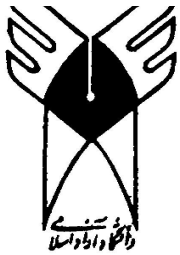

دانشگاه آزاد اسلامی

واحد تهران جنوب

مروری بر سیستم عامل های

WIN NT و XINU و MINIX و UNIX

استاد راهنما:

.....

مقدمه

ویندوز NT نسبت به سیستمهای عامل OS/2 و UNIX و ویندوز ۱۶ بیتی دارای برتری‌هایی است که این خصوصیات و امکاناتی که ویندوز NT دارد، با یک سیستم عامل یا بیشتر نیز قابل دسترسی است. ولی هیچ کدام از سیتم عاملها قابلیتهای ویندوز NT را ندارد. در این قسمت مهمترین خصوصیات NT را معرفی می کنیم و یکسری خصوصیات کلیدی NT را شرح می‌دهیم.

۱- آدرس‌دهی ۳۲ بیتی

در انیجا لازم است توضیحی در مورد اینکه آدرس چیست و نحوهٔ آدرس‌دهی که میکروسافت DOS چگونه است بدهیم. آدرس کلاً محل یک بایت از اطلاعات در حافظه کامپیوتر یا Mass storage می گویند. آدرس بر دو نوع است. آدرس می‌تواند فیزیکی و یا می‌تواند مجازی باشد.

۱-۱- آدرس‌دهی فیزیکی: به بایت دادهٔ معین در محل فیزیکی معینی از حافظه یا دیسک اشاره می کنند.

۱-۲- آدرس‌دهی مجازی: به آدرس منطقی (نرم‌افزاری) اشاره می کند که سیستم عامل به آدرس فیزیکی معینی اشاره می کند.

مطلب قابل توجه در انیجا این است که ویندوزNT از بین دو آدرس‌دهی، از آدرس‌دهی مجازی استفاده می کند که برای هر درخواست اصولاً چهار گیگابایت اختصاص داده می‌شود که البته ۲ گیگابایت آن برای سیستم عامل منظور می‌شود.

به دنبال پیشرفت مایکروسافت DOS به دلیل محدودیتهای حافظه، میکروپروسسورهایی از قبیل 8086 و 8088 شانزده بیتی که یک فضای آدرس‌دهی بیست بیتی را عرضه می‌داشتند طراحی کردند. یعنی در واقع این میکروپروسسورهای می‌توانستند یک مگابایت از حافظه را به طور فیزیکی آدرس‌دهی کنند. با توجه به این که بیست بیت به خوبی در کلمه (word) شانزده بیتی جای نمی‌گرفت مهندسین INTEL برای دستیابی به هر آدرس، طرحی به نام تقسیم حافظه ارائه دادند.

در این نوع آدرس‌دهی آمدند یک مگابایت را به ۱۶ قسمت تقسیم کردند که هر قسمت ۶۴ کیلوبایتی با ghunk بود. این کار بدین دلیل بود که برنامه‌نویسان اصولاً در پردازنده‌های ۸ بیتی قدیمی مانند INTEL 8085 و ZILOG Z80 با آدرس‌دهی ۶۴ kdyte آشنایی داشتند. برای اینکه برنامه‌نویسان بتوانند به هر آدرسی در داخل فضای آدرس یک مگابایتی دست پیدا کنند آدرس حافظه فیزیکی محاسبه شد. (قسمت در ۱۶ ضرب کرده و سپس یک offset به آن اضافه می‌شود. نتیجه به بایت مورد نظر اشاره می‌کند.) بقیهٔ پردازنده‌های ۶ بیت مانند Motorola 68000 از آدرس‌دهی خطی استفاده می‌کردند که هربیت از حافظه مستقیماً و بدون استفاده از offset & segment آدرس‌دهی می‌کرد. و علاوه بر تسهیلاتی مانند MS-DOS 5.0 و windows 3.1، طراحان زیادی طرحهای خود را برای آدرس‌دهی و بیش از ۶۴۰ کیلوبایت حافظه پیشنهاد کردند. می‌توان از معروفترین نمونه‌ها QEMM و QUALAS' 386 MA و QUARTERDECK'S را نام برد.

کل این و طرحها حافظهٔ مورد استفاده را بیش از حد ۶۴۰ کیلوبایتی بسط دادند، ولی مجبور کردن آنها به کار با یک موقعیت خاص PC و دنبالهٔ درخواستها و تسهیلات نرم‌افزاری

معمولاً یک هدر کردن زمان، پردازش خنثی کننده است. همیشه این‌طور به نظر می‌رسد که حداقل یک درخواست مهم با یکی از تسهیلات با مدیریت حافظهٔ شما سازگار نیست.

اولین پردازنده که فضای آدرس‌دهی خطی را به کار برد و نیز با DOS سازگاری داشت INTEL 386 بود که می‌بایست تقسیم‌بندی حافظه در کنار آن احتیاج به مدیریت شخص ثالث را حذف کرد که در واقع فضای آدرس‌دهی ۳۲ بیتی INTEL 386 با برنامه‌ها نوشته شده برای پردازنده‌های INTEL قبلی سازگاری نداشت.

این برنامه‌های ناسازگار، DOS و تمام برنامه‌های اجرا شده تحت DOS بودند. برایایجاد سازگاری با DOS و درخواستهای آن INTEL یک طریقهٔ دیگر آدرس‌دهی را طرح کرد بنام Real Mode .

این نوع آدرس‌دهی با سایر نرم‌افزارهای قبل سازگاری داشت ولی متذسفانه در هنگام کار با این نوع آدرس‌دهی یعنی Real moed ، 386 و (486) بیش از یک 8086 خیلی سریع عمل نمی‌کند. DOS که در مقابل محدودیتهای 8086 و 8088 نوشته شده بود، هنوز بسیار شبیه نسخه سریع حد خود در سال ۱۹۸۱ با تمام محدودیتهای آدرس‌دهی عمل می‌کنند.

دو طریق آدرس‌دهی حافظه:

Real Mode : یک طریقهٔ آدرس‌دهی حافظه است که از آدرس‌دهی نوع تقسیم‌بندی حافظه یعنی همان segment & offset استفاده می‌کنند. مانند پردازندهٔ INTEL 8086.

Protected Mode : یک طریقهٔ دیگر آدرس‌دهی حافظه که برای دستیابی به بایت حافظه به جای segment & offset از آدرس‌دهی خطی استفاده می‌کند. این نوع آدرس‌دهی

مشخصات حفاظت سخت‌افزاری که windows و windows NT آن را به کـار گرفتـه‌انـد، فعال می کند.

windows NT بـرای تهیـه آدرس‌دهـی ۳۲ بیتـی واقعـی Trune 32-bit addressing از آدرس‌دهـی خطـی 386 و 186 و پنتـوم (INTELS586) و همچنـین پردازنـده pisk همچنون Mips و DigtalAlpha استفاده می کند. قابل ذکر است کـه دیگـر NT از سازگار بودن با DOS و windos ۱۶بیتی صرفنظر مـی کنـد و دارای طرحـی اسـت بنـام (viriul dos machine) که در این صورت این امکان را به ویندوز NT مـی دهـد کـه بتواند در خواستهای Dos و windows۱۶ بیتی را اجرا کند (بعدا در مورد VDMصـحبت خواهیم کرد) .

مزیت های آدرس دهی ۳۲ بیتی :

۱-توسعه نرم افزار با حذف قسمت بندی حافظ آسانتر و سریع تر می شود.

۲- برنامه نویسان دیگر لآزم نیست حافظه مورد در خواستهایشان آشنا باشد .

۳-کارسیستم باحذف سربارپردازشی که موردلزوم مدیریت حافظه است بهبود می یابـد.یعنی در هیچ گونه نیازی به حافظه ثالثی ندارد.رهایی از مدیریت حافظه سـازگاریهای NT واقع ویندوز مختلف سخت افزار ونرم افزار را نیزحذفمی کندیعنی وضعیت اسـتقرارنرم افـزار مـی تواند ساده و ۱۶ بیتی باشد Windowsیا Dosابتدایی تر از

۴- میزان برنامه قابل دسترسی و اندازه داده در آدرس دهی ۳۲ بیتی زیاد می شود.

ویندوز NTاز ترکیب برنامه و سیستم با اندازه چهار گیگا با یت که صدها برابر بزرگتـر از حدودقابل اجرا روی بر نامه های DOS و ویندوز ۱۶ بیتـی تشـکیل شـده اسـت . فایلهـای

بزرگ که توسط ویندوز NT قابل پردازش می باشد غیر ممکن است که توسط DOS ویاwindos۱۶ بیتی پردازش شوددر خواست کننده های (در خواستهای) پیچیده که فایلهای بزرگ را پردازش میکنند فقط با ویندوز NTعمل می کندو آن هم فقط به دلیل آدرس دهی ۳۲ بیتی است . در خواستهای پیچیده همچون رزرواسیون، مبادله دارایی وسیستمهای پردازش طلبهای بیمه است

از دیگر خصوصیات کلیدی ویندوز NT،حافظه مجازی و یا VMاست که در زیر توضیح می دهیم .

۲- VIRTUAL MEMORY (حافظه مجازی)

هر در خواست در ویندوز NT میتواند به ۴ گیگا بایت حافظه دست پیدا کند (به خاطر فضای ۳۲ بیتی)

که البته از مقدار تصور شده برای هر در خواست بیشتر است . دو نوع حافظه اصلی کامپیوتر به شرح زیر است :

RAM : RAMیا حافظه تصادفی (حافظه دسترسی تصادفی) از نوع دیگر سریعتر است .

مزیتهای آن به شرح زیر است :

۱- PCبرای اینکه بتوانند یک بایت داده را در ۷۰ بیلیونیم ثانیه دریافت و بعد ذخیره کنندازRAM استفاده می کنند

۲-برنامه ها به طور مستقیم می توانند به آدرس حافظه بروند و بایت مورد نظر خود را دریافت کنند درواقع بجای اینکه به بلاک مورد نظر در روی دیسک مراجعه کند و با یتها ی

بلاک را بایت به بایت بخواند وبه بایت مورد نظر برسد می تواند بـه طـور مسـتقیم بـه آدرس حافظه بروند . ترجیحا استفاده از RAM را پیشنهاد می کنند.

عیوب استفاده ازRAM :

۱ -در هنگام قطع برق تمام اطلاعات ذخیره شده در RAMاز بین می رود.این نوع حافظه را STORAGE VOLATLنامیده میشود . در این جـا قابـل ذکـر اسـت کـه در بسـیاری از کامپیوتر هایی که قابل حمل هستند

هنگامی که کامپیوتر خاموش است مقداری برق به حافظه RAMکامپیوترمی رسدکه باعـث می شود داده های ذخیره شده در آن از بین نرود و در هنگام تمام شدن باطری آن اطلاعات در آن با تمام شدن باطری از بین می رود.

۲-عیب دیگـر RAMهـا در قیمـت آن مشـاهده مـی شـودوآن نیزگـران بـودن قیمـت RAMها است در واقع هر مگا بایت از RAM ها بـرای pc هـا حـدود ۳۵ دلار فروختـه مـی شود . از روی این قیمت می تواند تعداد RAMهایی که در یک pc می توان نصب کـردرا حدس زد .اصولا به pc های kigh-end ۸ مگا بایت RAM وصل می شود و ایـن مقـدار را خیلی ها می توانند تا ۲۰ مگابایت افزایش دهند و بعضی نیـز ایـن مقـدار را بـه ۶۴ مگابایـت میرسانند .برای این امر به RAM با چگـالی بـالا تـری نیـاز اسـت کـه قیمـت آن بـه ازای هـر مگابایت افزایش می یابد ولی فضای فیزیکی کمتری را اشغال میکند نوع دیگر حافظه اصلی در کامپیوتر hard diskاست که در زیر شرح داده شـده اسـت :(نـوع دیگـر حافظـه Mass storage ها هستند که روی hard diskسواری می شوند.

HARD DISK: Mass STORAGE

همان طور که در بالا گرفته شده نوع دیگر حافظه اصلی Mass storageها هستند که بر روی hard diskسوار می شوند hard diskها به مراتب کند تر از RAMها هستند .

از نظر قابلیت ذخیره سازی hard diskها از ۴۰ مگا بایت تا حدود ۲ گیگا بایت قابلیت ذخیره سازی دارند .

از نظر قیمت هم یک pc hard ۲۰۰ مگابایتی تقریبا ۴۰۰ دلار ویا به عبارتی دو دلار به ازای هر مگا بایت می باشد که در مقایسه با قیمت RAMکه در حدود $\frac{1}{15}$ قیمت یک مگا بایت RAMاست . در هنگام قطع برق محتویات hardها از بین نمی رود به این گونه حافظه Nonvolatile storageمی شود .

استفاده از هر دو نوع حافظه برای pc ضروری به نظر می رسد.از RAMبرای برنامه های اجرایی و ذخیره.

دادههای حساس هنگامی که اجرا ضروریست استفاده می شود واز hard disk برای ذخیره طولانی اطلاعات در زمانی که قیمت به ازای هر بایت مهم می باشد، استفاده میشود .

در این جا یک سئوالی که پیش می آید این است که اگر به بیش از RAM قابل دسترسی نیاز پیدا کردیم چه می کنیم؟مثلاً فرض کنید که یک برنامه spreed sheetبه دو مگا بایت و RAM ویک فایل spreed sheetکه به دو مگا بایت احتیاج دارد داشته باشیم . این سئوال پیش می آید که آیا فقط به همین ۴ مگابایت نیاز داریم ؟ خود سیستم عامل به مقداری زیاد RAM احتاج دارد . پس ما میزان لازم RAM برای سیستم عامل و پردازنش speed sheet به طور همزمان در دست نداریم .

در DOS بـرای فراخـوانی برنامـه و داده اگـر RAM بـه مقـدارکافی نداشـتیم مجبوربودیم یک RAMاضافه دیگر خریداری می کردیم وبه PCنصب مـی کـردیم. ولـی در ویندوز NTما راه چاره ای داریم وآن این اسـت کـه امکـان تبـدیل قسـمتی از hard بـه فضای RAMوجود دارد.به طوری که در خواسـتهایی بزرگتـر از آن هسـتند کـه در حافظـه RAM جـای بگیرنـد ، مـی تـوانیم اجـرا کنـیم . ایـن خصوصـیت ویـنـدوز NT را viriual Memory نامگذاری کردند . بعدا به طرز کار VM تحت ویندوز NTوتوضـیح وشـرح آن خواهیم پرداخت .

در هنگام نصب ویندوز NT(برای اولین)کاربران و یـا مدیرسیسـتم موظـف اسـت کـه برنامه راه انداز NTرا چک کند تا به فضای قابل دسترسی پـذیر RAM وHARDدر سیسـتم عاملی پی ببرد.برپایه فضای دسترسی پذیر دیسک RAM یک swap file ایجـاد میکننـد که اندازه آن حداکثر می تواند به اندازه RAM موجود در سیسـتم باشـد . در هنگـام نصـب ویندوز NT کـاربر میتوانـد انـدازه swap file را تغییـر بدهـد . رابطـه swap file بـا حافظـه مجازی از نظر اندازه مستقیم است . یعنی هر چـه انـدازه swap file بـزرگ باشـد ، حافظـه مجازی نیز بزرگ میباشد. ولی قابل ذکر است که بزرگی آن تا اندازه فضای ذخیـره سـازی فایل ثابت، ادامه خواهد داشت .

اندازه نهایی swap file را جابجایی وظرفیت کل دیسک مشخص می کند.در زیر توضـیحی در مورد swap fileو این که swap file چیست می دهیم .

SWAP FILE

swap file قسمتی از حافظه سخت است که توسط مدیریت حافظه مجازی بکار می رود که کارش در واقع نگهداری موقت بخشی از محتویات RAM است تا اینکه به سیستم این امکان داده شود که برنامه هلآیی که از نظر اندازه از RAM قابل دسترسی بزرگترند ،را بتواند اجرا کند.

دنباله بحث :بعد از نصب ویندوز NT و اجرای آن swap file به عنوان یک انباره موقت برای محتویات RAM بکار برده می شود. در زیر دو وظیفه مهم مدیر حافظه مجازی را بررسی می کنیم:

۱-مدیریت داده ذخیره شده بر روی دیسک وانتقال آدرس داده های روی پایه دیسک به فضای آدرس دهی ۳۲ بیتی ویندوز NT .در خواست میتواند عملیاتی را بر روی داده انجام دهد بدون اینکه توجه به این داشته باشد که داده از نظر فیزیکی درکجا قرارگرفته است.یعنی میتواند عملآیاتی را برروی داده درفضای حافظه مجازی انجام دهد

۲-در ویندوز NT هنگامی که بخواهند از RAMبیشتر از آنچه که در دسترس است استفاده کنند این مدیریت حافظه مجازی است که قسمتی از RAMرابه swap file منتقل می کندتا اینکه فضایی برای داده های لازم باز کند و در زمان لازم دوباره بهRAM بازگردانده می شود .برای جلوگیری از هدر رفتن زمان ،به جای اینکه داده ها بایت به بایت از RAM به ذیسک ویا از دیسک به RAMمنتقل شوند ،این انتقال به این صورت است که داده های به صورت۴ kilopagesمنتقل می شوند نه بایت به بایت.برنامه درخواستی لزومی

نداردکه چیزی در مورد پردازش swaping بداند swapingتکرار شونده روی پاسخ خرابی سیستم وهمچنین روی فعالیتهای سنگین دیسک تاثیر گذار است .

برنامه ای که یک فایل راروی فضای آدرس دهی مجازی ویندوزط NT مورد پردازش قرار می دهد(روی هارد)از نظر سرعت خیلی کند تر از برنامه ای است که از دادها در داخل RAMاستفاده میکند .یعنی برنامه اولی از برنامه دومی کندتر اجرا می شود .مدیریت حافظه مجازی برای این طرحی که در فایلهای داده ای خیلی بزرگتر و سیستمهای چند برنامه ای از به وجود آمدنswaping ها خیلی ضروری جلو گیری کند ولی در واقع میزان کار به مقدار RAMقابل دسترس بستگی دارد یعنی هر چه مقدار RAMقابل دسترس بیشترباشد کار کلی بیشتراست ویندوز ۱۶ بیتی هم نیز از حافظه مجازی استفاده می کند ولی در طرز کار آن در ویندوز NTپیچیده تر است .اندازه فضای swapحافظه مجازی در ویندوز ۱۶ بیتی وابسته به مقدار RAMقابل دسترس است وتا تقریبا اندازه ۳۰ مگا بایت مجاز است .این نکته قابل ذکر است که اگر کارتصمیم به حذف swap fileها با خاموش کردن حافظه مجازی بگیرد،و به این ترتیب در خواستهای بزرگتری را انجام دهد ،به swap fileاین نیاز نیست .

این حدود اندازه swap fileدر ویندوز ۱۶ بیتی برای top disk deskها (desk top pc ها مناسب میباشد .اگر مقدار ramدسترس پذیر شما در حدود ۳۲ مگا بایت یا بیشتر باشد ،کمترین مقدار swapفضای یکdiskرا اشغال میکنددرحالی که ۳۰مگابایت از فضایseap برای بسیاری تزدز خواستهایRAMبسیار کم است .ولی ویندوز NT این خصوصیت را دارد که شما هر مقدار swap(زیاد وکم)که نیازدارد ،اختصاص دهید . ولی این موضوع

هم هست که حافظه مجازی نمی تواند جوابگوی تمام مشکلات با شد .این امکان را می دهد (حافظه مجازی)که با مقداری RAMمحدود برنامه های بزرگ را پردازش کنیم ولی بدون توجه به سرعت پردازش آن یعنی ممکن است سرعت پردازش خیلی کم باشد .در هنگام کار با ویندوز NTهنگامی که حافظه مجازی مشغول مبادله dataبه hardاز RAM به hardویا بالعکس است.توجه داشته باشید که به محض پایین آمدن کارایی سیستم ،به RAM اضافه کنیم.البته ممکن است که با اضافه کردن hardسریعتر به اصلاحاتی دست یابیم ولی مزایای بالا بردن مقدار دیسک سریعتر قابل مقایسه با اضافه کردن RAM نمی باشد .از دیگر خصوصیات (خصوصیت سوم) در زیر توضیح داده می شود .

۳-PREEMPTIVE MULTITASKING

این خصوصیت به معنا ی چند کاره بودن یعنی قابلیت چند کاره بودن کامپیوتر برای انجام یک سری عملیات در واحد زمان است.این عملیات،اجرای برنامه ویا پردازش یک برنامه و یا به تمام رساندن یک کار ، می تواند باشد .درNT برنامه وپردازش ها از اجرایی به نام threadتشکیل شده اند که بعد در مورد آنها بحث می شود .

THREAD

معمولا قسمتی از پردازش به عهده سیستم عامل است که سیستم عامل آن را اجرا می کند که به آن قسمت thrad گفته می شود . یک thred نمی تواند متعلق به چند پردازش باشد ولی یک پردازش می تواند چند threadداشته باشد و می توانند به وسیله طرحی چند threadرا به طور همزمان اجرا کند.

حال می خواهیم ببینیم مفهومmultiasking یعنی چه و برای چه کاری است؟برای اینکه به این مفهوم بیشتر آشنا بشوم قبل از هر توضیحی اشاره به مثالی می کنیم .می دانیم که نوشتن یک documentکارراحتی است.هنگامی که داریم با DOSکارمی کنیم وبخواهیم ازنوشته پرینت تهیه کنیم قبل ازانجام هر کاری بایست فرمها،نوع قلم،کدهای خاص را به کدها یی قابل فهم برای .چاپگرتبدیل کنیم سپس سیستم را اندازه فایل را که می داند چاپگر توانایی تحمل آن را دارد به چاپگر می فرستد. چاپگر و سیستم عامل انقدر به تبادل اطلاعات ادامه می دهندکه تمام فایل به چاپگرفرستاده شود .در حین تبادل اطلاعات بین چاپگر و سیستم عامل ،دیگر ما به عنوان یک کاربر هیچ کار دیگری نمی توانیم با کامپیوترانجام دهیم .وجود یک سیستم multitaskingاست که در اینجا لازم دیده می شود در واقع multitaskingاین اجازه را به کاربرمی دهدکه همزمان با ارسال اطلاعات به چاپگر ،بتواند عملیات دیگری بر روی کامپیوتر انجام دهد

همانطور که میدانیم پردازنده در هر زمان فقط یک کار را انجام می دهد ولی این کارها سریع تغییروضعیت می دهد که به نظرمی آید در هرزمان بیش ازیک کارانجام می شود.مثل دیگری که برای فهم بهتر multitasking می شود زد ،در مورد فیلم است .اگر ما

یک حلقه فیلم را جلوی نور بگیریم می بینم از تعدادی تصویر و عکس ثابت تشکیل شده است ولی آن را با یک پروژکتور به صورت ۲۴ تصویر در ثانیه نمایش دهند ما همان تصویرهای ثابت رابه صورت یک فیلم متحرک می بینیم و برای ما غیر باور است که فکر کنیم آن همان عکسهای ثابت است که به صورت فیلم درآمده است .یک multitasking انعطاف پذیر خوب همانند این فیلم دنباله دار ومتحرک به نظر می آید .

در هنگام اجرای هر thread ، کنترل تمام کامپیوتر بدست آنها می افتد ولی در بعضی اوقات سیستم عامل می آید این thread و اطلاعات ضروری برای سیستم را به محلی که مشابه swap شرح شده در قبل می باشد ، انتقال دهد و thread دیگر بداخل فرستاده می شود و اجراو سپس به خارج فرستاده می شود این عمل این قدر سریع انجام می شودکه کاربر احساس می کند thread ها با هم در یک زمان اجرا می گردد .

دو نوع multitasking وجود دارد که اول آنها را نام برده سپس شرحمی دهیم .

۱-preemptive

۲- nonpreemptive

: nonpreemptive

دراین نوع multitasking ، قبل از جایگزین یک thread در حال اجرا با thread دیگر thread مشخص می کند که چه مدت زمان کار دارد .

: Preemptive multitasking

هرthread برایش مقدار ثابتی زمان مشخص می شود تا در این زمان ثابت کار خود را انجام دهند که به این زمان timeslice گفته می شود.وقتی که یکthread زمان time slicاش

تمام شد موظف است که به خارج انتقال یابد. مسئله دیگری هم که است مسئله حق تقدم است یعنی یک thread با اولویت بالا میتواند جلوی اجرای thread با اولویت پایین تر را بگیرد.در عوض ویندوز NTکه ازpreemptive استفاده می کند ویندوز ۱۶ بیتی از multitasking از نوع nonpreemptiveاستفاده می کنیم .

مقایسه nonpreemptive و Preemptive :

در nonpreemptive هر در خواست برای اجرای در خواستهای دیگر ناچاربه رها کردن کنترل پردازنده است که البته در مدل Preemptive دیگر به این صورت نیست بلکه سیستم عامل خود به طور اتوماتیک موقتا کنار کذاشته ودر خواستهای مورد لزوم برای کار،سرویس داده می شود .همان طور که در بالا گفته شدویندوز ۱۶بیتی ازnonpreemptive استفاده می کند به نام cooperative.دراین سیستمcooperative هردرخواستی برای اجرای درخواست دیگر کنترل پردازنده را رها می کند.

در یک سیستم nonpreemptive ،در حالی که یک thread خودش دارد اجرا می شود هیچ thread دیگری نمی تواند کامپیوتر را در اختیار گیرد. ولی درPreemptive اگر مدت زمان لازم برای هر thread Xباشد برای دو thread دو برابر این مقدار ،برای سه thread سه برابر این مقدار و ...می باشد . به علاوه threadهای با اولویت بالاتر می توانند بر thread ها با اولویت پایین تر تقدم یابند . برای فهمیدن بهتر مفهوم Preemptive و nonpreemptive به مثالی جوع می کنیم :

در این مدلها multitaskingمی توانیم پردازنده را مانند یک راهداری در جاده در نظر بگیریم .در یک سیستم Preemptive ، هر ماشینی زمانی که صرف ورود به راهدار وپول

دادن و خروج می کند،برای هر ماشین یکسان است . مثال آ یک ماشین ،که حدودا دهمین ماشین است میتواند تخمین بزند که چه زمان دیگر نوبت او می رسد .(مثل سیستم عامل ویندوز NT) ولی در سیستم nonpreemptive دیگر این طور نیست یعنی فقط هر ماشین با ورود به راهداری می تواند تخمین بزند که چه مقدار کار دارد دیگر ماشینی که در صف ماشین های است نمی تواند تخمین بزند که کی و چه زمانی نوبت او فرا می رسد. حال برای اینکه بفهمیم که سیستم nonpreemptive سست است مثالی میزنیم :

در ویندوز ۱۶بیتی مدیریت چاپ برای مسیر دهی ducumentها به چاپگر استفاده می شود . به این ترتیب که مدیریت چاپ همواره با چاپگر دررابطه است که در صورت بیکاربودنducumentهارا منتقل کنددر غیر این صورت روی فضای hardذخیره کند.در طی این اعمال کنترل به کاربر این قدر سریع انجام میشود که کاردیگری نمی تواند انجام دهد.ولی به صورت تئوری کارانتقال کنترل به کاربراین قدرسریع انجام می شود که کاربر به نظر می رسدکه دفعتآ کار کنترل را به عهده می تواند بگیرد. ولی در واقعیت حدود چند ثانیه ای طول می کشد.

ولی در Preemptive کاربر،براستی و واقعآ دفعتآ کنترل را به دست می گیرد .چون فعالیتهای کاربر دارای حق تقدم بالاتری است در نتیجه کنترول به او واگذار می شود .

حال می پردازیم به خصوصیات ذیگر سیستم عامل ویندوز NT بنام symmetric multipocess (چند پردازشگر متقارن)

symmetric multipocess-٤

تمام کامپیوتر های desk top، دارای یک پردازنده یا CPU هستند .می توانیم از CPUها ،از INTEL 80386 وهمینطور از MIPS4000 نام برد . یک CPU می تواند در زمان یک threadرا اجرا نماید و این قدر بین آنها تغییر وضعیت می دهد که کار بر احساس می کندچند thread رااجرا می کند . جاییجایی بین thread ها خود پردازشی است در پردازنده که احتاج به زمان دارد .هرچه threadهای بیشتری برای زمان پردازنده رقابت کنند ، منابع پردازنده درگیر در جابجایی بین thread ها قسمت بزرگتری از زمان قابل دسترس را مصرف میکنند و یا پردازنده دریافت thread های جدید را متوقف می کند ویا به هر یک از thread ها آن چنان زمان کمی می رسد که همه چیز در یک haltاز بین می رود. در اواخر دهۀ هفتاد ،طراحان تعداد پردازنده سه برابر در خواستها زیاد افزایش دادند (muliprocceing).با دو پردازنده ،دو برابر در خواستهاوبا سه پردازنده سه برابر در خواستها را می توانید انجام دهید وبه ... همچنین یک کامپیوتر که بیش از یک پردازنده داشته باشدمی تئاندد یک زمان چندthreadرابدون جابجایی کارها اجرا کند .همیشه میزان زیادی از نیروی پردازش صرف مدیریت فعالیتهای پردازنده های چند گانه می شود .

MULTIPROCESSING*

اگر در کامپیوتری بیش از یک پردازنده استفاده شود اصطلاحا می گوییم muliprocceing که این پردازنده ها با یک حافظۀ مشترک یا high –speed- link به هم متصلند .به کامپیوتری که از این سیستم استفاده می کنند میگوییم Multipoccesssor

LOAD BALANCING:در صورتی پردازنده ای اشغال باشد ،کارها برای افزایش در خواستها می توانند به سایر پردازتده ها منتقلآ شوند این خصوصیت load balancingگفته می شود .

انتقال کارها به پردازنده با توجه به حق تقدم صورت می گیرد .کارهای با حق تقدم بالاتر به پردازنده با ظرفیت کمتر و کارها با حق تقدم پاینتر می توانند در CPUهای مشغول تروشلوغت و منتظر نوبت خود باشند .

سیستم های دارای چند پردازنده نیز دارای مشکلاتی بودند .مثلآ یکی از آنها این است که بدون توجه به تعداد پردازنده میبایستی کپی از بانک اطلاعاتی(data base) داشته باشیم ولی اگر ما در حال اجرای یک برنامۀ پردازش تصویر باشیم،ترجیح می دهیم که فقط روی یک کوپی از تصویر ،بجای یک کپی برای هر پردازنده کار کنیم. داشتن کپی هایمتفاوت از یک data برای چند پردازنده فاقد کارائیست .

داشتن یک کپی نیز خطر می باشد. فرض کنید مثلآ پردازندۀ Aیک رکورد را از بانک اطلاعاتی بخواند و آماده شود که تغییراتی در آن دهد .پردازندۀ دیگری مانند Bهم همان رکورد را از بانک اطلاعاتی می خواند و تغییرات متفاوت در آن می دهد . بعد ، دو پردازنده Aو B رکوردهای تغییر یافته را (هر دو را)در بانک اطلاعاتی باز نویسی می کنند .اول پردازنده Aرکورد تغییر یاقته را می نویسد وبعد Bرکورد تغییر یافته را باز نویسی می کند. بدون سیستم حفاظتی نمی توان از درستی بانک اطلاغاتی اطمینان داشت و نمی توان فهمید که در حال آخرین نسخۀ جاری رکورد میباشیم.

MULTIPROCESSING متقارن و نا متقارن

سیستم عامل باید معمولاً دارایmultiprocessing باشندو تقسیم کار در بین پردازنده ها را خوب باید بداندو دارای سیستم حفاظت باشد تا از درستی dataاطمینان کامل کسب کند.

ما دو نوع سیستم عامل چند پردازنده داریم :۱-متقارن ۲- نا متقارن

در سیستم عامل چند پردازنده از نوع متقارن ،سیستمهای متقارن روی یک پردازندهٔ انحصاری اجرا می شوند و سایر کارهای سیستم عامل روی دیگر پردازنده ها اجرا می شود .تمام ورودی خروجی ها (I/O) توسط پردازندهٔ اجرا کنندهٔ سیستم عامل ، مدیریت می شود .سیستمهای متقارن ،سیستم عامل را روی یک پردازنده از پردازندها اجرا می کنند . ویندوز NT از یک مدل پردازنده متقارن استفاده منی کند .سیستمهای متقارن در مقایسه با سیتمهای چند پردازندهٔنامتقارن که به یک پردازنده فرمان می دهند که سیستم عامل رااجرا کند ، سیستمهای متقارن قابل اطمینان بیشتری دارند چرا که برای خرابی یک پردازندهٔ واحد بر خلاف بهم ریختن تمام سیستم است.

مزیتهاو عیوب سیستمها ی چند پرداشی نا متقارن :

اولین مزیت این است که راحتتربه انجام می رسد . برای افزایش در خواست ویا کارها تعداد زیادی پردازندهٔ خادم اضافه می کنیم.این سیستمها RAMرا در اختیار هر پردازنده قرار می دهدو برای درستی و یا بادرستی داده این کار انجام می شود .پردازندهٔ پایه ممکن است برای حمایت پردازنده های دیگر احتیاج به تغییراتی داشته باشد .

معایب سیستم های متقارن :

از بز رگترین عیوب این است که شدیدا در معرض شکستن هستند یعنی خطر شکستن برای آنها وجود دارد .اگر یکی از پردازنده های خادم خراب شود پردازندهٔ مخدوم می تواند کارها را بین پردازنده های باقیمانده تقسیم کند وشیفت دهد وبه این ترتیب سیستم را سر پا نگه می داردولآی اگر پردازندهٔ مخدوم خراب شود متوقف می شود و در واقع تنها کپی سیستم عامل متوقف میشود وتمام دستگاههای I/Oکه توسط سیستم عامل کنترل می شوند ،غیر قابل دسترس می شوند. پردازندهٔ خادم هنگام افزایش درخواست در سیستم عامل، میتواند دچار اشکال شود .در زمان اتفاق این حالت کار سیستم کند می شود حتی زمانی که بعضی از مخدوم ها کمی ظرفیت داشته باشند . کار با سیستمهای متقارن مشکل است چون فضای زیادی از حافضه را اشغال کرده است در نتیجه برای درستی ویا نا درستی داده نیاز به سیستم محافظ بیشتری دارد .

سیستم عامل بایدبتواند به طورهمرمان روی چند پردازنده اجرا شود بدون اینکه روی خودش بلغزد و باید بتواند در حال اجرا شدن روی هرپردازنده ای،هر پردازندهٔ خراب ویا از کار افتاده ای به خارج بفرستد .

مزایای سیستمهای متقارن :

۱-کارایی بیشتری دارند برای اینکه هم سیستم عامل و برنامه های کاربر میتوانند بین پردازنده ها تقسیم شود . امکان اینکه پردازنده ای مشغول باشد وپردازنده ای بیکار مانده باشد وجود ندارد چرا که در خواستها بین تمام پردازنده ها تقسیسم می شود .

۲-قابل اطمنان تر هستند چرا که خرابی یک پردازنده موجب از کار افتادن تمام سیستم نمی شود . آنها وابسته به MASTER/SIAVE که سیستم های متقارن را بین پردازنده ها قابلاً حملاً میسازد ،نیستتند.

THRAD

در ویندوز NT یک پردازش شامل دستور العمل ها ، فضای آدرس دهی مجازی که برای نگهداری برنامه لازم است ،داده و منابع سیستم عامل که توسط برنامه های در حال اجرا به کار رفته اند، می باشد. در هر پردازش حداقل یک thread وجود دارد و قسمتی است که معمولاً ویندوز NT برای اجرا، زمانبندی می کند . هر پردازش یک thread نیازمند است چرا که بدون thread هیچ ازرشی ندارد . در هنگام اجرای thread است که پردازش هم صورت می گیرد.اگر چه در ویندوز NT در یک زمان می شودچند برنامه را با هم انجام شوند و این بدین معنی است که هر پردازش بیش از یک thread دارد .

برنامه پایگاه اطلاعاتی را برای آ ژانس های مسافرتی در نظر می گیریم این برنامه آثار رکوردهای سرویس گیرندۀ آژانس را روی یک کامپیوتر محلی حفظ میکند وبرای به دست آوردن اطلاعات پرواز برای سرویس گیرنده ، به یک سیستم رزرواسیون از طریق شبکه یا مودم وصل شود .یک شرکت مسافرتی می تواند برای سرویس گیرنده در خواست اطلاعات بکند و فایل مسافرتی سرویس گیرنده را صدا زند .اتصال شبکه ای به سیستم رزرواسیون برای ایجاد و تکمیل ،زمان بیشتری لازم دارد تا دریافت فایل از هارددیسک اصلی .اگر این درخواست به صورت یک thread نوشته شود ، شرکت باید قبل از دستیابی به فایل مسافرتی انقدر صبر کند تا اتصال رزرواسیون انجام شود و اطلاعات مورد در

خواست انتقا ل داده شود گر چه برای یک درخواست چند thread،برنامه مـی توانـد دسـت یابی به یروفایل سرویس گیرنده را هر زمان که شرکت مسافرتی اتصال شبکه ای بـه سیسـتم رزرواسیون برقرار کرد ، آغاز کند ،در حالی که یک thread منتظر پاسخ رزرواسـیون اسـت ،thread دیگر پروفایل سرویس گیرنده دریافت و نمایش می دهد.

ویندوز NTهم می تواند این thread های مجزا را انجام دهد وهـم مـی توانـد thread را بـه پردازنده های گوناگون در سیستمهای چند پردازنده ای بفرستد. این thread های چندگانه به طورهمزمان با هم اجرا می شوند .

یک برنامه با ید نوشته شو دتا این thread های چند گانه را باهم ادغام کند . اکثـر برنامـه هـا که در سیستم چند پردازنده ای قابل اجرا هستند ویـک thread دارنـد ، بسـیاری از امکانـات اجرایی نهفته و پنهان سیستم را از بین میبرند .در بعضی از جاها یک پردازش با یـک thread می توانند سرعت پردازنده های دیگر را پایین بیاورد این مسئله معمولاً در سیستم های شبکه ای که پردازش های اجرا شونده روی پردازنده های یک منبع مشترک مانند مـدیریت شـبکه را بکارمی برند،واقعیت دارد اگر مدیریت شبکه چند thread باشـد ، بازیـابی مـی توانـد بـین thread هـایی کـه روی چنـد پردازنـده اجـرا مـی شـوند ماننـد تقاضـا بـرای افـزایش منـابع مدیریت،تقسیم شود .اگر یک طرح تکthread بکار رود ، مدیریت شـبکه مـی توانـد تنهـا روی یک پردازنده کار کند . با افزایش تقاضا، پردازنـده تضـعیف مـی شـود وبـرای سیسـتم موجود اشکالاتی پیدا می شوند .

حالا نوبت آن رسیده است که از خصوصـیت دیگـر ویـنـدوز NT (خصوصـیت پـنجم)نـام ببرید و آن client/server است که در صفحهٔ بعد توضیحات آن داده خواهد شد.

۵- ساختار CLIENT/SERVER

خیلی ها اصطلاح client/server را با شبکه های محلی و یا lan بکار می برند ما در اول شرح اصطلاح می پردازیم سپس در مورد چگونگی کار ویندوز NT که چگونه طرح client/server را از شبکه به سیستم عامل منتقل می کند.

برای درک بهتر متوسل به مثالی می شویم:

فرض کنید شما در شرکتی که کار می کنید در اتاق خود یک PC که در آن یک دیسک قرار دارد و یک چاپگر که متصل به کامپیوتر است و همچنین مودم، دارید، این PC برای شما یک نفر در هر زمان بخوبی کار نخواهد کرد اگر شرکت افراد استخدام شده خود را افزایش داد به هر نفر یک PC تعلق می گیرد. حال باید تصمیم بگیریم که ایا شما می توانید به هر کدام این افراد جداگانه یک چاپگر و یک مودم بدهید.

مسلم است که این کار پر هزینه استو بعلاوه چاپگر شما که اغلب بلا استفاده است چه لزومی دارد که برای هر نفر هم یک چاپگر خریداری شود.

سؤال دیگر که پیش می آید این است که ممکن است افراد دیگری هم در شرکت به فایلهایی که در هارد PC شما است احتیاج داشته باشد یکی از راهها این است که این فایلها روی فلاپی دیسک Save شود.

وقتی هر کدام از آنها بخواهند فایلی را کپی کنند یعنی پرینت بگیرند اول آن فایل را روی فلاپی دیسک می برد و بعد فلاپی را درPC شما قرار می دهد و توسط چاپگر شما می تواند پرینت بگیرند. در خیلی از سازمانها و شرکتها از این روش که به sneaker net معروف است استفاده میکنند ولی معایب و مشکلاتی دارد که در زیر شرح می دهیم :

۱- سودی ندارد

۲- ایمنی ندارد

۳- دارای دادهٔ غیر ضروری هستند

۴- امکان اینکه داده تحریف شود وجود دارد

۵- ...

بهترین راه حل برای این مشکل استفاده از شبکه ها است. در سیستم‌ای که از شبکه استفاده می کند تمام PC ها با هم در ارتباط هستند و تمام دستگاههای ورودی و خروجی با هم به اشتراک گذاشته می شود از جمله چاپگر و مودم. در این سیستمها که از شبکه lan استفاده می شود هر فردی می تواند از فایلهای شما استفاده کند بدون اینکه به کار شما صدمه و وقفه ای وارد آورد اینگونه شبکه بنام peer-to-peer معروف است. هنگامی که فردی بخواهد در شبکه فایلی را پرینت بگیرد، کار شما در PC آهسته تر انجام خواهد شدکه همزمان با اینکه آن باید کار شما را انجام دهد باید درخواست دیگری را هم پردازش کند. در شبکه های کوچکتر بر خورد درخواست به دلیل اشتراک منبع کمتر صورت می گیرد یعنی درواقع فرقی ندارد که PC چه کسی به چاپگر متصل می‌شود ولی هر چه شبکه بزرگتر باشد، تاخیر افزایش بیشتری میابد سرعت PC ها کمتر می‌شود چرا که تعداد افرادی که می‌خواهند به فایلهای مشترک دسترسی پیدا کنند بیشتر می‌شود. به همین دلیل لازم است که شبکهٔ peer-to-peer تکمیل شود برای همین بایدیک کامپیوتر را در نظر بگیریم که فایلهای مشترک را مدیریت کند. به همین دلدل است که از client/server استفاده میشود. فایلهای مشترک، ازPC شما به سرویس دهنده فایل یک کامپیوتر منتقل شود. منظوراز سرویس دهنده فایل

یک کامپیوتر انحصاری است که کارش ایت است که امکان دستیابی مساوی به سیستم فایل مرکزی را برای کاربران فراهم می کند. هر کس که فکر می کند در PC خود فایلهایی دارد که بقیه نیز به آن نیازمند است به سرویس دهنده فایل، این فایلها را منتقل می کند و بدین ترتیب کارایی PC افزایش میابد.

پس بطور کلی می توان گفت که سرویس دهنده فایل یک کامپیوتر با یک دیسک سخت با ظرفیت بالا است که به شبکه محلی وصل می‌شودوفایلهای بکار رفته توسط سایر کامپیوترهای متصل به همان شبکه را ذخیره می کند منظور از سرویس گیرنده از هر شبکه، PC شما و PC هرکاربردیگر است. درخواستهای سرویس گیرنده از یک سرویس دهنده منشاء می گیرد مانند فایلهای دیتا و دستیابی به چاپگر.

CUENT/SERVER در ویندور NT

ویندوز NT در دو لایه کار می کند یکی لایۀ کاربر و دیگری لایۀ Kernel با اجرای ویندوز NT تمام کارهای مدیریت مثل مدیریت ورودی و خروجی، حافظۀ مجازی تمام پردازش‌ها را انجام می دهد.

اجرای ویندوز NT در لایۀ Kernel انجام می‌شود چرا که دارای امنیت خاصی است و از دسترسی به پردازش های کاربر جلوگیری می کند. درلایۀ کاربر ویندوز NTموارد چندی از سیستمهای حفاظت شده را فراهم می کند برنامه نویسی win32 (API) مثالی از این مورد است کاربران به توسط win32 می توانند به لایۀ user و دیگر امکانات ویندوز NT دسترسی پیدا کنند. به توسط این برنامه نویسی، سازندگان بدون اینکه در مورد تفاوت سیستمهای

سخت افزاری NT نگران باشند می‌توانند یک درخواست یکسان را در کامپیوترهای مختلف اجرا کنند.

زیرا سیستم امنیت از جمله زیر سیستمهای محافظت شمرد می‌شود که دارای روش‌های مختلف محافظت و همچنین Password محافظتی است. DOS و درخواستهای ویندوز ۱۶ بیتی را می‌توان در فضایی از جنبهٔ کاربر که بنام virtual Dos Machine معروف است، مورد بررسی قرار داد.

virtual Dos Machine نیز با زیر سیستم win32 در ارتباط است. برنامه های درخواست در ویندوز NT سرویس گیرنده و زیر سیتمهای حفاظت شده، سرویس دهنده هستند. توسط اجرای NT این درخواستهای سرویس گیرنده پیغام خود را به زیر سیستمهای حافظت سرویس دهنده، که فضای مشترکی برای سرویس دهنده‌ها ایجادکرده، می فرستند. سرویس دهنده‌ها توسط همین فضا، پاسخ سرویس گیرنده‌ها را می دهند.client/server با حذف منابع یکسان، سیستم اجرایی را جهت می دهند. زیرا اگر اجرای NT بخواهد همراه API باشد سیستم خیلی دشوار می‌شود. مزیت دیگر این ساختار این است که بدون هیچگونه اجرای جدیدی می‌توانید به API اضافه کنید. به علاوه هر زیر سیستم در حافظه حفاظت شدهٔ خود دارای مراحل عمل مخصوص به خود است. در نتیجه اگر یکی از زیر سیستم دچار مشکلی شود، کار کلیه سیستم NT خراب نمی‌شود. طراحی مراحل اجرایی زیر سیستم بگونه ای است که هنگام کار قادر به تبادل اطلاعات متعدد و در نتیجه شکوفا کردن ظرفیت NT Multipracessor می‌شود. ساختارclient/server ، طراحی ویندوزNT را به صورتی خلاصه کرده، تا به عنوان بهتر کردن سیستم های اجراییMultipracessor و حمایت

اطلاعاتی گسترده شبکه، API های جدیدی در هر زمان دلخواه به سیستم اضافه کرد. همچنین کاربران زیرسیستم حفاظت شده می‌توانند به وسیلهٔ پردازندهٔ یکسانی با دیگر کاربران ارتباط برقرار کنند که این خود سبب کارایی بیشتر سیستم می‌شود. امنیت در سیستم ویندوز NT دارای دو بعد است و این دو بعد:

۱- کنترل بر چگونگی دستیابی به سیستم مرکزی NT و همچنین کنترل از جهت دستیابی به پرونده ها و subdirectoryهای داخل سیستم.

۲- حفاظت عملکرد کلی و جزء به جزء مراحل عمل سیستم، به جهت جلوگیری از مختل کردن کار کل و یاجزء به جزء سیستم بوسیلهٔ ویروس و دیگر برنامه های ایجاد شده جهت خراب کردن کار سیستم.

در سیستمهای شبکه، یک وقفه ممکن است دچار بسیاری از مشکلات شود یعنی مشکلات زیادی رافراهم کند چرا که باعث می‌شود که ویروس دهنده متوقف شود و همین توقف سرویس دهنده بر روی تمام در خواستهای اجرا شده روی سرویس دهنده تاثیر می‌گذارد و باعث می‌شود بسیاری از فایلهاکه ذخیره شده اند از بین بروند. ویندوز NTمزایای گسترده ای برای اطمینان از امنیت سیستم دارد تا بتواند سیستم‌های NT را در شرایط سخت نگه دارد و اشکالات سیستم رابهبود دهد و رفع کند.

خصوصیت کنترل دستیابی

سیستم‌های عامل DOS و ویندوز۱۶ بیتی دارای خصوصیت امنیت مانند ویندوزNT نیستند. چرا که هر PC متعلق به یک کاربر بوده و توسط همان یک نفر مورد استفاده قرار می‌گیرد و دیگر احتیاجی به password ندارد و همچنین هیچ گونه محدودیتی در استفاده از فایلها

ندارد. سیستم عامل‌های شبکه و مدیریت LAN (همچون شبکه فایل) برای دستگاههایی طراحی می‌شوند که ذر آنها یک کامپیوتر یا سیستم فایل میان تعداد زیادی کاربرمشترک است. این سیستمهای عامل محدوده ای از امکانات امنیتی را برای اجازه دادن Data به مشترک بودن بین کاربران و درخواستها بکار می‌برند در حالی که دستیابی به فایل شخصی و حساس را محدود می کند.

برای همین است که ویندورNT یک سیستم حفاظت و امنیت کنترل دستیایب را پیش بینی می کند هدف‌های ویندوز NT از سیستم حفاظت و امنیت این است که از اینکه فقط کاربران مجاز به اطلاعات سیستم دسترسی پیدا کند اطمینان حاصل کند.

مقدمه

xinu سیستم عاملی است که روی دستگاههای ۱۶ بیتی LSI11.2 و سازگار با آن قابل اجراست. این سیستم عامل قابلیت اجرای چند برنامه و شبکه را دارد. برنامه اصلی آن به زبان C و ۵۵۰ خط کد زبان اسمبلی است (بدونComment)

:System _ Calls

در sinu حدود ۳۳ سیستم call وجود دارد که در اینجا به ذکر چند نوع از آنها که در بیشتر سیستمهای عامل مشترک هستند،می پردازیم :

برای اجرای هر پروسه ای دو نوع sys-cal داریم، یکی create و دیگری resume میباشد.

Create: این sys-cal یک پروسه را به حال اجرا در می آورد و اگر پروسه ای در حال اجرا باشد و این sys-cal را اجرا کند خودش موقتاً به حال تعلیق میرود و پروسه ای را از حالت ready به حالت اجرا (current) می برد.

resume: وقتیکه یک فرآیند فرزند کارش تمام شود ، این sys-cal اجرا می شود تا فرآیند پدر که در حالت تعلیق (suspended) بود به کار خودش ادامه دهد.

Kill: در حقیقت این sys-cal برعکس cerate دمی باشد kill ، کار یک فرآیند را فوراً خاتمهن می دهد و کلیهٔ مشخصات آن را (preocess Identification) از Process-Table پاک می کند. اگر فرآیند در صف Semaphore باشد ، از این صف نیز خارج شده و

شمارهٔ semaphore یکی افزایش پیدا میکند . اگر فرآیند منتظر L/Q باشد ، کار L/Q در صورت امکان قطع می شود.

در xinu برای اجتناب از Busy wating از روش semaphore &producer (Consumrt) استفاده می شود، در این حالت ما دو sys-cal داریم به نامهای Wait و Singnal .

Wait: این sys-calیکی از عدد semaphor کم می کند، اگر جواب منفی بود موجب می شود که پروسه به تاخیر بیفتد (به حالت wait میرود)

Signal: این sys-cal دقیقاً برعکس wait عمل می کند ، یعنی یکی به semaphor اضافه می کند. اگر عدد مثبت بود به پروسه ای که در حالت انتظار بود (waiting) اجازه میدهد به حالت ready برود.

ساختار سیستم عامل xinu:

سیستم عامل xinu یک سیستم عامل لایه ای می باشد که از ۱۰ لایه تشکیل شده است که به ترتیب اهمیت عبارتند از:

۱) سخت افزار ۲)مدیریت حافظه

۳) مدیریت پردازش ۴) هماهنگی فرآیندها

۵) روابط بین فرآیندی ۶) مدیریت clock

۷) مدیریت I/Q ۸) لایه شبکه

۹) سیستم فایل ها ۱۰) برنامه های کاربر

زمانبندی و سوئیچ کردن(Scheduling & Context Switching):

Process State: یکی از فیلدهای ۱۲ گانهٔ Process Table می باشد

در xinu ما برای فرایندها ۶ حالت داریم :

1.Current 2.ready 3.wait 4.sleeping

5.receiving 6.suspended

تقریباً تمام سیستم ها به دو حالت ready و current نیاز دارند.

Current: حالتی است که یک پروسه یا فرآیند ، زمان cpu را دریافت کرده و در حال اجرا است .

تعریف زمانبند (scheduler) :عمل سوئیچ کردن از یک پروسه به پروسه دیگر شامل دو چیز است :

الف –انتخاب یک پروسه از میان پروسه هایی که در حالت ready هستند.

ب –واگذار کردن cpu به فرآیند انتخاب شده

نرم افزارهایی که این عمل را انجام می دهند ، اصطلاحاً (زمانبند) یا "scheduler" نامیده میشوند.

برای اجرای فرایند ها بطور همزمان نیاز به یک Process-Tabld است که اطلاعات هر فرآیند در آن ذخیره شود. در xinu این (جدول فرآیندها)شامل ۱۲ فیلد است ، این فیلدها عبارتند از :

۱) اولویت فرآیند ها ۲) semaphor ها ۳) حالت فرآیند ها (porcess state)

۴) مقادیر ثبات ها ۵) پیغام های دریافتی ۶) طول)(stack

۷) اسم فرآیند ۸) آدرس کد ادامه برنامه و ....

priority: در xinu اولویت هر فرآیند داخل فیلدی در process-T نگه داری می شود . در xinu پروسه در حال اجرا بالاترین اولویت را دارد. پروسه های که دارای اولویت برابر هستند به روش round-robin در یک صف قرار می گیرند . اولویت پروسه به ترتیب ، از صفر به بالا افزایش می یابد ، یعنی کمترین اولویت مقدارش صفر است که مخصوص فرآیندی بنام Null-Process می باشد.

فرآیند پوچ (Null Process):

برنامه زمانبند (scheduler) فقط می تواند زمان cpu را بین فرآیند هایی که در حالت current و ready هستند ، سوئیچ کند . اما نمی تواند خودش یک فرآیند جدید بسازد . بنابراین همیشه فرض بر این است که یک پروسه در دسترسی وجود دارد در نتیجه باید همیشه ، حداقل یک فرآیند در صف فرآیند های ready وجود داشته باشد. در xinu هنگامی که سیستم راه اندازی می شود یک فرآیند اضافی (Extra Process) ساخته می شود که Null Process نام دارد. این فرآیند دارای درجه اولویت صفر می باشد و شامل یک حلقه بینهایت است.

Process Suspension& Resumption

Suspend: توقف یک پروسه یا فرآیند بطور موقت

Resume: شروع دوباره فرآیند

Suspend State: یکی از ۶ حالت فرآیند هاست که با دو سیگنال suspend و resume در ارتباط است و به این معنی است که : (منتظر ماندن یک پروسه برای فراهم شدن یک یا چند شرط بدون توجه بهن اینکه این شرایط کی فراهم می شوند.)

هماهنگی فرآیند ها (Process Coordination)

برای هماهنگ کردن فرآیند ها از semaphor استفاده می شود. برای این کار ما دو روال wit و signal داریم:

wait: یکی از مقدار S کم می کند.

signal: یکی به مقدار S اضافه می کند.

اگر مقدار S منفی شود، فرآیند مربوطه به حالت wait می رود. اگر روال signal فراخوانی شود و مقدار S مثبت باشد آنگاه فرآیندی که در حالت wait است به حالت ready باز می گردد. اگر هیچگاه روال signal فراخوانی نشود ، فرآیند برای همیشه (forever) در حالت wait می ماند . قابل توجه است که فرآیندی که در حال wait است هیچ دستوری را نمی تواند اجرا کند . بنابراین هنگامیکه تمام فرآیند ها به حالت wait بروند دیگر سیستم نمی تواند هیچ کدی را اجرا نماید.

بخاطر اینکه سیستم هیچ وقت به حالت Halt نرود همیشه یک فرآیند در حالت ready باقی می ماند که آن فرآیند ، همان فرآیند null-process می باشد.

Interprocess Communication

ارتباط بین فرآیند ها توسط ارسال پیغام (message passing):

در xinu دو روش ارسال پیغام (message passing) وجود دارد که فقط یکی از آنها را توضیح خواهیم داد ، روش دوم برای شبکه ها استفاده می شود.

پیغام های بین فرآیندی (process to process message passing) :

در xinu برای ارتباط بین فرآیند ها از روش (ارسال پیغام) استفاده می شود که به این صورت است که سیستم عامل اجازه می دهد یک فرآیند برای فرآیند دیگر اطلاعاتش را بفرستد. این اطلاعات یا پیغام ها فقط بین فرآیند هایی ارسال می شود که در حالت Block نباشند.

در xinu ، در هر زمان فقط یک پیغام قابل دریافت است و اگر چند پیغام همزمان ارسال شود ، سیستم فقط اولی را دریافت کرده و به بقیه جواب نمی دهد.

روال send ، پیغام یک ID و آرگومان یک پروسه را می گیرد (پروسه ای که قرار است پیغام به آن فرستاده شود) سپس پیغام مذبور را به آن پروسه خاص تحویل می دهد و همچنین هنگام اجرای روال send ، فرآیندی که پیغام را دریافت می کند به حالت ready باز می گردد تا کاوش را ادامه دهد.

روال receive منتظر می ماند تا یک پیغام دریافت شود تا آن را به رای فرآیند مورد نظر بفرستند. هنگامی که این روال یک پیغام را دریافت کرد فیلد phasmsg در process-table چک نمی شود اگر فرآیندی منتظر دریافت باشد این فیلد (۱) است سپس پیغام دریافتی را برای فرآیند مورد نظر می فرستد.

مدیریت حافظه (Memory Management) :

برای مدیریت حافظه اصولاً از دو روش swapping و paging استفاده می شود .

روش paging به این شکل است که اگر یک پروسه نیاز به این داشته باشد که صفحه بعدی اش به حافظه وارد شود. تست می شود که آیا جا برای swap کردن هست یا خیر؟ اگر فضای لازم در دسترس نبود، process به حالت تعلیق (suspend) می رود و هنگامی که فضا برای swap کردن پیدا کرد آن را load می کند و process مربوطه را دوباره فرا می خواند (resume)

در xinu از روش paging استفاده نمی شود و کل برنامه یکجا در حافظه می نشیند .

در xinu همیشه قفل برنامه در قسمت low حافظه می نشیند و stack در قسمت high آدرس می نشیند . xinu قابلیت paging ندارد، به همین دلیل کل برنامه در هنگام اجرا در داخل حافظه می نشیند و تا کارش بطور کامل تمام نشده باشد، تمام فضایش را اشغال شده نگه می دارد.

در xinu اگر stack سریز (eveflow) کند هیچ راهی برای برطرف کردن آن پیش برمی نشده و بخاطر همین مسئله سیستم هنگامی (overflow) بحالت halt می رود. در xinu لیست فضاهای خالی بر حسب آدرس تنظیم میشود به این معنی که بلوکهای خالی حافظه در یک linklist بنام (memlist) نگه داشته میشوند. این لیست به ترتیب« صعودی آدرسها » مرتب شده است .

هرعنصر این linklist شامل دو قسمت است : یکی اشاره گری به عنصر بعدی و دیگری شامل سایز آن قسمت از حافظه است .

برای stack هم همین مکانیز را داریم فقط فرق در این است که چون stack قسمت high address می نشیند ، بنابراین linklist بلوکه های آزاد به ترتیب نزولی آدرسها حافظه مرتب میشوند (این روش تا حدی مشابه روش hirstfit است)

Interrupt Processing:

قبل از اجرای یک دستور العمل cpu خط Interrupt را چک میکند.اگر این خط فعال بود یک روالی برای کردن وقفه handle فراخوانی می شود و هنگامی که کار وقفه به پایان رسید ، کنترل به فرآیندی که در حال اجرابود (قبل از اینکه وقفه رخ دهد) برگردانده می شود و فرآیند بقیهٔ کارش را ادامه می دهد.

همهٔ وقفه ها به روتین هایی به نام interrupt dispatch انشعاب میکنند. Dispatcher ها اعمالی مانند ذخیره و بازیابی مقادیر ثبات ها ، تشخیص وقفه دستگاهها و عمل بازگشت از یک روتین وقفه ، هنگامی که رویتن وقفه کارش به پایان رسیده را، انجام می دهد. xinu ،شامل سه نوع مختلف از Interrupt Dispatcher می باشد. یکی برای handle کردن .clok Int و دیگری برای وقفه های دستگاههای ورودی (Input) و سوم برای دستگاههای خروجی (output) می باشد.

تمام دستگاههای ورودی به یک سری از dispatchroutine ها انشعاب کنند در این حالت ، dispatcher از کجا می فهمد که کدام روتین باید فراخوانی شود ؟ راه حل این مشکل به این شکل است که انتخاب یک Interrupt handتوسط و آدرس وسیلهٔ درخواست کنندهٔ وقفه مشخص میشود.

برای تشخیص اینکه کدام دستگاه درخواست وقفه کرده و حال باید کدام روتنی را اجرا کند، سیستم عامل xinu به این شکل عمل می کند که ، سیستم عامل کمله (word) دوم از .Int.Vector راکد برداری (Encode) می کند تامشخص شود که کدام وسیله و نوع وقفه چیست .

هنگامی که یک وقفه رخ می دهد cpu،ثباتهای PC و SP را از روی Int.Vector بار میکند و شروع میکند به اجرایInt.Voutine مورد نظر .

:Input Output Management

در xinu هر دستگاه جانبی با یک عدد صحیح که بعنوان Device Descriptor یا توصیف گر دستگاه است در هنگام راه اندازی اولیه دستگاه ، متناظر می شود. یعنی برای هر دستگاه جانبی سیستم عامل عدد صحیحی را به عنوان مشخص کنندۀ آن دستگاه انتخاب می کند.

در زمان اجرا (Runtime) برنامه ممکن است یک روتین I/Q را صدا بزند (مانند read یا putc) در این موقع devicedesriptor به عنوان یک آرگومان برای قسمت I/Q routine فرستاده می شود. I/Q routine ها از این عدد صحیح (devicedesriptor) به عنوان اندیسی برای یک جدول بنام جدول انتخاب دستگاهها (device switch table) استفاده می کند. این جدول هر عدد صحیحی را به آدرس یک دستگاه واقعی (real device) نگاشت (Map) می کند. یعنی با داشتن یک عدد می توان آدرس دستگاه خاص را در devtab (device table) به دست آورد . هر عنصر جدول devtab متناظر با یک دستگاه است که شامل آدرس روتین devicedriver های مخصوص آن دستگاه خاص

است و همچنین آدرس خود دستگاه و یکسری اطلاعات دیگر که برای driver ها لازم است .

تنها دانستن آدرس روتیش device driver کافی نیست زیرا چند دستگاه جانبی می تواند مشترکاً از یک روتین استفاده کنند. بنابراین device table شامل فیلدهای برای آدرس دستگاه سخت افزاری (Hardware Device)، همچنین آدرس Int. Vector و روتین Int. Dispatchمی باشد.

:Booting XINU

سیستم عامل xinu یک سیستمی که بطور مستقل روی دیسک مقیم باشد، نیست در حقیقت این سیستم عامل می تواند روی ماشینی که دیسک ندارد اجرا شود، برای اینکه این سیستم عامل توسط یک کامپیوتر دیگر که اصطلاحاً کامپیوتر میزبان (Host) نامیده می شود ، روی دستگاه بار می شود. (Down Ioad) .

عمل boot شدن بطور کلی به این ترتیب است که : کامپیوتر میزبان یک Condition Break تولید می کند و cpu کامپیوتر 11/2 را به حالت (Halt)می برد. سپس کامپیوتر میزبان یک برنامهٔ Initial Boot را روی آدرس صفر سیستم 11/2 بار می کند پس از آن سیستم 11/2 شروع به اجرای این برنامه که روی حافظه بار شده می کند. سپس کامپیوتر میزبان برنامه boot دوم را روی قسمت High حافظه بار می کند . سپسدوباره سیستم 11/2 این برنامه دوم را که در قسمت High حافظه نشسته اجرا می کند. در این قسمت سیستم عامل xinu شروع به اجرا شدن می کند که مکان شروع آن در بخش (01000 octal) می باشد.

File System:

در این سیستم عامل ، دیسک به سه قسمت : دایرکتری –اندیس –قسمت اطلاعات (Data) تقسیم میشود.

« فایل سیستم »زمانی که بخواهد یک فایل را در دیسک ذخیره کند یکسری از بلوکهایی را که استفاده نشده (unused) از Freelist بر میدارد و به فایل اختصاص می دهد و هنگامی که یک فایل پاک میشود ، فضای تخصیص داده شده به آن فایل به « لیست فضای آزاد (Freelist) » برگردانده میشود

جدااز« قسمت داد ه ها اندیس » روی هر دیسک شامل یک مجموعه ای از Index-Block (یا I-block) است . هر فایلی که روی دیسکت ذخیره شده ، برای خودش یک اندیس داردکه شامل یکlinklist یکطرفه از I-bock ها می باشد. هر I-block شامل اشاره گری به یک مجموعه از data block است .

قسمت « اندیس ها (Index)» شامل یک مجموعه ای از I-lock هایی به اندازۀ ثابت است و برای اینکه I-block ها کوچکتر از بلاکهای دیسک هستند، سیستم هر هشت I-block را در یک بلوک فیزیکالی ذخیره میکند.

هر I-block شامل یک آرایه از اشاره گرها به « بلوک داده ها » است . هر آرایه شامل ۲۹ عنصر است که این عنصرها شامل آدرس بلوک داده ها ست که این بلوکها ۵۱۲ بایتی می باشند در نتیجه هر I-block می تواند ۱۴۸۴۸=۵۱۲*۲۹ بایت باشد. هر دایرکتوری باید شامل یکسری از اطلاعات فایل باشد. این اطلاعات شامل اسم فایل – آدرس اولین و

همچنین شامل مجموعهٔ تعداد I-block های روی دیسک و اشاره گری به لیست بلوکهای خالی است .

سیستم عامل MINIX

تاریخچه MINIX

وقتی که unix تازه وارد بازار شده بود (v-6) منبع کدها بسیار فراوان بود البته زیر نظر و مجوز At&T و اغلب مورد استفاده قرار می گرفت . جان کایز استاد یکی از دانشگاههای استرالیا یک کتابچه ای در طرز کار مرحله به مرحله آن نوشت این کتابچه زیر نظر AT&T به عنوان یک کتابچه درسی در خیلی از دانشگاهها به عنوان دوره های عملی استفاده میشد.

وقتی که AT&T (v-7) را دیگر بکار نبرد ، تازه مشخص شد که unix یک محصول تجاری با ارزشی بوده بنابراین (v-7) (جلوگیری کند) بار دیگر چاپ شد با مجوز اینکه از منابع کند در این دو.ره استفاده نشود تا بتواند از خطری که موقعیت و رمز تجاری او را تهدید می کرد جلوگیری کند . خیلی از دانشگاهها unix را کنار گذاشته و به تدریس تئوری اکتفا کردند.

متاسفانه تدریس تئوری باعث میشود شاگردها ایده های متفائتی درباره طرز کار علمی و حقیقی آن خواهند داشت . موضوعات تئوری معمولاً بطور گسترده در دوره ها و کتابها درباره طرز کار سیستم ها گفته میشود از قبیل برنامه ریزی الگوریتم که زیاد مهم نیستند . موضوعاتی که معمولاً مهم هستند مثل مدیریت L/Q و مدیریت حافظه معمولاً بدست فراموشی سپرده میشودند. برای علاج این درد این استاد تصمیم گرفت که طرز کار این سیستم جدید را با استفاده از یک دستخط که با سیستم unix هماهنگی بسیار دارد و همچنین از نظرات استفاده کنندگان و تجربات آنان یکسان بود ، اما کاملاًمتفاوت از لحاظ کار

داخلی سیستم بود استفاده کرده انـد . و حتـی یکـی از کـدهای AT&T را در اینجـا اسـتفاده نکرد.

این سیستم محدودیت پروانه با مجوز را ندارد بنابراین هم برای کلاس و هـم بصـورت فردی می توان استفاده شود . بدین طریق یک خواننده می تواند طـرز کـار سیسـتم را کـاملاً موشکافی کرده تا بتواند داخل آن را ببیند درست مثل شاگردهای زیست که کـاملاً قورباغـه را کالبدشکافی می کننـد و در نتیجـه اسـم minix جـایگزین minunix شـد، بخـاطر اینکـه آنقدر کوچک وساده بود که حتی یک مبتدی هم می تواند با آن کار کند .علاوه بر مزایـای فوق دیگر اینکه مشکلات غیر منطقی را حذف می کند و این مزیت دیگری بـر unix اسـت .و دیگر اینکه یک دهه بعد از unix نوشته شـد وسـاختار اسـتاندارد عقبتـری داردبـرای مثـال سیستم بایگانی minix ابداً جزو قسمتی از طرز کار سیستم نمی باشد بلکه بعنوان یـک برنامـه مجزائی عمل می کند.

فرق دیگر این است که unix طراحی شده کـه خیلـی مـؤثر باشـد امـا minix طراحـی شده که قابل خواندن و رؤیت باشد. minix نوشته شده برای هماهنگی با unix(V-7) minix مثل unix به زبان C نوشته شده است و برای کامپیوترهای شخصی مخصوص IBM است . درنگهداری آن حتی احتیاج به دیسک سخت هم برای شاگردان ندارد.

میانگین شاگردانی که روی IBM با minix کـار میکننـد درسـت مثـل unix میباشـد. اکثر برنامه هایی مفید درون minix مثل make,eat و ...با unix همکاری می کنـد درسـت مثل خود سیستم اصلی .

در اینجا ابتدا به توضیح درباره مدیریت فرایند ، مدیریت حافظه و مدیریت فایل ها بحث خواهیم کرد.

۱-۱ نگاهی به مراحل پردازش در minix

برای کامل کردن مطالعاتمان در مورد کنترل و اداره کردن فرآیند ها حالا می توانیم نگاهی داشته باشیم به اینکه چگونه اینها در minix ظاهر میشوند.

بدون تشابه به unix که هسته مرکزی اش (kemel) یک برنامه .یکپارچه بود و به پیمانه ها شکسته نمی شود ، minix خودش به تنهایی یک مجموعه ای از مراحل پردازش مختلف است که با هم ارتباط برقرار می کنند و از طریق ارتباطات پردازشهای درون تکی ، در پیامهای اولیه ،با پردازش ها ی user ارتباط برقرار می کنند.

این روش به ا طرح و ساختاری خواهد داد که بیشتر پیمانه ای است و قابلیت انعطاف بیشتری را نیز دارد، علاوه بر این باعث آسانتر شدن ساختار هم می شود.

۲-۲ ساختمان داخلی minix

اجازه بدهید مطالعاتمان را راجع به minix به وسیله انجام یک دید دقیق روی سیستم آغاز کنیم .

minix در ۴ لایه ساخته شده است و وظیفه اش را با هر لایه به خوبی انجام می دهد.

لایه پایینی تمام موانع و محدودیتها و وقفه ها و در واقع وام هایی را که وجود دارد را میگیرد وسپس لایه های بالاتر را به وسیله مدلی از مراحل پردازش ترتیبی مستقل (independent) که به وسیله پیغام ها ارتباط برقرار می کنند،آماده می کند.

کندی که در این لایه قرار داردو فعالیت دارد:

اول اینکه وقفه ها و رام ها را میگیرد و سپس باز سازی می نماید و درون ثباتها ذخیره می نماید. و دوم اینکه ساختار مکانیسم پیغام ها را بررسی می کند .

این بخش از لایه ها به وسیله پائین ترین سطح از بررسی وقفه ها ، در زبان اسمبلی رسیدگی می گردد.

بقیه لایه ها درزبان C نوشته شده اند .

لایه دوم شامل مراحل L/Q می باشد،در هر نوع وسیله برای تشخیص آنها از مراحل پردازش user های معمولی، ماآنها را شسن (وظیفه) می نامیم .

اما تفاوت بین Task و process بسیار کم است و در واقع task ها نوعی از process اند. در خیلی از سیستم های L/Q همین task نامیده می شوند. به جای واژه task از driver نیز می توان استفاده کند و در minix می توانیم به جای آن device driver نیز بگوئیم.

هر نوع device یک device driver یا task مخصوص خودش می خواهد تمام task ها در لایه دوم و تمام کدها در لایه اول به هم بوسیله یک برنامه تکی باینری که بنام kernel می شناسیم وصل شده اند.

task های لایه دوم همگی کاملاً بدون وابستگی یا مستقل (independence) هستند و از یکی به دیگری مستقلاً دسته بندی شده اند و ارتباط برقرار می کنند و این ارتباط از طریق پیام ها می باشد.

لایه سوم شامل ۳ مرحله است که اولاً سرویس های مفیدی برای پردازش user تهیه می کند . دوم اینکه مدیریت حافظه را انجام می دهد و سوم اینکه روی سیستم فایل مدیریت

دارد. همانطوری که می دانیم یک operating system (سیستم عملیاتی) دو کار را انجام می دهد:

اداره کردن و مدیریت منابع و تهیه یک ماشین توسعه یافته و به وسیله انجام فراخوانی های سیستم .

در minix مدیریت منابع به صورت وسیع در kernel صورت می پذیرد (لایه ۱ و ۲) و ترجمه فراخوانی های سیستم در لایه ۳ می باشد.

سیستم فایل مانند یک فایل (server) طراحی شده است و میتواند حرکت کند و به ماشین دیگری برود بدون هیچ گونه تغییری .

پس = >که مدیریت فرآیند ها از نوع client server می باشد در نهایت لایه چهارم شامل تمام مراحل user می باشد مثل editor ها ، کامپایلرها و برنامه های نوشته شده user .

۳-۲ مراحل مدیریت فرآیند ها در minix

فرآیند ها در minix از یک سری مدل های فرآیندی پیروی می کنند . فرآیند هامی توانند یک زیر فرآین هایی را به وجود بیاورند و هر کدام از این زیر فرآیند ها ، زیر فرآیند های دیگری ایجاد کنند و در نتیجه یک درختی از فرآیند ها به وجود می آید.

در حقیقت تمام user processe ها در کل سیستم یک قسمت از یک درخت هستند که ریشه آنها (init) است .

برای فهمیدن اینکه چگونه یک چنین موقعیتی به وجود بیاید ، نگاهی به راه اندازی minix در فلاپی دیسک خواهیم داشت .

فلاپی و دیسک

وقتی کامپیوتر روشن است سخت افزار اولین سکتور از اولین شیار را می خواند و می فرستد به حافظه . این سکتور شامل یک برنامه bootstrap می باشد که تمام سیستم عامل رادر حافظه لود می کند و شروع به اجرای آن می نماید .

بعد از kerkel مدیر حافظه و سیستم فایل اجرا و مقداردهی اولیه می شود و پس از کنترل به init می رود. init شروع به خواندن فایل می کند تا ببیند چند ترمینال به آنها وصل است ، سپس شروع به تقسیم process ها برای child ها در هر ترمینال می کند .

هر کدام از این child ها مراحل زیر را دارند :

login را اجرا می کنند و بعد از یک login موفق bin /login اجرا می شود البته درون shell user, shell duser منتظر می ماند برای command ها تا تایپ شود و سپس به فرآیند های جدید برای هر command تقسیم می گردند . در این روش shell نوه های init,children هستند و user process ها نوه هایی نوه ها هستند و تمام process ها قسمتی از یک درخت تکی یا single tree می باشند.

دو سیستم اصلی فراخوانی شده در minix برای مدیریت process ها Fork هستند و exec

FORK: روشی برای ایجاد process جدید می باشد.

EXEC: اجازه اجرای یک برنامه مشخص را به process می دهد.

وقتی یک برنامه اجرا می شودیک بخش از حافظه را که اندازه آن در هیدر فایل (eaer file) برنامه مشخص شده است ، اشغال می کند و این قسمت حافظه در طول اجرا حفظ می کند. تمام اطلاعات مربوط به process ها درون procrss table نگهداری می شود. وقتی

یک process را updare می کند و سپس پیغام هایی به سیستم فایل و kernel می فرستد .و kernel به آنها می گوید که چه کاری انجام دهند.

٤-٢ مراحل جدول بندی شده در minix

یک minix طبقه بندی شده از یک سیستم چند مرحله ای queing استفاده می کند که در ۳ مرحله می باشد که منطبق با لایه های ۳ و ۲ و ۴ می باشد. در هر level یک چرخه طولانی صورت می پذیرد.

task ها بالاترین اولویت را دارند، مدیریت حافظه و server فایل ها اولویت بعدی را دارند و user processes ها آخرین اولویت را دارا می باشند.

وقتی یک process آماده اجرا است ، جدول ما چک می کند که آیا هیچ task ای وجود دارد که آماده باشد؟

اگر یکی یا بیشتر آماده بودند، آنکه در جلوی صف قرار دارد اجرا می شود. اگر هیچ task ای آماده نباشد یک server (ES,MM) انتخاب می شود ، اگر ممکن باشد در غیر این صورت user اجرا می گردد.

اگر هیچ process ای آماده نباشد سیستم یک حلقه را در نظر می گیرد که برای وقفه بعدی آماده باشد.

در هر clock-tick یک چک انجام می شود که ببیند آیا process در جریان یک user process است که دارای اجرایی بیشتر از 100.msec است. اگر چنین باشد جدول مربوط فراخوانی می شود به آخر صف مربوطه و process ای که در head صف ها قرار گرفته اجرا می شود.

task ها ، مدیریت حافظه و سیستم فایل به وسیله clock به دست نمی آیند چون هیچ مسئله ای نیست که چه مدت وقت می گیرند برای ا

۲-۱ نگاهی به سیستم ورودی / خروجی در minix

در بخش های ذیل نگاهی مختصر به هر یک از لایه ها و همچنین تآکیدی بر روی نرم افزار راه اندازی خواهیم داشت . مدیریت وقفه در فصل پیش توضیح داده شد و هنگامی که به مبعث سیستم فایل در فصل پنجم رسیدیم ، سیستم ورودی و خروجی غیر وابسته به وسائل سخت افزاری را توضیح خواهیم داد.

۲-۳ برنامه مدیریت وقفه در سیستم minix

بسیاری از نرم افزارهای راه اندازی ، برخی از وسائل ورودی و خروجی را بکار انداخته و سپس با بلوکه کردن آنها منتظر رسیدن پیغام می مانند ، این پیغام همان طوری که دیدیم توسط برنامه مدیریت وقفه تولید می شود . دیگر نرم افزارهای راه اندازی هیچ نوع سیستم ورودی و خروجی فیزیکی را بکار نمی اندازند و نیز منتظر رسیدن پیام از یک وسیله ورودی و خروجی نمی مانند.

نرم افزار راه اندازی در سیستم unix

برای هر ردیف از وسایل ورودی و خروجی موجود در سیستم minix یک نرم افزار راه اندازی مجزا وجود دارد ، این وسایل با حالت ویژه ، فهرستها، نقشه حافظه و غیره پردازش های کاملی هستند . نرم افزارهای راه اندازی در صورت لزوم با استفاده از مکانیسم استاندارد عبور پیام که بوسیله تمامی پردازشها ی مینیکسی صورت میگیردبایکدیگر و همچنین با سیستم فایل در ارتباط هستند .بعه علاوه اینکه هر نرم افزاری که راه اندازی ه

عنوامن یک فایل مبدا تک نوشته شده است نظیر ساعت C یا فلاپی C .تنها تفاوت موجود بین نرم افزارها ی راه اندازی بین نرم افزارهای راه اندازی و دیگر پردازشها در این است که نرم افزارهای راه اندازی در هسته اصلی خود به یکدیگر متصل اند و به دین ترتیب همه آنها دارای فضای آدرس دهی مشترکی می‌باشند.

این طرح تا حد زیادی ماژولار است و نسبتاً مؤثر. همجنین این قسمت یکی از چند مکانی است که سیستم مینیکس را از سیستم یونیکس از جت راه و روش اصلی متمایز می کند. در سیستم مینیکس ، پردازش با فرستادن پیامی به پردازش سیستم فایل ، یک فایل را مطالعه می کند سیستم فایل ممکن است متقابلاًپیامی را به دسک گردان بفرستد و ازآن بخواهد که بلوک مورد احتیاج را بخواند . با ایجاد این تقابل از طریق مکانیسم پیام ، می توانیم قسمتهای مختلف این سیستم را با روش استاندارد مجبور نمود تا با قسمتهای دیگر رابطه برقرار کند . با این وجود با قرار دادن تمامی نرم افزارهای راه اندازی در فضای آدرس دهی هسته اصلی ، آنها می توانند در صورت احتیاج به سادگی به جدول پردازش و دیگر ساختارها ی داده های کلیدی دسترسی پیدا کنند.

هنگامی که فراخوانی سیستم انجام میگیرد ، سیستم عامل بطور معجزه آسایی از بخش فضای کاربرد به بخش فضای هسته اصلی تغییر می کند. این ساختار مانند سیستم یونیکس مانع طرح کالتیکس می باشد، طرحی که در آن تغییر تنها فراخوانی یک مرحله معمولی بوده تا یک تله ، کد به همراه ذخیره سازی وضعیت قسمت کاربرد می باشد.

نرم افزارهای راه اندازی در سیستم یونیکس مراحل کرنالی ساده ای هستند که بوسیله بخش فضای هسته اصلی ، پردازش فرا خوانده میشوند. هنگامی که راه اندازی احتیاج به

انتظار برای وقفه دارد، مرحله کرنال را فرا می خواند ، این مرحله راه انـدازی را بخـواب مـی برد تا یک مدیر وقفه آن را از خواب بیدار کند. توجه داشته باشید که این عمل فرآیند خـود کاربر است که در اینجا بهخواب برده شده زیرا بخشـهای کـاربر و کرنـال د رواقـع بخشـهای متفاوت از این فرایند یکسان می باشند.

بحث و تبادل نظر در مورد شایستگی سیستمهای یک پارچه همچون مینیکس در مقابل سیستم های ساختاری –پردازشی مانند یو نیکس د رمیان طراحان سیستم عامل تمام ن شـدنی است .در سیستم مینیکس ساختار بهتر و تقابل واضح تری در بـین اجـزاء دیـره میشـود و ایـن مورد به راحتی به سیستم توزیع ده که در آنها پردازشهای گوناگون ، کامپیوتر هـای متفـوت را بکار می اندازدند گسترش یافته است . دیدگاه یونیکس کار آمد تر اسـت زیـرا فراخـوانی مراحل بسیار سریعتر از فرستادن پیام می باشد . سیستم مینیکس به پردازشهای زیادی تفکیک شده است زیرا به عقیده من با توجه به افزایش رو به رشد میکـرو کامپیوترهـای قـوی و قابـل دسترس ، ساختار نرم افزارهـای مشـخص تـر د رایجـاد سیسـتم هـای آهسـته تـر (کنـدتر) بـا ارزشت بوده اند . باید توجه داشت که بسیاری از طراحان سیستم های عامل از ایـن نظـر هـم عقیده نیستند.

چهار چوب سیستم مینیکس که در این کتاب شرح داده شده است شامل نرم افزارهای راه اندازی برای حافظه با دستیابی تصادفی (RAM) دیسک ، فلاپی دیسک . ساعت و پایانـه می باشد .(توزیع نرم افزار مینیکس شـامل راه انـدازی هـای اضـافی نظیـر چـاپگر و دیسـک سخت می باشد.)پیام های درهواستی فرستاده شده به این task ها شامل عناوین مختلفـی مـی

باشند که برای نگاه داشتن کد عامل (operation code) و پارامترهاب آن مورد استفاده قرار می گیرند

عناوین دستگاهها ی کارکتر ها اساساً شبیه به هم هستند اما می توان از یک برنامه برنامه دیگر دارای اختلاف کمی باشند. مثلاً پیامهای فرستاده شده به برنامه ساعت شامل زمان می شود و پیامهای فرستاده به برنامه پایانه ، کارکترها را برای استفاده از عمل چاپ بین خطی تعیین می کند.

۳-۳ نرم افزار ورودی /خروجی غیر وابسته به وسایل سخت افزاری در minix

پردازش سیستم فایل د رمینیکس شامل تمامی کدهای ورودی و خروجی غیر وابسته به وسایل سخت افزار ی می شود . سیستم ورودی وخروجی ارتباط تنگاتنگی با سیستم فایل دارد به طوری که انه در یک پردازش ترکیب شده اند .

سیستم فایل غیر از اینکه مدیریت ارتباط بین درایو ها ، حافظه باز ، تعیین کننده ASCII و موار مشابه بر عهده دارد، حمایت و مدیریت فهرستها و سیستم فایل نصب شده را نیز به عهده دارد.

مدل کلی و عمومی که قبلاً دراین بحث خلاصه شده . در این نیز بکار م رود . مراحل کتابخانه برای ساخت سیستم فرا خوان ها و برای تغییر از مبنای ۲ به ASCII و بلعکس قابل دسترسی است . در چهار جوب و در پیکره سیستم استاندارد مینیکس دی مونها ی هماهنگ کننده چاپ وجود ندارد اما چون آنه تنها پردازش های کابرد هستند ، در صورت نیاز استفاده کردن دی مونهای هماهنگ کننده چاپ آسان است .

٤-۳ مدیریت وقفه در سیستم minix (Dead Lock)

سیستم مینیکس با توجه به موضوع وقفه مسیری را طی می کن که سیستم یونیکس می پیماید و روی هم رفته مشکل را نادیده می گیرد . سیستم مینیکس دارای دستگاههای ورودی و خروجی اختصاصی نمی باشد، حتی اگر کسی قصد داشته باشد به منظور ساخت نرم افزارها به کامپیوتر شخص آی – بی - ام نوار مغناطیسی q شیاره استاندارد نصب کند هیچ مشکل مهمی بوجود نمی آید . بطور خلاصه می توان گفت تنها جایی که وقفه ها در ان رخ می دهد منابع مشترک مجازی همچون شیار جدول پردازش ، شیار جدول I-node و غیره می باشند . هیچ یک از الگاریتم های شناخته شده وقفه نمی توانند با منابعی نظیر آنهایی که بطور آشکارا درخواست نشده اند در ارتباط باشند .

در حقیقت مطالب بالا دقیقاً درست نیست . مکانهای چندی وجود دارند که از آنها برای جلوگیری از مشکلات بطور قابل ملاحظه ای مراقبت می شود . مهمترین مکان عمل متقابل ین سیستم فایل و مدیریت حافظه می باشد. مدیریت حافظه مانند کانتکس های دیگر برای خواندن فایل باینری در طی فرا خوانی سیستم EXEC پیامی را به سیستم فایل می فرستد.

هنگامی که مدیریت حافظه سعی در فرستادن پیام به سیستم فایل می باشد ، در صورت ایدال نبودن سیستم فایل مدیریت حافظه با وقفه (بلوک) مواجه خواهد شد . اگر سیستم فایل نیز سعی کند پیامی را به مدیریت حافظه بفرستد متوجه خواهد شد که قرار ملاقات یا برخورد این در از بین رفته است و بلوک خواهد شد و این مسئله منتهی به بن بست می شود .

به هنگام ساختن چنین فتیلی به طریقه بالا اجتناب می شود بطوری که سیستم فایل هرگز پیام های در خواستی را به مدیریت حافظه نمی فرستد و تنها جوابها را می فرستد .

یک استثنای کوچک وجود دارد و آن اینکه به هنگام شروع کار سیستم فایل اندازه و گنجایش خود را به مدیریت حافظه اعلام می کند و مدئیریت حافظه نیز ملزم به انتظار برای دریافت آن می باشد .

۵-۳ دیسک گردان RAM

دیسک گردان RAM در واقع جهار گرداننده نزدیک به هم می باشد که بصورت یک گرداننده دیده می شود. هر پیام ورودی آن به یک برنامه کوچک به صورت زیر تعیین می گردد:

0:/dev/ram 1:/dev/kmem

3:/dev/nvll

اولین فایل از بالا یک دیسک RAM واقعی می باشد . نه اندازه آن ، نه منشاءاصلی آن درون نرم افزار راه اندازی ساخته شده است . آنها توسط سیستم فایل و به وسیله راه اندازی ریشه سیستم فایل . و در زمانی که مینیکس راهاندازی شده است تعیین می شوند . این روش بدون توجه سیستم عامل ، افزایش و کاهش میزان آمار دیسک RAM را امکانپذیر می سازد . تمامی آنهیی که به این کار احتیاج دارند از دیسک سیستم فایل ریشه ای متفاوت استفاده می کنند.

دو نرمافزار دیگر به ترتیب برای خواندن ونوشتن حافظه فیزیکی و حافظه هسته ای اصلی بکار می روند . آخرین فایل ، فایل ویژه ای است که داده ها را قبول کرده و آنها را بیرون می ریزد . از این فایل معمولاً در دستور shell استفاده می گردد.

ساختار کلی دیسک گردان حافظه با دستیابی تصادفی (RAM) د راین جا حلقه اصلی پیام ها را قبول کرده و آنهارا به do-mem برای خواندن و نوشتن و یا به do-semp برای پیام ویژه ارسال می کند . پیام ویژه نیز به نوبه خود به دسک گردان می گوید که دیسک RAM در کجا قرار گرفته است . هنگامی که دستگاه مینیکس آماده شد ، هسته اصلی بکار می افتد و بعد از آن مدیریت حافظه بکار افتاده و سپس سیستم فایل شروع بکار می کند . یکی از اولین کارهایی که سیستم فایل انجام می دهد مشاهده مقدار گسترش سستم عامل در حافظه است . سپس ابیلاک سیستم فایل ریشه ای را می خواند تا ببیند بزرگی آن جه اندازه است . هرگاه سیستم فایل محل اتمام سیستم عامل و میزان احتیاج دیسک RAM به حافظه نداد، پیام را به دیسک گردان RAM فرستاده و حدود RAM را به آن می گوید.

٦-٣ نگاهی به دیسک گردان فلاپی در سیستم minix

دیسک گردان فلاپی دو پیام را قبول و پردازش می کند : خواندن بلاک و نوشتن یکبلوک . یکبلاک به اندازه block-size است که در h/typeh تشریح شده است و در سیستم توزیع استاندارد دارای 1024 بایت است البته می توان این مقدار را به آسانی تغییر داد.اندازه قطاع در دیسک 512 بایت است ، بنابراین دو قطاع متوالی همیشه با هم خوانده ونوشته می شوند . فایده بلاکهای بزرگتر کاهش در تعداد دسترسی های مورد لزوم دیسک است و در نتیجه پیشرفت در نمایش آن .

پیام های مورد قبول توسط دیسک گردان فلاپی از مزیت شکل صفحه استفاده می شود . این پیامها توسط سیستم فایل و داده های در خواستی انتقال یافته و یا آمده از حافظه بافر به سیستم فرستاده می شوند. سیستم فایل با ساختن فراخوان سیستم ، مراقب انتقال به فضای نشانی پردازش و بالعکس می باشد .

دیسک گردان فلاپی از روش بالابر یا SSF استفاده نمی کند . دیسک گردان فلاپی دقیقاً متوالی است یعنی حتی قبل از اینکه در خواست بعدی را قبول کند ، در خواست اولی راقبول کرده و آن را انجام می دهد . (FCFS) دلیل استفاده از این چنین استراتژی ساده ای مربوط به محیطی که minix قصد آن را داشته یعنی کامپیوتر شخصی.

در یک کامپیوتر شخصی ، بیشتر یک پردازشگر فعال است . گاهی اوقات یک تا دو پردازشگر پشت صحنه ای نیز وجود دارد .

تعدادی از ماحل فرعی که در دیسک گردان مورد استفاده قرار می گیرد به صورت ذیل می باشد:

stop –motor-1 موتورگردان را ه موقت کن

fac – out-2 فرمرمانی که به کنترل کننده صادر می کند

fac – results-3 نتایج دستور را از نتایج کنترل کننده مجزا می کند

recalibrate-4 بعد از جستجوی اشتباه ، درایو را بار دیگر می سنجد

reset-5 خاموش و سپس روشن کردن کنترل کننده بعد از یک اشتباه مهم

send – mess-6 مراقبت در فرستادن پیام

۳-۷ نگاهی به نرم افزار ساعت در minix

نرم افزار ساعت شامل فایل ساعت C می شود . این نرم افزار ۴ نوع پیام را به همراه پارامتریی قبول می کند.

۱- set - alarm

۲- get - time

۳- set - time

۴- clock - tick

حال به تعریف کارهای ۴ نوع پیام بالا در minix می پردازیم :

۱- set Alarm

این پیام به یک پردازش اجازه می دهد تا یک زمان سنج را با استفاده از اعدادی در ساعت ویژه بکار اندازد. وقتی که فراخوانی Alarm صورت می گیرد. در واقع پردازش فراخوان پیامی را به مدیریت حافظه ارسال می کند و سپس مدیریت حافظه پیام را به نرم

افزار ساعت می فرستد. همچنین این پیام توسط برنامه هایی که احتیاج به روشن کردن زمان شنج نگهبان دارند بکار می رود.

Get time-۲

Set time-۳

تنها هنگامی که اعداد ثانیه ها ، از اول ژوئن ۱۹۷۰ در ساعت دوازده ظهر شروع به سپری شدن کرده اند ، زمان واقعی فعلی را بر می گرداند. Set time زمان واقعی را تنظیم می کند . set time فقط توسط ابر کاربرها در خواست می شود، سیستم زمان واقعی راه اندازی را به صورت متغییر ذخیره می کند . سپس هنگامی که get time فراخوانده می شود، مقدار فعلی شمارس tick را به ثانیه ها تبدیل می کند و آن را به زمان راه اندازی شده اضافه می کند.

clock tick-٤

پیلمی است که به هنگام رویداد وقفه در ساعت به نرم افزار فرستاده می شود. پارامتر ندارد، هنگامی که نرم افزار این پیام را دریافت کرد زمان واقعی را بیان کرده ، برسی زمان علامت بعدی و یا فراخوان نگهبان را انجام داده ، tick فعلی را با پردازشی شارژ کرده و برسی می کند که آیا quantum بالا است یا نه .

۹-۳ نگاهی به نرم افزار پایانه در minix

نرم افزار پایانه بزرگترین فایل مبدآ در minix محسوب می شود به طوری که تقریباً دو برابر دیسک گردان فلاپی می باشد که خود دومین فایل بزرگ محسوب می گردد. اندازه نرم افزار پایانه تا حدی به وسیله مشاهده مدیریت صفحه نمایش و صفحه کلید توسط

نرم افزار بیان می شود، زیرا این دو صفحه در مقابل خود و وسایل پیچیده ای هستند. هنوز درک و آموختن این مطلب که ترمینال ورودی خروجی به اندازه برنامه نویس احتیاج به ۳۰ برابر کد دارد، برای اکثر مردم عجیب بنظر می آید .

نرم افزار پایانه پنج پیام را به شرح ذیل قبول می کند :

۱-بخوان کارکترها را از پایانه

۲-بنویس کارکترها را به پایانه

۳-پارامترهای پایانه را برای IOCTL تنظیم کن

۴-کارکتر قابل دسترسی

۵-درخواست قبلی در مورد خواندن را کنسل کن.

نرم افزار پایانه از یک ساختار داده ای اصلی (try struct) استفاده می کند که این داده پایانه به پایاته آرایه ساختارها محسوب می شود . حتی اگر کامپیوترهای شخصی IBM تنها یک صفحه نمایش و یک صفحه کلید داشته باشند ، نرم افزار راه اندازی طوری نوشته شده که براحتی می توان پایانه اضافی ، اضافه نمود. این مسئله بسیار مهم است زیرا ممکن است سیستم مینیکس را به سیستم های بزرگتر نقل مکان داد.

سیستم خروجی پایانه

خروجی پایانه در minix از ورودی پایانه آن آسانتر است . زیرا صفحه نمایش حافظه ای است که نقشه برداری شده است . هنگامی که یک پردازش می خواهد چیزی تایپ کند عموماً متغییر prinit را برای تشکیل یک خط فرا می خواند . متغییر print برای فرستادن پیام به سیستم فایل ،write را فرا می خواند. پیام شامل نشانگری است که به

کارکترهای چاپ شده اشاره می کند. سپس سیستم فایل پیامی را به نرم افزار پایانه می فرستد. نرم افزار پایانه رفته و آنها را آورد. و به روی RAM ویدئوکپی می کند.

۱-٤مدیریت حافظه در minix

مدیریت حافظه در سیستم مینیکس ساده است به طوری که در آن از روش صفحه بندی و تعویض همان paging و swaping استفاده نشده است. مدیریت حافظه لیست حفره های مرتب شده در دستور نشانی حافظه را مرتب می کند. هنگامی که حافظه به علت فراخوانی سیستم مورد احتیاج باشد ، لیست حفره با استفاده از اولین فراخوان مناسب برای تکه ای که به اندازه کافی برای پردازش جدید بزرگ است ، جستجو می شود.

زمانی که پردازشی در حافظه جای گرفت ، تا پایان پردازش در این مکان باقی می ماند ، این پردازش هرگز تغییر نمی کند و هرگز به مکان دیگری در حافظه نقل مکان نمی کند . مکان اختصاصی داده شده به این پردازش هرگز بزرگ یا کوچک نمی شود.

استراتژی ذکر شده احتیاج به توضیح دارد. عوامل دخالت کننده در این عبارتند از :

۱-اعتقاد به اینکه minix به جای اختصاص داشتن به سیستم های اشتراک زمانی به کامپیوتر های شخصی اختصاص دارد.

۲-تمایل به داشتن سیستم minix در کامپیوترهای شخصی IBM

۳-تلاش در جهت اجرای این سیستم در کامپیوترهای شخصی و کوچک در آینده .

جنبه غیر معمول از سیستم minix روش اجرای مدیریت حافظه است . این روش قسمتی از هسته اصلی نیست . در عوض این روش به وسیله اجرای پردازش مدیریت حافظه در فضای کاربر و ارتباط با هسته اصلی از طریق مکانیسم پیام استاندارد، مدیریت می شود. اکثر

کدهای مدیریت حافظه به جای اینکه فقط لیست پردازش ها و حفره ها را بکار برند ، برای مدیریت سیستم مینیکس ، مدیریت فراخوانیهای در گیر در مدیریت حافظه ، Fork,EXEC های اولیه اختصاص داده شده اند.

۲-٤ شمای حافظه

حافظه در سیستم minix در دو موقعیت معین میشود: اول هنگامی که یک پردازش منشعب میشود که در این حالت مقداری از حافظه که از نظر اندازه با والدین یکسان است به فرزندان اختصاص داده می شود . دوم وقتی که یک پردازش ، تصویر حافضه خود را از طریق فراخوانی سیستم EXEC تغییر میدهد که در این حالت تصویر قدیمی به عنوان یک حفره به فهرست خالی برگردانده شده و حافظه برای دریافت تصویر جدید اختصاص داده میشود. هرگاه پردازشی از طریق اخراج و یا نابود شدن به وسیله یک علامت ، پایان یابد حافظه آزاد میشود .

باید توجه داشت که حافظه قدیمی برای فرزند قبل از اینکه حافظه جدیدی به C اختصاص داده شود . بنابراین C می تواند حافظه فرزند را مورد استفاده قرار دهد. در این روش یکسری از جفت های Fork و EXEC بدون اینکه حفره ای بین آنها باشد در تمامی پردازش های همجوار حاصل می شوند. به طوری که کپی می شوند که قبل از آزادسازی حافظه قدیمی ، حافظه جدید را اختصاص داده است.

هنگامی که حافظه از طریق فراخوانی سیستم EXEC و یا Fork تعیین می شود، مقدار مشخصی از آن برای پردازش جدید در نظر گرفته می شود، در مورد قبلی ، مقدار گرفته شده با مقداری که پردازش والدین داشته اند یکسان است.

در مورد دیگری ،مدیریت حافظه مقدار حافظه ای را می گیرد که در header صفحه فایل اجرایی تعیین شده است، هرگاه که این تخصیص انجام پذیرد پردازش دیگر تحت هیچ شرایطی حافظه بیشتری اختصاص نمی دهد.

برای برنامه ای که از فضای جداگانه I و D استفاده می کند (توسط یک پیت در سرصفحه که بوسیلهٔ اتصال دهنده تنظیم می شود نشان داده می شود) تمامی فیلد موجود در سرصفحه تنها برای صفحه داده بکار می رود. برای برنامه ای که دارای ۴ کیلومتن ، ۲ کیلومتن داده و ۱ کیلو پشته و در نتیجه اندازه کلی ۶۴ کیلو می باشد باید ۶۸ کیلو حافظه اختصاص داده شود. (۴ کیلو فضای آموزشی + ۶۴ کیلو فضای داده) سرحد بخش داده تنها می تواند به وسیله فراخوانی سیستم BRK جابجا می شود. تمامی BRK ها بررسی می شوند تا مشخص شود که آیا بخش داده ای جدید برخورد تصادفی با نشانگر پشته فعلی داشته است یا خیر و اگر نداشته ، باید به تغییر در تعدادی از جدول های داخلی توجه داشت.

۱-۵- نگاهی به سیستم فایل در ninix

در minix دو نوع فایل موجود است. یکی فایل ordinary و دیگری فایل های special فایلهای ordinary مثل فایل read ، write و فایل special مثل file می باشد.

به ازاء هر file در minix یک I-node داریم. در هر سیستمی برای هر file یک file description داریم.

نتیجه گیری

نوع سیستم عامل از نوع client server می باشد.

مدیریت فرآیندها در ۴ لایه کار می کند:

۱- process management

۲- I/O tasks

۳- server process

۴- user process

مدیریت فرآیندها از نوع Round-Robin می باشد.

در مدیریت حافظه از paging و swaping استفاده نمی شود و از روش huk ها از طریق first file عمل می کند.

سیستم عامل UNIX

مقدمه

ویرایش نخستین UNIX توسط keu thempson نوشته شد و بعدها به Dennis Ritchie پیوست. در Bell Babs در اواخر ۱۹۶۰ ، UNIX یک سیستم تک کاربر بر روی کامپیوترهای PDP-7 بود که به زبان اسمبلی نوشته شده بود بعد از یک بازنویسی مجدد به زبان C نوشته شد که قابل اتصال به خانواده کامپیوترهای PDP-11 بود و بعد از آنUNIX برای استفاده کننده های خارج از کمپانی AT&T قابل دسترس شد. دستمزدهای مساعد licening اجازهٔ حق کپی کم برای مؤسسات آموزشی زمینه اقتباس (استفاده) UNIX در بسیاری از دانشگاهها بود. هم اکنون UNIX به صورت تجاری از AT&T قابل دسترس است. به همراه انواع مختلف سیستم که توسط فروشنده های دیگر عرضه می شود به کار برده می شود. ویرایش های تخصصی متعددی از یونیکس موجود است مانند PWB (Programmer work bench) و WWB / (witten work bench) و ...

بعضی از خصوصایت UNIX که در اینجا مطرح می شود عبارتند از:

1- Portability قابلیت حمل و نقل

2- Multy user operation عملکرد چند کاربرده

3- Device independence مستقل بودن از وسایل

4- Tools & - bulting utilities نرم افزارهای ابزارسازی و ابزار

همانطور که اشاره شد UNIX قابل انتقال و قابل دسترس از سخت افزارهای مختلف است. چیزی که عجیب است این است که قابلیت انتقال از اهداف طراحی UNIX نبوده است.

بلکه این امر ناشی از کد کردن سیستم با یک زبان سطح بالا می شود. در حین تشخیص اهمیت نقل و انتقال طراحان UNIX کدهای مربوط به سخت افزار را به واحدهای کنترلی به منظور انتقال آسانتر محدود کرده اند. UNIX استفاده کننده های متعددی را با اتصال شایسته ای و مناسبی با سخت افزار مدیریت حافظه و اختصاص دادن وسایل جانبی ارتباطی حمایت می کند.

یک سیستم UNIX را می توان به صورت یک هرم نمایش داد. در پایین ترین قسمت این هرم سخت افزار شامل پردازشگر CPU و دیسکهای (disks) و ترمینالها، و غیره قرار گرفته اند.

سیستم عامل UNIX که مستقیماً روی سخت افزار عمل می کند در سطح بعدی قرار گرفته است و وظیفه اش کنترل کردن سخت افزار و برقرار کردن ارتباط سیستم برای برنامه ها است. این system call امکان ایجاد و مدیریت پراسه ها و فایلها و غیره را برای برنامه های کاربران ایجاد می کند.

برنامه ها برای ایجاد system call ها مقادیر مربوطه را در رجیسترها قرار می دهند و به این ترتیب از قسمت usermode به kerne mode منتقل می کنند و از آ"جایی که در زبان C دستور یا راهی برای اجرای این دستورالعملها نیست یک کتابخانه library تهیه شده است که در آن به ازای هر system call یک procedure که به زبان اسمبلی نوشته شده است وجود دارد که این procedure ها اگر چه به زبان C نوشته نشده اند ولی می توانند فراخوانده شوند بنابراین برای اجرای دستور read برنامه C می تواند procedure خواندن را از library فرا بخواند. بنابراین در حقیقت اینها system call نیستند بلکه libraty call

هستند و سیستم عامل مشخص می کند .که کدام procedure باید فراخوانده شود و پارامترهای آن چیست؟ کارش چیست؟ و چه نتایجی را باید برگرداند؟

به غیر از سیستم عامل و کتابخانه سیستم call ها همه نسخه های UNIX تعداد زیادی برنامه های standard نیز دارند که برخی از آنها بین تمام نسخه ها یکسان هستند و برخی دیگر از نسخه ای به نسخه دیگر فرق می کنند. این برنامه ها شامل پوسته یا shell و کامپایلرها و editor ها و برنامه های پردازشگر متن processing programs text و برنامه های ایجاد فایل هستند و کاربر با این برنامه سر و کار دارد.

بنابراین ما می توانیم از سه interface مختلف در UNIX صحبت کنیم:

۱- ارتباط حقیقی بین سیستم callها

۲- ارتباط کتابخانه ای

۳- ارتباط ایجاد شده از طریق برنامه های استاندارد کمکی

بنابراین قسمتی که کاربر فکر می کند که سیستم عامل است در حقیقت برنامه های استاندارد کمکی هستند و هیچ ارتباطی با سیستم عامل ندارند و به راحتی قابل جایگزینی با برنامه های دیگر هستند.

به عنوان مثال بسیاری از نسخه های UNIX به جای ارتباط از طریق صفحه کلید ، ارتباط با ماوس را جایگزین کرده اند و این کار بدون ایجاد هیچگونه تغییری در سیستم عامل انجام گرفته است.

این قابلیت انعطاف پذیری در سطح کاربر یکی از دلایل رواج UNIX است که آن را قابل استفاده به شکل دلخواه بدون تغییر در اساس کار می کند.

ساختار داخلی UNIX :

ساختار داخلی UNIX به صورت لایه ای و خادم و مخدوم می باشد.

فایل ها سیستم مراتبی و به صورت درختی می باشند. نام یک فایل حداقل ۱۴ حرف و حداکثر ۲۵۶ حرف است. Special فایلها در ریشه dev قرار دارند. Kernel روی سخت افزار سوار است و shell روی kernel در UNIX بیش از 80 system call روی PC ها و main ها یکجور هستند. و در روی سیستمهای مختلف یکسان است.

ورود به UNIX :

برای استفاده از UNIX اول باید وارد آن بشوید، ورود به UNIX بوسیله تایپ کردن اسم و رمز ورود ،Pasword انجام می شود. برنامه ای به نام login این اسم و رمز ورود را می خواند و چک می کند تا از امنیت داده ها مطمئن باشد و هر کاربری از فایلهای مربوط به خود استفاده کند. اگر چه این روش برای سیستمهای timesharing روش استاندارد و معمولی است ولی سیستم عاملهایی مانند DOS اصلا به این موضوع محافظت اهمیت نمی دهند و هر کاربری که با سیستم عامل DOS کار می کند می تواند هر فایل و داده ای را که در ماشین ذخیره شده را مورد ارزیابی قرار دهد و بوجود آورنده آن فایل اهمیت ندارد و همه می توانند از آن استفاده کنند.

بیشتر سیستمهای اشتراک زمانی timesharing تمام اسامی کاربران و رمزهای آنان را در یک فایل مخفی secre file ذخیره می کنند ولی سیستم عامل UNIX از روش بهتری استفاده می کند.

فایل رمزها شامل یک خط برای هر کاربر است که در آن اسم ورود کاربر user login name و شماره کاربر numerical user id و رمز کاربر password و دایرکتوری مربوطه home directory و سایر اطلاعات نگهداری می شود. وقتی کاربری قصد ورود به سیستم را داشته باشد برنامه ای به نام login از ترمینال مربوطه رمز داده شده توسط کاربر را می خواند و با ستون رمزهای فایل مقایسه می کند اگر رمز نوشته شده با رمز درون فایل مطابقت کند اجازه ورود صادر می شود.

فایلها و دایرکتریهای در UNIX :

یک فایل در سیستم UNIX می تواند شامل صفر یا بیشتر بایت از اطلاعات اختیاری باشد در این سیستم هیچ تفاوتی بین فایلهای اسکی و باینری و یا هر نوع دیگری وجود ندارد و معنای بیتها در هر فایلی فقط به صاحب یا بوجود آورندهٔ فایل بستگی دارد و سیستم اهمیتی به معنا و محتوای بیتها یک فایل نمی دهد.

اسم فایل می تواند معمولا تا ۱۴ بیت کاراکتر اختیاری باشد که در نسخهٔ Berkely تا ۲۵۶ کاراکتر نیز مجاز است.

بسیاری از برنامه ها انتظار دارند که اسم فایل شامل یک اسم اصلی و یک پسوند باشد که این دو یک کارکتر محسوب می شوند. مثلاً prog.c بیانگر یک برنامه C است و prog.p بیانگر برنامه ای به زبان پاسکال است هر چند که این شیوه اسم گذاری برای فایلها از طرف سیستم عامل اجباری و تعیین شده نیست و کاربر برای انتخاب نام فایلها آزاد است.

برای حافظت از فایلها ۹ بیت به هر فایلی اختصاص داده می شود. این ۹ بیت به نام high bits شناخته می شوند.

هر ۳ بیت وضعیتهای خواندن ،نوشتن و اجرای یک فایل را روشن می کند مثلا کد ۶۴۰ اکتال بیانگر این است که صاحب فایل می تواند فایل را بخواند و در آن بنویسد. اعضای گروه صاحب فایل فقط اجازه خواندن آن را دارند و ناشناسان هیچ حقی برای دسترسی به آن ندارند. کد ۱۰۰ (اکتال) به صاحب فایل اجازهٔ اجرای آن را می دهد ولی بقیه هیچ حقی در دسترسی به فایل ندارند و حتی صاحب فایل حق خواندن فایل را ندارد اگر چه صاحب فایل این کد را می تواند تغییر دهد روی صفحه این دو مثال به شکل

... x...., 1w-r...

نمایش داده می شوند، کدی که به همهٔ افراد اجازه هر نوع دسترسی به فایل را می دهد که ۷۷۷ اکتال است که شکل 1wx1wx1wx نمایش داده می شود.

تجمع فایلها تشکیل دایرکتوری ها را می دهد. دایرکتوری ها نیز مانند فایلها ذخیره می شوند و می توان با آنها مانند فایلها در سطحی وسیعتر برخورد نمود. دایرکتوری ها نیز مانند فایلها دارای ۹ بیت برای حفاظت می باشند بااین تفاوت که جای بیت سوم در هر سه بیت که اجازهٔ اجرای فایل را می داد در دایرکتوری اجازهٔ جستجو در آن را صادر می کند.

دایرکتوری ها می توانند طبق سیستم سلسله مراتبی شامل زیردایرکتور نهایی باشند دایرکتوری ریشه با کارکتر / قابل دسترسی است و برای جداکردن اسامی دایرکتوری ها به کار می رود. بنابراین ust/ast/x نمایانگر فایل x است که در دایرکتوری ast قرار دارد که خود دایرکتوری ast در دایرکتوری UST قرار دارد.

UNIX Implementation (اجرای UNIX):

در این بخش بعضی از جنبه های عملکرد اجرای UNIX را تشریح می نمائیم:

برای تشریح منظور خود ابتدا با مدیریت پروسه آغاز می کنیم سپس با مدیریت حفاظه در UNIX ادامه می دهیم و این قسمت را با توضیح در مورد سیستم مدیریت فایل به پایان می رسانیم.

Kernel ، یونیکس ذاتا یک مونیتور یکپارچه با اندازه متوسط است ،و سیستم کال ها به صورت مجموعه ای از Coroutine ها (هم روال ها) کامل شده اند. یک coroutine کرنل با برچسب درخواستی (user) سنکرون است.

مفهوم resident : به قسمتهایی از سیستم عامل ها به کرات نیاز داریم ما آنها را مقید می کنیم در حافظه تا موقعی که سیستم خاموش شود. در بعضی از سیستم ها که خیلی سرعت بالایی دارند ما این قسمت ها را در rom قرار می دهیم تا نیازی به load و unload کردن نداشته باشیم. بعضی از اطلاعات نگهداری شده در قسمت مقیم دائمی ، بلوک کنترل فرآیند، نام فرآیند، پوینترهایی به شکل حافظه ای آن و اطلاعات جدول بندی شده را در بر دارد.

قسمت قابل تبادل بلوک کنترل فرآیند شامل موارد زیر است: به همراه سایر چیزها، رجیسترهای ذخیره شده CPU ، لیست فایلهایی که توسط فرآیند باز شده اند و اطلاعات ارتباطات اشاره گرهای مقیم مربوط به بلوک های کنترل فرآیند بطور دائم در جدول فرآیند نگهداری می شوند.

به منظور تقسیم بندی به هر فرآیند یک اولویت واگذار شده است. سطح اولویت یک فرآیند سیستم معمولا به اهمیت نسبی رویداری که فرآیند با آن سر و کار دارد وابسته است. برای مثال رویدادهای دیسک دارای اولویت بالا و رویدادهای پایانه دارای اولویت پایینتری هستند مقام فرآیندهای سیستم دارای اولویت بیشتری نسبت به فرآیندهای user دارد. اولویت فرآیندهای کاربر به صورت ابتدایی تغییر می کند و به مقدار زمانی که فرآیند از CPU استفاده کرده است. بستگی دارد.

پردازشها در UNIX :

تنها موجودیتهای فعال در UNIX پراسه ها هستند. در سیستم عامل یونیکس هر پراسه ای یک برنامه را راه اندازی می کند و راه کنترل مخصوصی دارد به عبارت دیگر یک شمارنده برنامه (PC) برای هر دستور العمل وجود دارد که دستورالعمل بعدی قابل اجرا را نشان می دهد.

UNIX یک سیستم عامل چند برنامه ای Multi programming است بنابراین پراسه های مستقل می توانند به صورت همزمان اجرا شوند هر کاربری ممکن است برنامه های فعالی در آن واحد داشته باشد بنابراین در یک سیستم بزرگ در یک لحظه ممکن است صدها یا هزاران پراسه در حال اجرا باشند. در حقیقت در بیشتر سیستم های تک کاربره حتی وقتی که کاربر غایب است نیز کارهای زیادی در حال اجرا وجود دارد که آنها را با نام daemons می شناسیم. این پراسه ها همراه با boot شدن سیستم بطور اتوماتیک شروع به فعالیت می کنند. یکی از این برنامه ها cron daemon است این برنامه یکبار در هر دقیقه فعال می شود تا ببیند کاری برای انجام وجود دارد یا نه؟ اگر کاری باشد انجام می دهد و اگر نباشد دوباره

به حالت خواب می رود تا در دقیقه بعد دوباره فعال شود. این برنام در UNIX کاربرد زیادی دارد زیرا برنامه ریزی آینده برنامه ها بوسیله این برنامه ممکن می شود برای مثال تصور کنید یک کاربر ساعت ۳ وقت دندانپزشک داشته باشد. این کاربر می تواند یک موجودیت تازه در بانک اطلاعاتی اش برنامه بوجود آورد که به برنامه می گوید که سر ساعت ۲:۳۰ بوق برند وقتی که روز و ساعت برنامه ریزی شده فرا برسد برنامه می بیند که کاری برای انجام دادن هست و برنامه تولید صدا را بکار می اندازد.

این برنامه (cron daemon) برای راه اندازی برنامه های هر روزه نیز بکار می رود مانند ایجاد کردن دیسک پشتوانه (Backup) هر روز در ساعت ۴ بعدازظهر یا کارهایی نظیر این.

Daemon های دیگر هماهنگ کننده های رسیده یا ارسالی هستند و می توانند صف پرینترها را مرتب کنند یا مقدار حافظه را چک کنند و ...

Daemon ها در سیستم عامل UNIX می توانند همزمان اجرا شوند زیرا برنامه های مستقل از هم هستند و به پراسه های دیگر ربطی ندارند.

این حقیقت که حافظه اختصاصی و متغیرها و رجیسترها و همه چیز دیگر برای پراسه های فرزند و پدر مستقل و منحصر به فرد هستند باعث ایجاد مشکلاتی می شود یکی از این مشکلات این است که پراسه ها از کجا باید بدانند که باید code مربوط به پراسه فرزند را اجرا کنند یا کد مربوط به پراسه پدر را ؟ رمز این کار این است که مقدار برگشتی return value سیستم Fork call برای پراسه فرزند و برای پراسه پدر غیر صفر خواهد بود این مقدار برگشتی بنام pid که مخفف process identification است و به معنای شناسه پراسه

شناخته می شود هر دو پراسه فرزند و پدر با کنترل مقدار برگشتی pid عمل خود را انجام می دهند.

پراسه ها با pid ها یشان شناخته می شوند. همانطور که دیدیم وقتی پراسه ای ایجاد می شود Pid پراسه فرزند به پراسه پدر داده می شود اگر پراسه فرزند بخواهد pid مخصوص به خود را داشته باشد system call ای وجود دارد بنام GET PID که این کار را انجام می دهد. Pids راههای استفاده زیادی دارند به عنوان مثال اگر پراسه فرزندی terminate شود pid پراسه فرزندی که اخیراً تمام شده به پراسه پدر داده می شود این امر به دلیل اینکه پراسه پدر ممکن است دارای پراسه فرزندهای متعددی باشد مهم است و از آنجایی که هر پراسهٔ فرزند ممکن است پراسه های فرزند مخصوص به خود را داشته باشد یک درخت پراسه های فرزند برای هر پراسه پدر به وجود می آید .

امکان ایجاد درخت پراسه ها کلید اصلی کارکردن سیستم عامل به صورت تقسیم زمانی است وقتی سیستم شروع به کار می کند (boot می شود) یک پراسه به نام init بوسیله لایه kernel اجرا می شود این پراسه فایلی بنام /etc/hys را می خواند این فایل به سیستم عامل می گوید که چند ترمینال وجود دارد و مشخصات برای توصیف هر ترمینال را تهیه می کند. برنامه init سپس برنامهٔ Fork را برای یک پراسه فرزند برای هر ترمینال اجرا می کند و به حالت خواب یا استراحت می رود تا پراسه فرزند خاتمه یابد.

هر پراسه فرزند برنامه ورود را با تایپ کردن :login روی صفحه ترمینال آغاز می کند و سعی می کند نام کاربر را بخواند وقتی شخصی پشت ترمینال قرار می گیرد و اسم را وارد می کند برنامه login رمز ورود را می پرسد و اگر رمز درست بود برنامه login در حالت

غیر فعال موقتی تا دریافت دستور یا command بعدی باقی می ماند ولی اگر رمز اشتباه بود در فایل رمزها وجود نداشت برنامه login رمز دیگری را خواهد داشت.

این مکانیزم در شکلی نشان داده شده است این سیستم دارای سه ترمینال است برنامه login که در ترمینال اجرا می شود هنوز منتظر اسم است برنامه login در ترمینال ۱ با موفقیت اجرا شده و در حال حاضر در حال اجرا کردن shellاست که منتظر یک دستور العمل است. یک ورود موفق در ترمینال ۲ نیز انجام شده است و فقط در اینجا کاربر برنامهٔ CP را آغاز کرده است که بصورت پراسه فرزند shellدر حال اجرا می باشد و shell در حالت غیر فعال منتظر به پایان رسیدن برنامه فرزند می باشد که در این زمان shellیک پیغام آمادگی (prompt) را برای خواندن دستور بعدی keyboard اعلام خواهد کرد. اگر کاربر در ترمینال ۲ به جای CP برنامه CC که ccempiler است را اجرا می کرد کامپایلر C نیز برنامه های دیگری را فرا می خواند و درخت بزرگتر می‌شد.