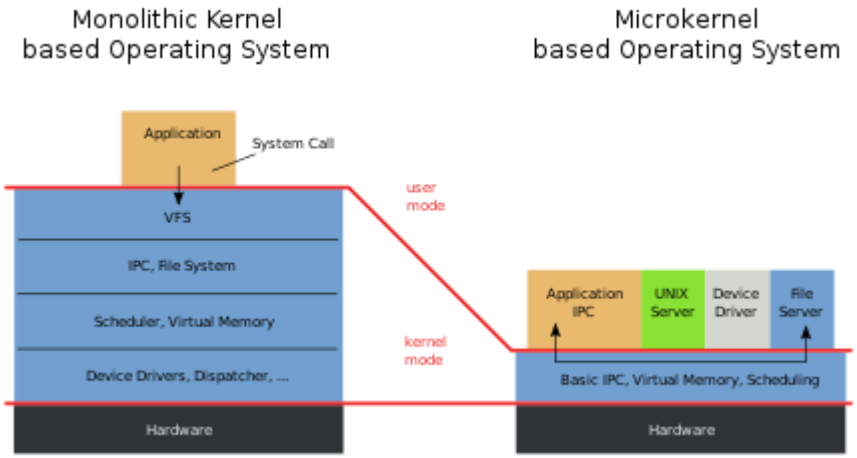
ریزهسته

از ویکی‌پدیا، دانشنامهٔ آزاد



ساختار ریزهسته و هستهٔ یکپارچه در سیستم‌عامل‌ها (به ترتیب)

در علم رایانه، **ریزهسته** یا **میکروکرنل** یک هستهٔ کامپیوتری است که سازوکارهای لازم برای پیاده‌سازی سیستم‌عامل را فراهم می‌کند، چیزهایی مانند مدیریت فضای آدرس سطح پایین، مدیریت ریسه‌ها (به انگلیسی: thread management)، و ارتباطات میان‌پردازه‌ای. اگر سخت‌افزار چندین سطح دسترسی داشته باشد، آنگاه ریزهسته تنها بخشی از نرم‌افزار است که بالاترین سطح دسترسی را داراست که معمولاً به آن وضعیت هسته می‌گویند. در ساختار ریزهسته‌ای سرویس‌های واقعی سیستم‌عامل مانند راه انداز قطعات، پشته‌های پروتوکل، سیستم فایل‌ها و دستورهای رابط کاربری در فضای کاربر نگاهداری می‌شود.

اگر سخت افزار چندین حلقه یا حالت CPU را فراهم کند ، میکروکرنل ممکن است تنها نرم افزاری باشد که در سطح ممتاز اجرا می شود ، که معمولاً از آن به عنوان سرپرست یا حالت هسته یاد می شود. عملکردهای سیستم عامل سنتی مانند درایورهای دستگاه ، پشته های پروتکل و سیستم های پرونده به طور معمول از میکروکلن حذف می شوند و در عوض در فضای کاربر اجرا می شوند.

از نظر اندازه منبع ، ریزهسته ها غالباً کوچکتر از هسته های یکپارچه هستند. به عنوان مثال ، ریزهسته های MINIX 3 فقط حدود 12000 خط کد دارند.

محتویات

منبع

معرفی

هسته سیستم‌عامل‌های اولیه، به نسبت کوچک بودند، علت این موضوع تا حدی مربوط به کمبود حافظه می‌شد. با رشد کارایی‌های رایانه، تعداد قطعاتی که هسته باید کنترل می‌کرد نیز افزایش یافت. در اوایل ظهور یونیکس، با وجود اینکه هسته شامل درایورهای قطعات و مدیریت‌کننده‌های سیستم فایل می‌شد، اما اغلب هسته‌ها کوچک بودند. هنگامی که فضای آدرس از ۱۶ بیت به ۳۲ بیت افزایش یافت، طراحی هسته دیگر در قید معماری سخت‌افزار باقی نماند و هسته‌ها شروع به رشد کردند.

یونیکس (بی‌اس‌دی) برکلی، دوران هسته‌های بزرگ را آغاز کرد. افزون بر اداره‌کردن یک سیستمِ پایهٔ متشکل از سی‌پی‌یو، دیسک‌ها و چاپگرها، بی‌اس‌دی فایل‌سیستم‌های دیگری را به کار گرفت، از جمله یک سیستم شبکهٔ تی‌سی‌پی/آی‌پی کامل، و شماری از قطعات مجازی که اجازه می‌داد برنامه‌های موجود به صورت پنهان در شبکه اجرا شوند. این بزرگ‌شدن برای سال‌ها ادامه یافت و منجر به هسته‌هایی با میلیون‌ها خط دستور در سورس‌شان شد. در نتیجهٔ این رشد، هسته‌ها بیشتر در معرض باگ‌ها قرار گرفتند و نگهداری آنها به شدت سخت گشت.

ریزهسته برای رفع مشکلات بزرگ‌شدن بیش از حد هسته‌ها طراحی شد. از نظر تئوری، مدیریت دستورها در ریزهسته‌ها ساده‌تردستورهای است زیرا کد آن در سرویس‌های فضای کاربر تقسیم می‌شود. این موضوع همچنین باعث افزایش پایداری و امنیت خواهد شد که نتیجهٔ کاهش مقدار دستورهای اجرایی در وضعیت هسته است. برای مثال اگر یک سرویس شبکه بر اثر سرریز حافظهٔ نهان دچار خرابی شود، تنها حافظهٔ سرویس شبکه دچار خرابی خواهد شد و بقیهٔ سیستم کاملاً کارا خواهد ماند.

سرورها

سرورهای میکروکرنل در واقع مانند سایر برنامه ها برنامه های Daemon هستند ، به جز این که هسته به برخی از آنها امتیاز می دهد تا با بخش هایی از حافظه فیزیکی که در غیر این صورت در بیشتر برنامه ها محدود است ، تعامل برقرار کنند. این اجازه می دهد تا برخی از سرورها ، به ویژه درایورهای دستگاه ، به طور مستقیم با سخت افزار ارتباط برقرار کنند.

یک مجموعه اصلی از سرورها برای یک میکروکرنل با هدف کلی شامل سرورهای سیستم فایل ، سرورهای درایور دستگاه ، سرورهای شبکه ، سرورهای نمایشگر و سرورهای دستگاه رابط کاربر است. این مجموعه از سرورها (تهیه شده از QNX) تقریباً مجموعه خدمات ارائه شده توسط یک هسته یکپارچه یونیکس را ارائه می دهند. سرورهای لازم در هنگام راه اندازی سیستم شروع می شوند و خدماتی مانند دسترسی به پرونده ، شبکه و دستگاه را به برنامه های کاربردی عادی ارائه می دهند. با وجود چنین سرورهایی که در محیط یک برنامه کاربر فعال هستند ، توسعه سرورها به جای فرایند ساخت و بوت مورد نیاز برای توسعه هسته ، شبیه به توسعه نرم افزار معمولی است.

علاوه بر این ، بسیاری از "خرابی ها" را می توان با توقف و راه اندازی مجدد سرور ، اصلاح کرد. با این حال ، بخشی از حالت سیستم با سرور خراب از بین می رود ، از این رو این رویکرد برای مقابله با خرابی به برنامه هایی نیاز دارد. مثال خوب سرور مسئول اتصالات TCP / IP است: اگر این سرور مجدداً راه اندازی شود ، برنامه ها یک اتصال "گمشده" را تجربه می کنند ، یک اتفاق عادی در یک سیستم شبکه ای. برای سایر خدمات ، شکست کمتر انتظار می رود و ممکن است نیاز به تغییر در کد برنامه داشته باشد. برای QNX ، قابلیت راه اندازی مجدد به عنوان ابزار دستیابی بالا QNX ارائه می شود.

درایورهای دستگاه

درایورهای دستگاه معمولاً دسترسی مستقیم به حافظه (DMA) را انجام می دهند ، بنابراین می توانند در مکان های دلخواه حافظه فیزیکی ، از جمله ساختارهای مختلف داده هسته بنویسند. بنابراین باید به چنین رانندگان اعتماد کرد. یک تصور غلط رایج است که این بدان معنی است که آنها باید بخشی از هسته باشند. در حقیقت ، یک راننده ذاتاً کم و بیش با اعتماد به نفس بودن بخشی از هسته نیست.

در حالی که اجرای یک درایور دستگاه در فضای کاربر لزوماً باعث کاهش صدمه ای نمی شود که یک درایور نادرست از آن ایجاد می کند ، در عمل برای ثبات سیستم در حضور درایورهای حشره دار (به جای مخرب) مفید است: نقض دسترسی به حافظه توسط خود کد درایور (برخلاف دستگاه) ممکن است هنوز توسط سخت افزار مدیریت حافظه گیر بیفتد. علاوه بر

این ، بسیاری از دستگاه ها دارای قابلیت DMA نیستند ، درایورهای آنها با اجرای آنها در فضای کاربر قابل اعتماد نیستند. اخیراً ، تعداد فزاینده ای از رایانه ها دارای IOMMU هستند که بسیاری از آنها برای محدود کردن دسترسی دستگاه به حافظه فیزیکی قابل استفاده هستند. این همچنین باعث می شود درایورهای حالت کاربر غیرقابل اعتماد شوند.

درایورهای حالت کاربر در واقع ریزگردها را ردیابی می کنند. سیستم ترمینال میشیگان (MTS) ، در سال 1967 ، از درایورهای فضای کاربر (از جمله پشتیبانی سیستم فایل آن) پشتیبانی کرد ، اولین سیستم عامل که با این قابلیت طراحی شده است. از نظر تاریخی ، رانندگان از مشکل کمتری برخوردار بودند ، زیرا به هر حال تعداد دستگاه ها اندک و قابل اعتماد بودند ، بنابراین داشتن آنها در هسته طرح را ساده تر کرده و از مشکلات احتمالی عملکرد جلوگیری می کند. این امر منجر به سبک سنتی راننده در هسته هسته های یونیکس ، لینوکس و ویندوز NT شد. با تکثیر انواع لوازم جانبی ، میزان کد درایور افزایش یافته و در سیستم عامل های مدرن از نظر اندازه بر هسته غالب است.

مؤلفه های اساسی و حداقل

از آنجا که یک ریزهسته باید اجازه ایجاد خدمات سیستم عامل دلخواه را در بالا داشته باشد ، باید عملکردهای اساسی را ارائه دهد. حداقل این شامل موارد زیر است:

- برخی از مکانیسم های مقابله با فضاهای آدرس ، لازم برای مدیریت حافظه
- برخی از انتزاع اجرای برای مدیریت تخصیص CPU ، به طور معمول موضوعات یا فعال سازی برنامه ریز
- ارتباطات فرایندی ، برای فراخوانی سرورهای فعال در فضای آدرس خود ، مورد نیاز است

این طراحی مینیمم توسط Brinch Hansen's Nucleus و فوق تخصص مشاور IBM در VM آغاز شد. از آن زمان در اصل حداقلی Liedtke رسمیت یافته است:

یک مفهوم در داخل میکروکرنل قابل تحمل است تنها در صورتی که آن را به خارج از هسته منتقل کنید ، یعنی اجازه اجرای رقیب را داشته باشید ، مانع از اجرای عملکرد مورد نیاز سیستم می شود.

هر کار دیگری را می توان در یک برنامه کاربری انجام داد ، اگرچه درایورهای دستگاه که به عنوان برنامه های کاربر اجرا می شوند ممکن است در بعضی از معماری های پردازنده برای دسترسی به سخت افزار I / O از امتیازات ویژه ای استفاده کنند.

جداسازی مکانیسم و سیاست ، مربوط به اصل حداقلی است و برای طراحی میکروکرن نیز به همان اندازه مهم است ، همان چیزی است که ساخت سیستم های دلخواه را در بالای یک هسته حداقل ممکن می کند. هر خط مشی ساخته شده در هسته نمی تواند در سطح کاربر رونویسی شود و بنابراین کلی بودن ریزگردها را محدود می کند. خط مشی اجرا شده در سرورهای سطح کاربر را می توان با جایگزینی سرورها تغییر داد (یا به برنامه اجازه دهید بین سرورهای رقیب ارائه دهنده خدمات مشابه انتخاب کند).

برای کارآیی ، بیشتر ریزگردها حاوی برنامه ریز هستند و تایمر را مدیریت می کنند ، با نقض اصل حداقلی و اصل تفکیک سیاست - مکانیسم.

راه‌اندازی (بوت شدن) یک سیستم مبتنی بر ریزگردها به درایور دستگاه نیاز دارد که جزئی از هسته آن نیست. به طور معمول این بدان معنی است که در تصویر بوت با هسته بسته بندی می شوند و هسته از یک پروتکل bootstrap پشتیبانی می کند که نحوه نصب و راه اندازی درایور را مشخص می کند. این روش بوت استرپ سنتی ریزگردهای L4 است. برخی از ریزگردها با قرار دادن برخی از درایورهای کلیدی درون هسته (با نقض اصل حداقلی) این کار را ساده می کنند ، LynxOS و اصلی Minix نمونه هایی هستند. برخی حتی برای ساده کردن بوت کردن ، یک سیستم پرونده در هسته دارند. سیستم مبتنی بر میکروکلنر ممکن است از طریق لودر سازگار با چند بوت راه اندازی شود. چنین سیستمهایی معمولاً برای ایجاد بوت استرپ اولیه یا سوار کردن تصویر OS ، سرورهای متصل به استاتیک را بارگیری می کنند تا به ادامه راه اندازی ادامه دهند.

یک مؤلفه اصلی یک میکروکرنل یک سیستم IPC خوب و طراحی حافظه مجازی مجازی است که اجازه می دهد تا با استفاده از پردازش خطای صفحه و مبادله در سرورهای usermode به روشی ایمن انجام شود. از آنجا که کلیه خدمات توسط برنامه های usermode انجام می شود ، ارتباطات کارآمد بین برنامه ها بسیار مهم تر از هسته های یکپارچه است.

طراحی سیستم IPC باعث ایجاد خرد یا خرد شدن خرد می شود. برای مؤثر بودن ، سیستم IPC نه تنها باید از سربار کم برخوردار باشد ، بلکه با برنامه ریزی CPU نیز ارتباط خوبی دارد.

کارایی

در اکثر پردازنده های اصلی ، دریافت یک سرویس ذاتاً در یک سیستم مبتنی بر میکروکرنل گران تر از یک سیستم یکپارچه است. در سیستم یکپارچه ، خدمات با یک تماس سیستم واحد به دست می آیند که به دو سوئیچ حالت (تغییر حلقه پردازنده یا حالت CPU) نیاز دارد. در سیستم مبتنی بر میکروکرنل ، این سرویس با ارسال پیام IPC به سرور و به دست آوردن نتیجه در پیام IPC دیگری از سرور حاصل می شود. اگر درایورها به عنوان فرآیند یا یک فراخوانی عملکردی در صورت اجرا به عنوان رویه انجام شوند ، این نیاز به سوئیچ زمینه دارد. علاوه بر این ، انتقال داده های واقعی به سرور و پشت ممکن است باعث کپی اضافی شود ، در حالی که در یک سیستم یکپارچه هسته می تواند مستقیماً به داده های موجود در بافر مشتری دسترسی داشته باشد.

بنابراین عملکرد در سیستم های میکروکرنل یک مسئله بالقوه است. در واقع ، تجربه ریزهسته های نسل اول مانند Mach (ماک) و ChorusOS نشان داد که سیستم های مبتنی بر آنها عملکرد بسیار ضعیفی دارند. با این حال ، ژوشن Liedtke نشان داد که مشکلات عملکرد ماک نتیجه طراحی و اجرای ضعیف است ، به طور خاص رد پای بیش از حد ماک. Liedtke با میکروکرنل L4 خود نشان داد که با طراحی دقیق و اجرای دقیق ، و به ویژه با پیروی از اصل حداقلی ، هزینه IPC را می توان بیش از یک سفارش بزرگتر از ماک کاهش داد. عملکرد IPC L4 هنوز در طیف وسیعی از معماری ها بی نظیر است.

در حالی که این نتایج نشان می دهد که عملکرد ضعیف سیستم های مبتنی بر ریزهسته های نسل اول نماینده ای برای هسته های نسل دوم مانند L4 نیست ، این هیچ اثبات کننده ای نیست مبنی بر اینکه می توان سیستم های مبتنی بر میکروکرنل را با عملکرد خوب ساخته کرد. نشان داده شده است که یک سرور یکپارچه لینوکس که به L4 منتقل شده است ، تنها چند درصد بالای سرور لینوکس را نشان می دهد. با این حال ، چنین سیستم تک سرور ، در صورت وجود ، تعداد کمی از مزایای استفاده از میکروکرنل ها را با ساختار عملکرد سیستم عامل در سرورهای جداگانه ارائه می دهد.

تعدادی سیستم چند سرور تجاری ، به ویژه سیستم های واقعی QNX و Integrity وجود دارند. هیچ مقایسه جامع از عملکرد نسبت به سیستم های یکپارچه برای آن سیستم های چند سرور منتشر نشده است. علاوه بر این ، به نظر نمی رسد عملکرد مهمترین نگرانی برای آن سیستم های تجاری باشد ، که در عوض بر زمان پاسخگویی سریع کنترل سریع وقفه (QNX) و سادگی به دلیل استحکام تأکید می کنند. تلاش برای ساخت یک سیستم عامل چند کاره با کارایی بالا ، پروژه IBM Sawmill Linux بود. با این حال ، این پروژه هرگز تکمیل نشد.

در این میان نشان داده شده است که درایورهای دستگاه در سطح کاربر حتی می توانند برای دستگاههای با توان بالا و با فشار زیاد مانند Gigabit Ethernet به عملکرد درایورهای درون هسته نزدیک شوند. به نظر می رسد که سیستم های چند سرور با کارایی بالا امکان پذیر است.

امنیت

فواید امنیتی ریزهسته ها به طور مکرر مورد بحث قرار گرفته است. در چارچوب امنیت ، اصل حداقلی ریزهسته ها نتیجه ای مستقیم از اصل حداقل امتیاز است که براساس آن ، کلیه کد ها فقط باید از امتیازات لازم برای تأمین کارکردهای مورد نیاز برخوردار باشند. حداقل بودن نیاز دارد که پایگاه محاسباتی قابل اعتماد سیستم (TCB) حداقل نگه داشته شود. از آنجا که کرنل (کدی که در حالت ممتاز سخت افزار اجرا می شود) دسترسی ناخواسته به هر داده ای دارد و می تواند یکپارچگی یا محرمانه بودن آن را نقض کند ، هسته همیشه بخشی از TCB است. به حداقل رساندن آن در یک طرح امنیتی محور طبیعی است.

در نتیجه ، از طرح های میکروکرن برای سیستم هایی که برای برنامه های با امنیت بالا از جمله EROS ، KeyKOS و سیستم های نظامی طراحی شده اند ، استفاده شده است. در حقیقت معیارهای رایج (CC) در بالاترین سطح تضمین (سطح تضمین ارزیابی (7 (EAL) الزام صریح دارند که هدف ارزیابی "ساده" باشد ، اذعان به عدم امکان عملی ایجاد اعتماد واقعی برای یک سیستم پیچیده است. متأسفانه ، بار دیگر ، اصطلاح "ساده" گمراه کننده و بد تعریف شده است. حداقل وزارت دفاع اعتبار معیارهای ارزیابی سیستم رایانه ای ، کلمات کمی دقیق تر را در کلاس های A1 / B3 معرفی کرد:

"TCB باید مکانیسمهای حفاظتی کاملاً ساده و مفهومی را با معناشناسی دقیق تعریف شده [پیاده سازی] کند. مهندسی مهم سیستم به سمت به حداقل رساندن پیچیدگی TCB هدایت می شود ، و همچنین از TCB آن ماژولهایی را که محافظت از اهمیت زیادی ندارند ، هدایت می شود."

- معیارهای ارزیابی سیستم کامپیوتری وزارت دفاع

نسل سوم

کار اخیر در زمینه ریزگردها بر روی مشخصات رسمی API هسته و اثبات رسمی از ویژگی های امنیتی API و صحت اجرای آن متمرکز شده است. مثال اول این یک اثبات ریاضی مکانیسم های محصور در EROS است که بر اساس یک مدل ساده شده از EROS API ساخته شده است. اخیراً (در سال 2007) مجموعه کاملی از اثبات ماشین بررسی شده از خواص مدل محافظت از seL4 ، نسخه L4 انجام شد.

این امر به آنچه به عنوان ریزگردهای نسل سوم گفته می شود ، انجام شده توسط API امنیتی گرا با دسترسی به منابع کنترل شده توسط قابلیت ها ، مجازی سازی به عنوان نگرانی طبقه اول ، رویکردهای جدید در مدیریت منابع هسته و هدف طراحی مناسب بودن برای تجزیه و تحلیل رسمی ، علاوه بر هدف معمول از عملکرد بالا. نمونه هایی از ، seL4 ، Coyotos Redox ، Nova و Fiasco.OC هستند.

در مورد seL4 ، تأیید رسمی کاملی از اجرای به دست آمده است ، یعنی یک اثبات ریاضی که اجرای هسته با مشخصات رسمی آن سازگار است. این تضمین می کند که خواص اثبات شده در مورد API در واقع هسته را حفظ می کنند ، درجه ای از اطمینان که حتی از CC EAL7 فراتر رود. پس از آن ، اثبات خصوصیات امنیتی امنیتی API به اثبات رسیده است ، و اثبات این امر نشان می دهد که کد باینری قابل اجرا ترجمه صحیح از اجرای C است و کامپایلر را از TCB خارج می کند. با هم جمع شده ، این اثبات پایان دهنده اثبات خواص امنیتی هسته را ایجاد می کنند.

نانوکرنل

اصطلاح نانوکرنل یا پیکوکرنل از لحاظ تاریخی به آن اشاره شده است:

- هسته ای که مقدار کل هسته ، یعنی اجرای کد در حالت ممتاز سخت افزار ، بسیار اندک است. اصطلاح پیکوکرنل گاهی برای تأکید بیشتر بر اندازه کوچك به کار می رفت. اصطلاح نانوکرنل توسط جاناتان س. شاپیرو در مقاله معماری KeyKOS NanoKernel ابداع شده است. این پاسخ ساردونیکی به ماک بود که ادعا می کرد که یک میکروکرنل است در حالی که شاپیرو آن را یکپارچه ، اساساً بدون ساختار و کندتر از سیستم هایی که می خواست جایگزینی کند تلقی می کرد. استفاده مجدد و پاسخ به این اصطلاح ، از جمله سکه پیکوکرنل ، نشان می دهد که این نقطه تا حد زیادی از دست رفته است. متعاقباً ، هر دو نانوکرنل و پیکوکرنل به معنای یکدیگر با اصطلاح میکروکرنل بیان شده اند.
- یک لایه مجازی سازی در زیر یک سیستم عامل ، که به صحیح تر به عنوان مشاور Hypervisor گفته می شود.
- یک لایه انتزاع سخت افزاری که پایین ترین سطح یک هسته را تشکیل می دهد ، گاهی اوقات برای ارائه قابلیت های زمان واقعی برای سیستم عامل های عادی مانند Adeos استفاده می شود.

همچنین حداقل یک مورد وجود دارد که اصطلاح نانوکرنل برای اشاره به هسته کوچک نیست ، بلکه موردی برای پشتیبانی از وضوح ساعت نانو ثانیه است.

جستارهای وابسته

- هسته
- هسته پیوندی
- هسته یکپارچه
- اگزو کرنل

منبع

- مشارکت‌کنندگان ویکی‌پدیا. «Microkernel (https://en.wikipedia.org/wiki/Microkernel)». در *دانشنامهٔ ویکی‌پدیای انگلیسی*، بازبینی‌شده در ۱۲ خرداد ۱۳۹۹.