره‌های» استفاده نشده جلوگیری می‌کند.

هر ورودی ۳۲–بیتی در دایرکتوری صفحه و جدول صفحه، ترجمه آدرس و دسترسی‌های مربوطه را کنترل می‌کند. بیت‌ها به‌صورت زیر تعریف شده‌اند:

بیت‌های ۱۲–۳۱ :(PPN) **شماره صفحه فیزیکی. به آدرس پایه ساختار بعدی (جدول صفحه یا فریم صفحه فیزیکی) اشاره می‌کند.**

بیت ۰ :(P – Present) **نشان می‌دهد صفحه در حافظه فیزیکی حضور دارد (۱ = بلی، ۰ = خیر).**

بیت ۱ :(W – Writable) **تعیین می‌کند صفحه قابل نوشتن است یا نه (۱ = خواندن/نوشتن، ۰ = فقط خواندنی).**

بیت ۲ :(U – User) **کنترل دسترسی حالت کاربری) ۱ = دسترس برای کاربر، ۰ = تنها** supervisor/kernal).

بیت ۳ :(WT – Write-Through) **تنظیم سیاست کش (۱ = نوشتن–از–راه، ۰ = نوشتن–به–عقب).**

بیت ۴ :(CD – Cache Disabled) **کش را برای این صفحه غیرفعال می‌کند.**

بیت ۵ :(A – Accessed) **توسط سخت‌افزار هنگام خوانده یا نوشته شدن صفحه تنظیم می‌شود.**

بیت ۶ :(D – Dirty) **توسط سخت‌افزار زمانی که صفحه نوشته می‌شود، تنظیم می‌شود.**

بیت‌های ۱۱–۹ :(AVL) **برای استفاده نرم‌افزار سیستم رزرو شده‌اند.**

بر اساس سند، ساختار بیت‌ها یکسان است به‌جز بیت .(Dirty) D **در ورودی جدول صفحه** (PTE) **بیت** D **نشان می‌دهد صفحه تغییر کرده (نوشتن شده) است؛ در ورودی دایرکتوری صفحه** (PDE) **این بیت معمولاً صفر است) برای صفحات ۴** KB **استفاده نمی‌شود.**

(4

تابع kalloc **حافظه فیزیکی را تخصیص می‌دهد.مسئول مدیریت** RAM **سیستم است و یک صفحه فیزیکی آزاد ۴** KB **را از فهرست آزاد هسته بر می‌دارد. اگرچه آدرسی که باز می‌گرداند یک آدرس مجازی هسته است تا هسته بتواند به آن دسترسی پیدا کند، منبع واقعی رزرو شده یک صفحهٔ حافظهٔ فیزیکی است که معمولاً با توابعی نظیر** mappages **به آدرس مجازی خاصی نگاشته می‌شود. این متفاوت از توابعی مثل** allocuvm **است که فقط فضای آدرسی مجازی را تنظیم می‌کنند.**

(5

تابع mappages **برای ایجاد نگاشت بین یک بازه خاص از آدرس‌های مجازی و آدرس‌های فیزیکی در داخل جدول صفحه استفاده می‌شود. ورودی‌های آن شامل دایرکتوری صفحه آدرس مجازی، اندازه حافظه مورد نظر برای نگاشت، آدرس فیزیکی متناظر و پرچم‌های دسترسی)مانند** Writable **یا** (User-accessible **است. این تابع قلمرو آدرس‌های مجازی را صفحه به صفحه مرور می‌کند؛ برای هر صفحه،** PTE**مربوطه را) در صورت نیاز با** walkpgdir **یک جدول صفحه جدید می‌سازد (پیدا می‌کند و آدرس فیزیکی و بیت‌های دسترسی را در آن می‌نویسد.این**

تابع به‌طور حیاتی در هنگام بوت سیستم (برای نگاشت هسته به حافظه) و در زمان ایجاد فرآیند (برای نگاشت کد، داده و پشته کاربر در فضای آدرس مجازی‌اش) به‌کار گرفته می‌شود.

(6

نه، همه آن‌ها قابل استفاده نیستند.فضای آدرس شامل یک صفحه محافظ (Guard Page) **است که دقیقاً زیر استک قرار دارد. این صفحه در بازه اختصاص داده شده به فرآیند وجود دارد ولی به‌عنوان نامنقشه علامت‌گذاری شده، لذا برنامه کاربر نمی‌تواند به آن دسترسی داشته باشد؛ دسترسی منجر به خطای** Page Fault **می‌شود.**

(7

تابع mappages **برای ایجاد نگاشت بین یک بازه خاص از آدرس‌های مجازی و آدرس‌های فیزیکی در داخل جدول صفحه استفاده می‌شود.ورودی‌های جدول صفحه را به‌گونه‌ای به‌روزرسانی می‌کند که صفحات مجازی به فریم‌های فیزیکی مشخص اشاره کنند.هسته برای نگاشت نواحی حافظهٔ خاص (مانند فضای آدرس هسته یا دستگاه‌ها) از این تابع بهره می‌برد.**

(8

تابع allocuvm **برای تخصیص حافظه فیزیکی و نگاشت آن در فضای آدرس مجازی کاربر مسئول است.صفحات فیزیکی جدید) معمولاً با** (kalloc **را تخصیص داده و آن‌ها را به جدول صفحه فرایند اضافه می‌کند تا اندازه حافظه معتبر آن افزایش یابد.به‌ویژه در فراخوانی** exec **برای اختصاص فضای لازم برای کد، داده و پشته برنامه جدید استفاده می‌شود و اطمینان می‌دهد که آدرس‌های مجازی مورد نیاز برنامه به حافظه فیزیکی واقعی پشتوانه دارند.**

(9

فراخوانی exec **برنامه جدیدی را داخل فرآیند موجود** (PCB) **اجرا می‌کند و کد، داده‌های ثابت، پشته و هیپ را در فضای آدرس فرآیند می‌سازد**.

1.**آغاز فضای آدرس هسته** exec :**ابتدا تابع** setupkvm **را صدا می‌زند تا بخش هسته‌ی فضای آدرس را تعیین و مقداردهی اولیه کند**.

2.**تخصیص کد و داده: سپس با** allocuvm **حافظهٔ لازم برای کد و داده‌های برنامه جدید تخصیص می‌یابد**.

3.**تخصیص پشته و صفحه محافظ** exec :**حافظه مورد نیاز برای پشته و یک صفحه** Guard Page **را اختصاص می‌دهد**.

4.**بارگذاری داده‌ها: پس از تخصیص صفحات حافظه،** exec**محتوای برنامه را از دیسک خوانده و در این فضاهای اختصاص–یافته می‌نویسد**.

(10

۱ .سیستم‌عامل لینوکس

لینکس عمدتاً از گونه‌ای از الگوریتم LRU **(کم‌ترین استفاده شده اخیراً) استفاده می‌کند که به صورت** LRU» **دو–لیست» پیاده‌سازی می‌شود تا** LRU **استاندارد را به‌صورت کارآمد تقریب بزند**.

دو لیست پیوندی برای صفحات دارد و لیست فعال (Active List) **و لیست غیرفعال**.(Inactive List)

-**لیست فعال: صفحاتی که در حال حاضر مورد استفاده و دسترسی مکرر هستند**.

-**لیست غیرفعال: صفحاتی که اخیراً دسترسی نداشته‌اند و کاندید حذف هستند**.

نحوه کار: وقتی صفحه‌ای دسترسی می‌شود، به لیست فعال منتقل می‌شود. با پر شدن حافظه، صفحات از لیست فعال به لیست غیرفعال جابجا می‌شوند. زمان آزادسازی حافظه، صفحات از

انتهای لیست غیرفعال برداشت می‌شوند.

این روش از «thrashing» **ناشی از اسکن یکباره (مثلاً خواندن یک پرونده بسیار بزرگ) جلوگیری می‌کند؛ در** LRU **ساده چنین خوانش بزرگی کش مفید را پاک می‌کرد، ولی در لینکس این صفحات به لیست غیرفعال می‌روند و سریع حذف می‌شوند بدون اینکه صفحات فعال را مختل کنند.**

سیستم‌عامل ویندوز

ویندوز از مدل Working Set **ترکیب‌شده با الگوریتم** Clock **یک نوع الگوریتم** Second-Chance **برای جایگزینی صفحات استفاده می‌کند. به هر فرایند یک** Working Set**اختصاص می‌دهد که حداقل و حداکثر حافظه فیزیکی مجاز به استفاده را تعریف می‌کند.**

- Working Set **وقتی حافظه آزاد سیستم زیر آستانه‌ای می‌افتد، رشته سیستمی** Balance Set Manager**بیدار می‌شود و** Working Set **های فرایندها را «قصد قطع» می‌کند.**

- Clock **برای تعیین صفحاتی که باید قطع شوند، سیستم به صورت دورانی (مانند ساعت) صفحات داخل** Working Set **را اسکن می‌کند و بیت** Accessed **ورودی جدول صفحه را بررسی می‌نماید. اگر بیت تنظیم شده باشد (صفحه استفاده شده)، بیت پاک شده و صفحه یک فرصت دوم می‌گیرد؛ اگر بیت پاک باشد (صفحه از آخرین اسکن استفاده نشده)، صفحه برای تعویض به فایل صفحه** (paging file)**واجد شرایط می‌شود.این روش تضمین می‌کند که فرایندهای بیکار حافظه فیزیکی خود را به فرایندهای فعال می‌دهند و پاسخگویی کلی سیستم بهبود می‌یابد.**