Specialized Scientific Quarterly Journal of

Arman Process (APJ)

Examining memory management approaches in modern operating systems

***A. Sadeghi*[*,1], *H. Hoseinzadeh*[*,1]**

[1] *Department of Electrical and Computer Engineering, North Tehran Branch, Islamic Azad University, Tehran, Iran*
[1] *Department of Electrical and Computer Engineering, North Tehran Branch, Islamic Azad University, Tehran, Iran*

ABSTRACT

Memory management is one of the critical and major functions of the modern operating systems, that manages the main memory operations and shifts processes between main memory and disk during the application execution. Memory management manages them all, whether or not different memory locations are allocated to a process. This process determines the amount of memory that must be allocated to operating system processes. Memory management approches decide which process to access memory at a time and track the amount of memory allocated or free up and update the corresponding states. In this study, we have examined the major dimensions of the memory management issue from the perspective of the operating system administrators, the validation of quality requirements and also how to use the quality dimensions in the memory management of the modern operating systems, due to its great importance in this field for ensuring efficiency and performance.

KEYWORDS:

Memory management
Modern Operating Systems
Paging
Segmentation
Fragmentation

Corresponding author
A.sadeghi56@gmail.com

NUMBER OF REFERENCES	NUMBER OF FIGURES	NUMBER OF TABLES
11	3	3

فصلنامه تخصصی

آرمان پردازش

بررسی رویکردهای مدیریت حافظه در سیستم های عامل

امیر صادقی*[۱] و حمید حسین زاده [۲]

۱ گروه کامپیوتر، دانشکده برق و کامپیوتر، واحد تهران شمال، دانشگاه آزاد اسلامی، تهران، ایران

۲ گروه کامپیوتر، دانشکده برق و کامپیوتر، واحد تهران شمال، دانشگاه آزاد اسلامی، تهران، ایران

چکیده

مدیریت حافظه یکی از وظایف مهم و اصلی سیستم عامل های مدرن است که عملیات اصلی حافظه را مدیریت کرده و فرآیندها را بین حافظه اصلی و دیسک در حین اجرای برنامه تغییر می دهد. مدیریت حافظه همه فعالیت های سیستم عامل را مدیریت می کند، خواه محل های مختلف حافظه به یک فرآیند اختصاص داده شود یا خیر. این فرایند مقدار حافظه ای را که باید به فرآیندهای سیستم عامل اختصاص داده شود، تعیین می کند. مدیریت حافظه تصمیم می گیرد که کدام فرایند برای دسترسی به حافظه در یک زمان و پیگیری میزان تخصیص داده شده یا آزاد شدن و به روز رسانی حالت های مربوطه باشد. در این مطالعه، ما ابعاد مهم مساله مدیریت حافظه را از دیدگاه مدیران سیستم عامل، اعتبار سنجی الزامات کیفیت و همچنین نحوه استفاده از ابعاد کیفی در مدیریت حافظه سیستم عامل های امروزی، به دلیل مهم بودن چالش ها و پراهمیت بودن مساله، بررسی کرده ایم. اهمیت در این زمینه برای اطمینان از کارایی و عملکرد سیستم عامل ها ضروری می باشد.

واژگان کلیدی:
مدیریت حافظه
سیستم عامل های مدرن
صفحه بندی
قطعه بندی
فرگمنت کردن

[۱]نویسنده مسئول

A.sadeghi56@gmail.com

تعداد جداول	تعداد شکل ها	تعداد مراجع
۳	۳	۱۱

مقدمه

سیستم عامل درواقع برنامه سیستمی است که مدیریت منابع رایانه را به عهده گرفته و بستری را فراهم می‌سازد که نرم‌افزارهای کاربردی اجرا شده و از خدمات آن استفاده کنند. سیستم‌عامل جزء ضروری‌ترین نرم‌افزارهای یک سیستم رایانه‌ای است. سیستم‌عامل خدماتی به برنامه‌های کاربردی و کاربر ارائه می‌دهد. برنامه‌های کاربردی یا از طریق واسط‌های برنامه‌نویسی کاربردی یا از طریق فراخوانی‌های سیستم به این خدمات دسترسی دارند. با فراخوانی این واسط‌ها، برنامه‌های کاربردی می‌توانند سرویسی را از سیستم‌عامل درخواست کنند، پارامترها را انتقال دهند و پاسخ عملیات را دریافت کنند. ممکن است کاربران با بعضی انواع واسط کاربری نرم‌افزار مثل واسط خط فرمان یک واسط گرافیکی کاربر با سیستم‌عامل تعامل کنند. برای رایانه‌های دستی و رومیزی، عموماً واسط کاربری به عنوان بخشی از سیستم‌عامل در نظر گرفته می‌شود. در سیستم‌های بزرگ و چند کاربره مثل یونیکس و سیستم‌های شبیه یونیکس، واسط کاربری معمولاً به عنوان یک برنامه کاربردی که خارج از سیستم‌عامل اجرا می‌شود پیاده‌سازی می‌شود [1]. نمونه‌هایی از محبوب‌ترین سیستم‌عامل‌های مدرن شامل: اندروید، ویندوز، آی او اس، مک اواس، لینوکس، کروم اواس، ویندوز فون، بی‌اس‌دی، کیوان‌اکس، و زداواس می‌باشند. مدیریت حافظه یکی از کارکردهای سیستم عامل است که به مدیریت حافظه اصلی و جابجایی پردازش‌ها بین حافظه اصلی و دیسک در طی اجرا می‌پردازد. مدیریت حافظه صرف نظر از این که موقعیت‌های مختلف حافظه به پردازشی تخصیص یافته‌اند یا نه، همه آن‌ها را مدیریت می‌کند. این فرایند میزان حافظه‌ای که باید به پردازش‌ها تخصیص یابد را تعیین می‌کند. مدیریت حافظه تصمیم می‌گیرد که در هر زمان کدام پردازش به حافظه دسترسی داشته باشد. مدیریت حافظه به ردگیری مقدار حافظه تخصیص یافته یا آزاد شده پرداخته و وضعیت‌های متناظر با آن‌ها را به‌روزرسانی می‌کند. مدیریت حافظه یکی از کارکردهای سیستم عامل است که به مدیریت حافظه اصلی و جابجایی پردازش‌ها بین حافظه اصلی و دیسک در طی اجرا می‌پردازد. مدیریت حافظه تصمیم می‌گیرد که در هر زمان کدام پردازش به حافظه دسترسی داشته باشد. مدیریت حافظه به ردگیری مقدار حافظه تخصیص یافته یا آزاد شده پرداخته و وضعیت‌های متناظر با آن‌ها را به‌روزرسانی می‌کند. فضای آدرس پردازش به مجموعه‌ای از آدرس‌های منطقی گفته می‌شود که یک پردازش در کد خود مورد ارجاع قرار می‌دهد. سیستم عامل وظیفه نگاشت آدرس‌های منطقی به آدرس‌های فیزیکی را در زمان تخصیص حافظه بر عهده دارد. سه نوع از آدرس پیش و پس از تخصیص حافظه در برنامه های سیستم عامل به شرح جدول زیر مورد استفاده قرار می‌گیرند:

جدول ۱. انواع آدرس دهی های حافظه

نوع آدرس حافظه	توضیح آن
آدرس های نمادین	آدرس‌های مورد استفاده در کد سورس. نام‌های متغیرها، ثابت‌ها و برچسب‌های دستورالعمل اجزای اصلی فضای آدرس‌دهی نمادین هستند.
آدرس‌های نسبی	کامپایلر در زمان کامپایل، آدرس‌های نمادین را به آدرس‌های نسبی تبدیل می‌کند.
آدرس‌های فیزیکی	بخش بارگذار این آدرس‌ها را در زمان بارگذاری برنامه در حافظه اصلی تولید می‌کند.

آدرس‌های مجازی و فیزیکی در زمان کامپایل یکسان هستند و در زمان بارگذاری طرح‌های متصل به آدرس ایجادی را تشکیل می‌دهند. آدرس‌های مجازی و فیزیکی در زمان اجرا طرح‌های اتصال آدرس متفاوتی دارند. مجموعه همه آدرس‌های منطقی تولید شده از سوی یک برنامه به نام فضای آدرس منطقی نامیده می‌شود. مجموعه همه آدرس‌های فیزیکی متناظر با این آدرس‌های منطقی به نام فضای آدرس فیزیکی شناخته می‌شود. نگاشت زمان اجرا از آدرس‌های مجازی به فیزیکی به وسیله واحد مدیریت حافظه (MMU) صورت می‌گیرد که اساسا قطعه‌ای با جنسیت مغناطیسی سخت‌افزاری است.

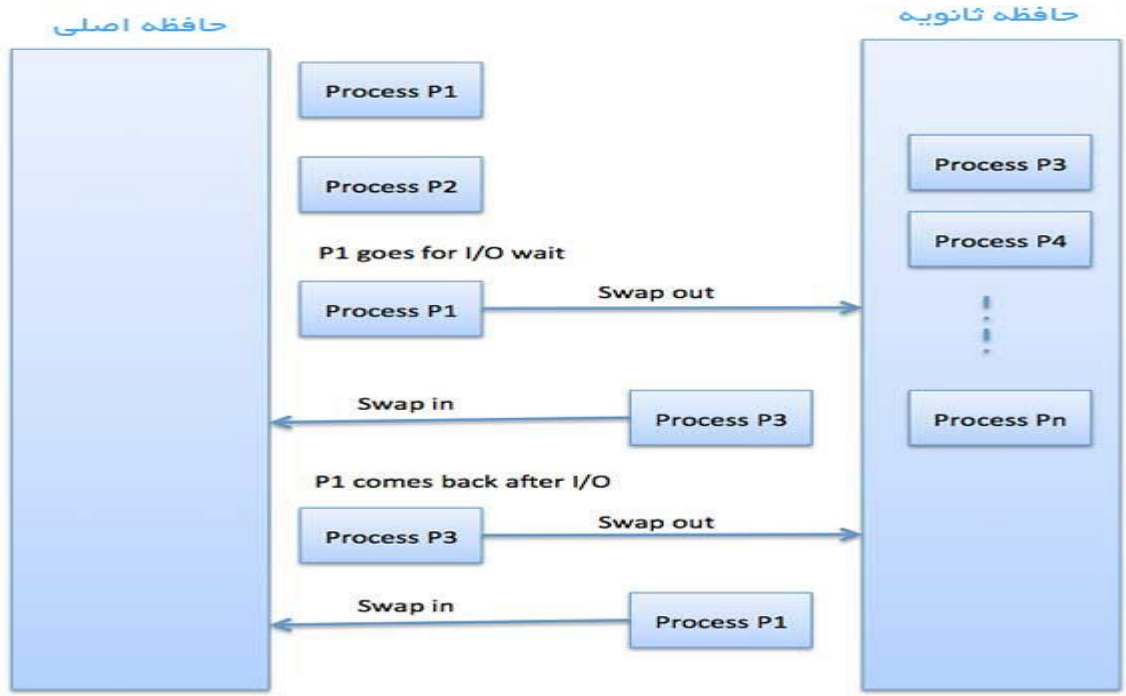
مقدار موجود در ثبات پایه به همه آدرس‌های تولید شده از سوی پردازش کاربر اضافه می‌شود و به عنوان یک افست در هنگام ارسال به حافظه عمل می‌کند. برای نمونه اگر مقدار ثبات پایه برابر با ۱۰۰۰۰ باشد، در این صورت هنگامی که کاربر تلاش می‌کند از ۱۰۰ موقعیت حافظه استفاده کند، مکان‌های حافظه مورد استفاده وی به صورت دینامیک تا موقعیت ۱۰۱۰۰

تخصیص می‌یابند. برنامه کاربر با آدرس‌های مجازی سر و کار دارد و هرگز آدرس‌های فیزیکی واقعی را نمی‌بیند.

بارگذاری ایستا[1] و پویا[2]

انتخاب بین بارگذاری استاتیک و دینامیک در زمان نوشتن برنامه رایانه‌ای صورت می‌گیرد. اگر می‌خواهید برنامه را به صورت استاتیک بارگذاری کنید، در این صورت در زمان کامپایل، برنامه‌ها بدون برجای گذاشتن هیچ گونه برنامه بیرونی یا وابستگی ماژول، به صورت کامل کامپایل و لینک می‌شوند. در این حالت لینکر، برنامه object را با دیگر ماژول‌های object ضروری در قالب یک برنامه کامل کامپایل می‌کند که می‌تواند شامل آدرس‌های منطقی نیز باشد. اگر مشغول بارگذاری دینامیک یک برنامه باشید، در این صورت کامپایلر برنامه را کامپایل خواهد کرد و برای همه ماژول‌هایی که می‌خواهید به صورت دینامیک بارگذاری شوند، تنها ارجاعی ارائه می‌شود و باقی کارها در زمان اجرا صورت می‌گیرند. در زمان بارگذاری به صورت بارگذاری استاتیک، برنامه کامل و داده‌ها در حافظه بارگذاری می‌شوند تا اجرای برنامه آغاز شود. اگر از بارگذاری دینامیک استفاده می‌کنید، رویه‌های دینامیک کتابخانه روی یک دیسک به صورت قابل بازیابی ذخیره می‌شوند و تنها زمانی که از سوی برنامه مورد نیاز باشند در حافظه بارگذاری می‌شوند.

همان طور که در بخش قبلی اشاره کردیم، وقتی از روش لینک کردن استاتیک استفاده کنیم، لینکر همه ماژول‌های دیگر مورد نیاز از سوی برنامه را در قالب برنامه اجرایی منفردی ترکیب می‌کند تا از وابستگی‌های زمان اجرا جلوگیری کند. زمانی که از روش لینک کردن دینامیک استفاده می‌شود، نیاز نیست ماژول واقعی یا کتابخانه به برنامه لینک شود و به جای آن ارجاعی به ماژول دینامیک در زمان کامپایل و لینک کردن ارائه می‌شود. کتابخانه‌های لینک دینامیک (DLL) در ویندوز و اشیای مشترک (Shared Objects) در یونیکس، نمونه‌های خوبی از کتابخانه‌های دینامیک محسوب می‌شوند. سواپ کردن سازوکاری است که در آن یک پردازش می‌تواند به طور موقت از حافظه خارج شود و به حافظه ثانویه (دیسک) منتقل شود. بدین ترتیب این حافظه برای پردازش‌های دیگر باز می‌شود. متعاقب آن پس از مدتی سیستم، پردازش را از حافظه ثانویه به حافظه اصلی وارد می‌کند. با این که عملکرد سیستم معمولاً با سواپ کردن پردازش‌ها تحت تأثیر قرار می‌گیرد؛ اما این فرایند به اجرای پردازش‌های چندگانه و بزرگ به طور موازی کمک می‌کند و به همین دلیل سواپ کردن تحت عنوان تکنیک فشرده‌سازی حافظه[3] نیز نامیده می‌شود.



شکل ۱. ارتباط بین اجزاء حافظه اولیه و ثانویه

[1] Static
[2] Dynamic
[3] Memory compaction

لازم به ذکر است، کل زمان مورد نیاز برای سواپ کردن پردازش، شامل زمانی است که برای انتقال کل پردازش به دیسک ثانویه و سپس کپی کردن مجدد آن به حافظه و همچنین زمان دریافت حافظه اصلی از سوی پردازش می‌شود. در این حوزه مهم ترین روش های مدیریت حافظه به شرح زیر می باشد:

- بخش بندی ایستا: حافظه اصلی به تعدادی بخش ایستا در زمان ایجاد سیستم تقسیم می شود؛ فرآیند به داخل بخشی با اندازه برابر یا بزرگتر برود.
- بخش بندی پویا: بخش ها به صورت پویا ایجاد می شوند؛ هر فرآیند به داخل بخشی برابر با اندازه خودش می رود.
- صفحه بندی ساده: حافظه به قاب هایی هم اندازه تقسیم می شود و فرآیند به صفحات هم اندازه با قاب ها، تقسیم می شوند.
- قطعه بندی ساده: هر فرآیند به قطعه هایی تقسیم می شود و از طریق باز کردن تمام قطعاتش اجرا می شود که لزوماً پیوسته نیستند.
- صفحه بندی حافظه مجازی: مانند صفحه بندی ساده است ولی نیاز نیست که تمام صفحات یک فرآیند باز شوند.
- قطعه بندی حافظه مجازی: مانند قطعه بندی ساده است ولی نیاز نیست که تمام صفحات یک فرآیند باز شوند.

در اغلب رایانه‌ها، می‌توان ظرفیت حافظه را ارتقاء و افزایش داد. مثلاً می‌توان میزان حافظه RAM موجود را از یک مگابایت به دو مگابایت ارتقاء داد. روش فوق یک راهکار فیزیکی برای افزایش حافظه بوده که در برخی موارد دارای چالش‌های خاص خود می‌باشد. در این زمینه می‌بایست راهکارهای دیگر نیز مورد بررسی قرار گیرند . اغلب اطلاعات ذخیره شده توسط برنامه‌ها در حافظه، در تمام لحظات مورد نیاز نخواهد نبود. پردازنده در هر لحظه قادر به دستیابی به یک محل خاص از حافظه‌است. بنابراین اکثر حجم حافظه در اغلب اوقات غیر قابل استفاده‌است. از طرف دیگر با توجه به اینکه فضای ذخیره‌سازی حافظه‌ها ی جانبی نظیر دیسک‌ها بمراتب ارزانتر نسبت به حافظه اصلی است، می‌توان با استفاده از مکانیزم‌هایی اطلاعات موجود در حافظه اصلی را خارج و آن‌ها را موقتاً بر روی هارد دیسک ذخیره نمود. بدین ترتیب فضای حافظه اصلی آزاد و در زمانی‌که به اطلاعات ذخیره شده بر روی هارد دیسک نیاز باشد، مجدداً آن‌ها را در حافظه مستقر کرد. روش فوق در سیستم عامل های مدرن «مدیریت حافظه مجازی» نامیده می‌شود.

تخصیص حافظه

حافظه اصلی معمولاً دو پارتیشن دارد: اول حافظه پایین که پردازش‌های سیستم عامل در این حافظه هستند. دوم حافظه بالا که پردازش‌های کاربر در حافظه بالا نگهداری می‌شوند. سیستم عامل از ساز و کار تخصیص حافظه بصورت زیر استفاده می‌کند.

جدول۲. مقایسه رویکردهای تخصیص پارتیشنی در مدیریت حافظ

روش تخصیص حافظه	توضیح
تخصیص تک پارتیشنی	در این نوع از تخصیص، از طرح‌بندی relocation-register برای حافظت از پردازش‌های کاربر در برابر همدیگر استفاده می‌شود تا بدین ترتیب کد و داده‌های سیستم عامل تغییری پیدا نکند. ثبات relocation شامل کوچک‌ترین مقدار آدرس فیزیکی است؛ در حالی که ثبات imit شامل بازه‌ای از آدرس‌های منطقی است. هر آدرس منطقی باید مقداری کمتر از ثبات limit داشته باشد.
تخصیص چند پارتیشنی	در این نوع از تخصیص، حافظه اصلی به چند پارتیشن با طول ثابت تقسیم می‌شود که در آن هر پارتیشن شامل تنها یک پردازش است. وقتی یک پارتیشن آزاد می‌شود، پردازشی از صف ورودی انتخاب شده و درون پارتیشن آزاد بارگذاری می‌شود. وقتی یک پردازش خاتمه می‌یابد، پارتیشن در اختیار پردازش دیگری قرار می‌گیرد.

فرگمنت کردن[۴]

وقتی پردازش‌ها وارد حافظه شده و از آن خارج می‌شوند، فضای آزاد حافظه به قطعه‌های کوچکی تقسیم می‌شود. این وضعت پس از مدتی منتهی به این نتیجه می‌شود که پردازش‌ها نمی‌توانند در بلوک‌های حافظه تخصیص یابند، زیرا اندازه بلوک‌های حافظه کوچک و تکه تکه شده و در واقع بی‌استفاده می‌مانند. این مسئله به نام فرگمنت شدن[۵] نامیده می‌شود. اساسا در سیستم عامل ها دو نوع فرگمنت شدن به شرح زیر وجود دارد:

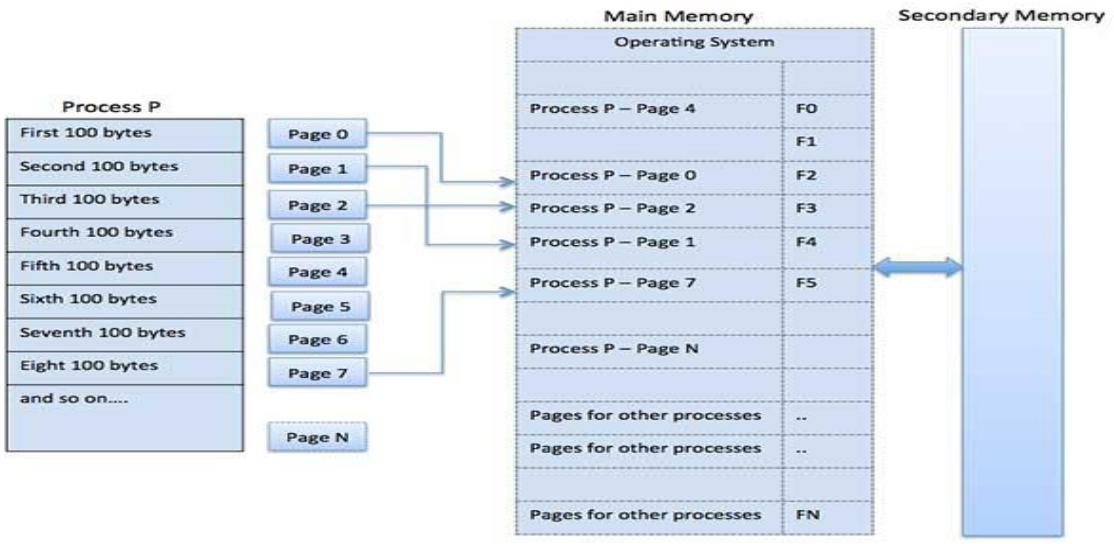
جدول۳. انواع فرگمنت کردن حافظه

نوع فرگمنت شدن	توضیح
فرگمنت شدن بیرونی	در این حالت فضای کلی حافظه برای تأمین یک درخواست یا جایگیری یک پردازش کافی است؛ اما از آن جا که فضای حافظه پیوسته نیست، نمی‌تواند بدین منظور مورد استفاده قرار گیرد.
فرگمنت شدن درونی	در این حالت، بلوک‌های حافظه تخصیص یافته به یک پردازش، بزرگ‌تر است. در این حالت بخشی از حافظه بی‌استفاده مانده است و از این رو نمی‌تواند از سوی پردازش‌های دیگر استفاده شود.

در نمودار زیر نشان داده شده است که چگونه فرگمنت شدن می‌تواند منجر به اتلاف حافظه شود. فرگمنت شدن بیرونی را میتوان به وسیله روش های فشرده سازی یا درهم سازی محتوا برای جای دادن همه استفاده کرد.

صفحه‌بندی[۶]

یک رایانه می‌تواند مقدار حافظه‌ای بیشتر از آن چه روی آن نصب شده است را آدرس‌دهی کند. حافظه بیشتری که در واقع حافظه مجازی نامیده می‌شود بخشی از هارددیسک است که برای تقلید RAMرایانه اختصاص یافته است. تکنیک صفحه‌بندی نقش مهمی در پیاده‌سازی حافظه مجازی دارد. صفحه‌بندی تکنیک مدیریت حافظه‌ای است که در آن فضای آدرس پردازش به بلوک‌هایی با اندازه یکسان به نام صفحه (page) تقسیم می‌شود. اندازه این صفحه‌ها از توان ۲ است و بین ۵۱۲ بایت و ۸۱۹۲ بایت است. اندازه پردازش بر اساس تعداد صفحه‌ها اندازه‌گیری می‌شود. به طور مشابه حافظه اصلی به بلوک‌های با اندازه ثابت کوچکی از حافظه فیزیکی تقسیم می‌شود که فریم (frame) نامیده می‌شود و اندازه یک فریم به همان اندازه یک صفحه حفظ می‌شود تا کارکرد بهینه‌ای از حافظه اصلی داشته باشد و از فرگمنت شدن بیرونی اجتناب شود.



شکل۲. تکنیک صفحه بندی در مدیریت حافظه

۴ Fragmentation

5 Fragmentation

۶ Paging

فرایند ترجمه آدرس

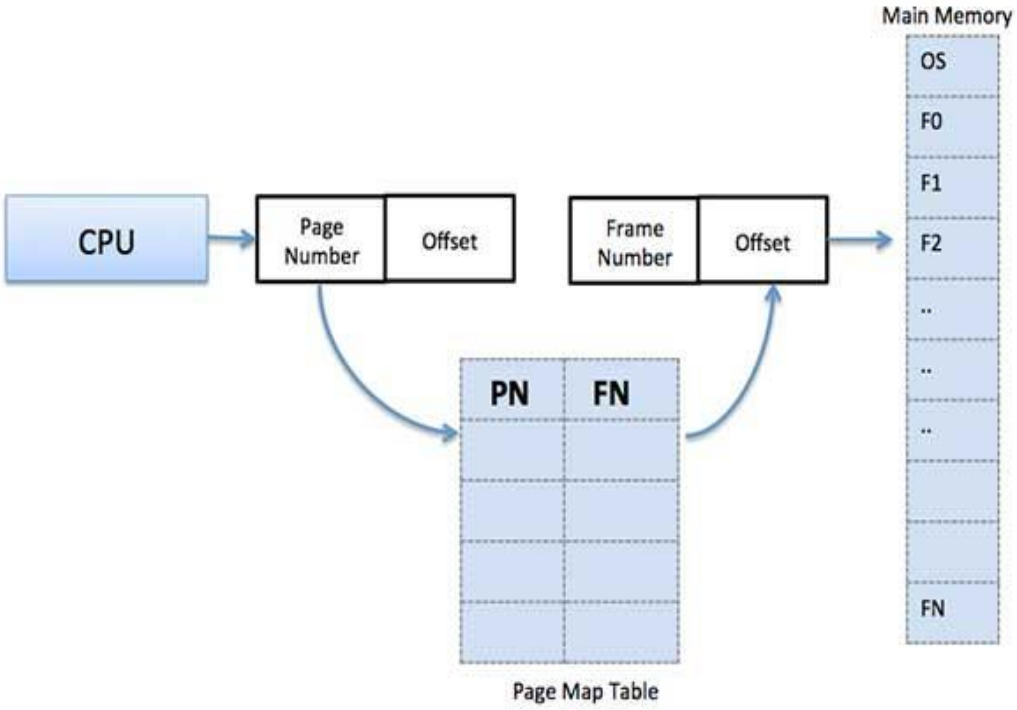
آدرس صفحه به نام آدرس منطقی نامیده می‌شود و با شماره صفحه (page number) و افست (offset) نمایش می‌یابد.

Logical Address = Page number + page offset

آدرس فریم به نام آدرس فیزیکی نامیده می‌شود و به وسیله شماره فریم (frame number) و افست مشخص می‌شود.

Physical Address = Frame number + page

ساختمان داده‌ای که جدول نگاشت صفحه (page map table) نامیده می‌شود، نیز برای ردگیری رابطه بین یک صفحه از پردازش به یک فریم در حافظه فیزیکی مورد استفاده قرار می‌گیرد.



شکل۳. تکنیک صفحه بندی در مدیریت حافظه

وقتی سیستم یک فریم را به صفحه‌ای تخصیص می‌دهد، آدرس منطقی‌اش را به آدرس فیزیکی ترجمه می‌کند تا مدخلی در جدول صفحه ایجاد کند که در سراسر زمان اجرای برنامه مورد استفاده قرار خواهد گرفت. وقتی یک پردازش قرار است اجرا شود، صفحه‌های متناظر آن در فریم‌های موجود حافظه بارگذاری می‌شوند. فرض کنید برنامه‌ای با حجم ۸ کیلوبایت داشته باشید؛ اما حافظه شما در هر زمان می‌تواند صرفاً ۵ کیلوبایت تخصیص دهد، در این صورت مفهوم صفحه‌بندی مورداستفاده قرار می‌گیرد. زمانی که مقدار RAM رایانه پایان می‌یابد، سیستم عامل صفحه‌های بیکار یا ناخواسته حافظه را به روی دیسک منتقل می‌کند تا RAM برای پردازش‌های دیگر آزاد شود و آن‌ها را مجدداً در زمان نیاز از سوی برنامه گرد هم می‌آورد این فرایند در طی کل زمان اجرای برنامه که سیستم عامل صفحه‌های بیکار را از حافظه اصلی خارج می‌کند و آن‌ها را در حافظه ثانویه می‌نویسد تداوم می‌یابد. در ادامه مقاله حاضر فهرستی از مزایا و معایب تکنیک صفحه‌بندی را بررسی می‌کنیم:

- صفحه‌بندی، فرگمنت شدن بیرونی را کاهش می‌دهد، اما همچنان از فرگمنت شدن درونی رنج می‌برد.
- صفحه‌بندی، پیاده‌سازی آسانی دارد و به عنوان روش کارآمدی برای مدیریت حافظه نگریسته می‌شود.
- به دلیل اندازه یکسان صفحه‌ها و فریم‌ها، سواپ کردن بسیار آسان است.
- جدول صفحه، نیازمند فضای حافظه مازادی است و از این رو ممکن است برای سیستمی که RAM کوچکی دارد مناسب نباشد.

سگمنت کردن[۷]

سگمنت کردن یک تکنیک مدیریت حافظه است که در آن هر وظیفه به چند سگمنت با اندازه‌های مختلف تقسیم می‌شود. یک سگمنت برای هر ماژول تشکیل می‌شود که شامل تکه‌هایی برای اجرای کارکردهای مرتبط است. هر سگمنت در عمل یک فضای آدرس منطقی متفاوت از برنامه است.وقتی یک پردازش قرار است اجرا شود، سگمنت‌های متناظر آن درون حافظه غیر پیوسته بارگذاری می‌شود؛ اما هر سگمنت در بلوک پیوسته‌ای از حافظه موجود بارگذاری می‌شود. مدیریت حافظه سگمنتیشن در فرایند کاملاً شبیه صفحه‌بندی عمل می‌کند؛ اما در این روش سگمنت‌ها دارای طول متغیر هستند؛ در حالی که در روش صفحه‌بندی، صفحه‌ها اندازه ثابتی دارند. یک سگمنت برنامه شامل تابع اصلی برنامه، تابع‌های کاربردی، ساختمان‌های داده و مواردی از این دست است. سیستم عامل یک جدول نگاشت سگمنت برای هر پردازش نگهداری می‌کند و لیستی از بلوک‌های حافظه آزاد را همراه با عددهای سگمنت، اندازه آن‌ها و مکان‌های متناظر حافظه اصلی در آن قرار می‌دهد. جدول در هر سگمنت آدرس‌های آغازین سگمنت و طول سگمنت را نگهداری می‌کند. یک ارجاع به موقعیت حافظه شامل مقداری است که یک سگمنت و یک افست را تعیین می کند. مهمترین نیازهای مدیریت حافظه شامل موارد زیر می باشد:

- جابجایی: سیستم، برای این منظور، باید از محل فرآیندها آگاهی داشته باشد و آدرس آن ها را ذخیره کند.
- حفاظت: باید در مقابل تداخل های ناخواسته حفاظت شود؛ خواه تصادفی باشد یا عمدی.
- اشتراک: باید با داشتن حفاظت به گونه ای باشد تا دیگران نیز بتوانند به آن دستیابی داشته باشند.
- سازمان منطقی: حافظه به صورت فضای آدرس خطی یا یک بعدی سازمان یافته است و شامل دنباله ای از بایت ها و کلمه ها می باشد.
- سازمان فیزیکی: حافظه به دو صورت *اصلی* که ناپایدار، سریع، گران و *مجازی* که پایدار و نسبتاً ارزان می باشد و مدیریت بین این دو است.

مهمترین مزایای سازماندهی منطقی به شرح زیر می باشد:

- هر مؤلفه را می توان به صورت مستقل نوشت و ترجمه کرد.
- با یک سر بار اضافی مراتب مختلف حفاظتی را داریم.
- امکان معرفی راهکارهایی برای اشتراک مؤلفه ها در بین فرایندها وجود دارد.

لازم به ذکر است، محققین IBM نشان داده اند که سیستم های جایگذاری حافظه مجازی نسبت به سیستم های دستی قدیمی عملکرد بهتری دارند. تا این لحظه همچنان این موضوع مورد بحث است و در مورد آن شک و تردید وجود دارد. مین فریم ها (بزرگ رایانه ها) و مینی کامپیوترها عموماً از حافظه مجازی استفاده می کنند. تکنولوژی حافظه مجازی در کامپیوترهای شخصی ابتدایی به کار گرفته نشده است زیرا توسعه دهندگان بر این باور بودند که تمام شدن حافظه نمی تواند مشکلی برای این کامپیوترها ایجاد کند. البته در ادامه پیشرفت سیستم عامل های مدرن نادرست بودن این فرضیه اثبات شد.

نتیجه گیری

سرعت بالای توسعه تکنولوژی‌های جدید و ارتباطات دیجیتال منجر به افزایش اهمیت مقوله مدیریت حافظه به‌عنوان یک منبع حیاتی برای کسب مزیت رقابتی سیستم عامل ها گردیده است. سیستم عامل های مدرن باید با مقایسه بین وضعیت کنونی و گذشته خود سعی در بهبود ارائه راهکارهای مدیریت حافظه و خدمات سیستم عامل بوده و تلاش کنند تا سیستم عامل ها فرایندهای درونی خود را به نحو اثربخشی انجام دهند. در این مطالعه، ما ابعاد عمده مساله مدیریت حافظه را از دیدگاه مدیران سیستم عامل، اعتبار سنجی الزامات کیفیت و همچنین نحوه استفاده از ابعاد کیفی در مدیریت حافظه سیستم عامل های مدرن، به دلیل مهم بودن چالش ها و پراهمیت بودن مساله، بررسی کرده ایم. اهمیت در این زمینه برای اطمینان از کارایی و عملکرد سیستم عامل ها ضروری می باشد.

مراجع

[1] 4. Silberchatz G, Galvin G, Gagne PB (2003) Operating systems concepts 4.

[2] Cudré-Mauroux, P., Wu, E., Madden, S.: Trajstore: an adaptive storage system for very large trajectory data sets. In: ICDE, pp. 109–120 (2010) 17.

[۷] Segmentation

International Symposium on Field Programmable Gate Arrays (FPGA), February 2011 (ACM, New York, 2011), pp. 25–28.

[8] Agne, M. Platzner, E. Lübbers, Memory virtualization for multi threaded reconfigurable hardware, in Proceedings of the International Conference on Programmable Logic and Applications (FPL), September 2011 (IEEE Computer Society, Los Alamitos, 2011), pp. 185–188.

[9] Agne, M. Happe, E. Lübbers, B. Plattner, M. Platzner, C. Plessl, ReconOS – an operating system approach for reconfigurable computing. IEEE Micro 34(1), 60–71 (2014).

[10] Tanenbaum A (2016) Modern operating systems. Pearson. ISBN-10: 93325.

[11] Stallings W (2017) Operating systems, internals and design principles, 9th edn. Pearson. ISBN 10: 0-13-380591-3.

[3] Xie, D., Li, F., Yao, B., Li, G., Zhou, L., Guo, M.: Simba: efficient in-memory spatial analytics. In: SIGMOD, pp. 1071–1085 (2016) 20.

[4] Yu, J., Wu, J., Sarwat, M.: Geospark: a cluster computing framework for processing large-scale spatial data. In: SIGSPATIAL, pp. 70:1–70:4 (2015).

[5] Baddeley, A. (1997). *Human memory: Theory and practice*. Hove: Psychology Press.

[6] Muja, M., & Lowe, D. G. (2009). Fast approximate nearest neighbors with automatic algorithm configuration. In *Proceedings of international conference on computer vision theory and application* (pp. 331–340).

[7] M. Adler, K. Fleming, A. Parashar, M. Pellauer, J. Emer, LEAP scratchpads: automatic memory and cache management for reconfigurable logic, in *Proceedings of the 19th ACM/SIGDA*