**سیستم عامل**

**مقدمه**

قابل توجه دوستان عزیز این جزوه درسی صرفا برای فهم هر چه بهتر درس سیستم عامل، طراحی گردیده است.

لازم به ذکر است که مطالب جمع آوری شده از تمام مباحث گفته شده در کلاس استاد سپهرزاده و باقی آن جمع آوری شده از سایت های تخصصی و کتاب اصلی تننباوم می باشد.

به امید موفقیت تمام دانشجویان عزیز

علیرضا سلطانی

**فصل اول**

**تعریف سیستم عامل**

سیستم عامل مهمترین نرم افزار سیستمی که واسط بین کاربر و سخت افزار است، سیستم عامل باعث می شود که کاربر به راحتی برای رفع نیاز های خودش از سخت افزار به واسطه گری سیستم عامل، استفاده کند.

**تعریف سیستم عامل از نظر کتاب Silver shot**

سیستم عامل مهمترین نرم افزار سیستمی است که کامپوتر را راه اندازی کرده و دائما در حال اجراست

اما این تعریف کامل و صحیح نیست چرا که کامپیوتر بعد از روشن شدن از نظر کاربر دائما در حال اجراست اما در حقیقت همیشه آماده برای انجام کار است.

**هدف استفاده از سیستم عامل**

1.مدیریت بهتر منابع از نظر تخصیص منابعی چون (فایل ها،پرونده ها یا مثلا منابع سخت افزاری CPU ,Hard ,GPU).

2.ساده کردن کار بار سخت افزار برای کاربر .

**انواع منابع در تعریف سیستم عامل**

از نظر یک سیستم عامل دو نوع منبع وجود دارد:

1.منابع فیزیکی که قابل مشاهده هستند ماننده هارد یا سی پی یو و دیگر سخت افزار رایانه

2.منابع منطقی که قابل مشاهده نیستند مانند فال ها یا پرونده ها

**پردازنده (Processor)**

پردازنده قلب تپنده یک دستگاه کامپیوتری است که وظایف متعددی دارد، مهمترین وظیفه آن اجرای دستورات است

هر پردازنده از سه بخش مهم و اصلی تشکیل شده است:

**ALU بخش محاسبه و منطق:**

این بخش وظیفه انجام عملیات ریاضیاتی را بر عهده دارد

**CU بخش کنترل:**

وظیفه اصلی این بخش، مدیریت و هماهنگی سازی بین بخش ALU وRegister است

**Register(ثبات)**

این بخش وظیفه آن را دارد که تمام برنامه هایی که در حال حاضر در بخش CPU اجرا شده است را به وظیفه معلق در آورده است که در دفعات بعدی مراجعات CPU را به برنامه مورد نظر را کمتر کند که در نتیجه سرعت لود شدن برنامه کاربر بیشتر از اولین زمان اجرا بوده است.

**انواع حافظه Register:**

در اینجا 5 مدل از حافظه ثبات آورده شده است:

**ثبات دادهData Register:**

از 4 بخش اصلی AX,BX,CX,DX تشکیل شده اند و وظیفه کلی آنها کار بر روی دیتا های ورودی است.

**ثبات شاخصIndex Register:**

این نوع ثبات وظیفه آدرس دهی به داده ها را بر عهده دارد که دارای مبدا و مقصد است:

SI: که مخفف Source Index است وظیفه آدرس دهی داده در مبدا را به عهده دارد.

DI: که مخفف Destination Index است که وظیفه آدرسی در مقصد بر روی داده ها را بر عهده دارد.

**ثبات پشتهStack Register:** برای درک این بخش بایستی با مفهوم پشته یا همون Stack آشنا باشیم،کلمه Stack یا پشته را بار ها در همه جا شنیدیم، در شبکه، برنامه نویسی یا در همین مفاهیم پایه سیستم عامل، **به معنای واقعی پشته یه سری عملیات رو روی داده ها در یک سمت انجام می دهد.**که بطور کلی دو دسته دارد:

**Base Pointer (BP):** انجام عملیات از مبدا تا مقصد.

**(SP) Stack Pointer:** انجام عملیات از مقصد تا مبدا.

**رجیستر های کنترلی:**

**دو دسته اند:**

Instruction register (IR)

این بخش وظیفش آن است که دستور عمل بعدی که قرار است در CPU انجام شود را در خود ذخیره می کند.

# Instruction Pointer / Program Counter (IP/PC)

اما این بخش آدرس دستورعمل بعدی که قرار است توسط CPU اجرا شود را در حافظه ذخیره میکند.

تفاوت این بخش در ذخیره سازی آدرس دستورعمل و اجرای آنها می باشد!

**ثبات پرچمی یا Flag Register :**

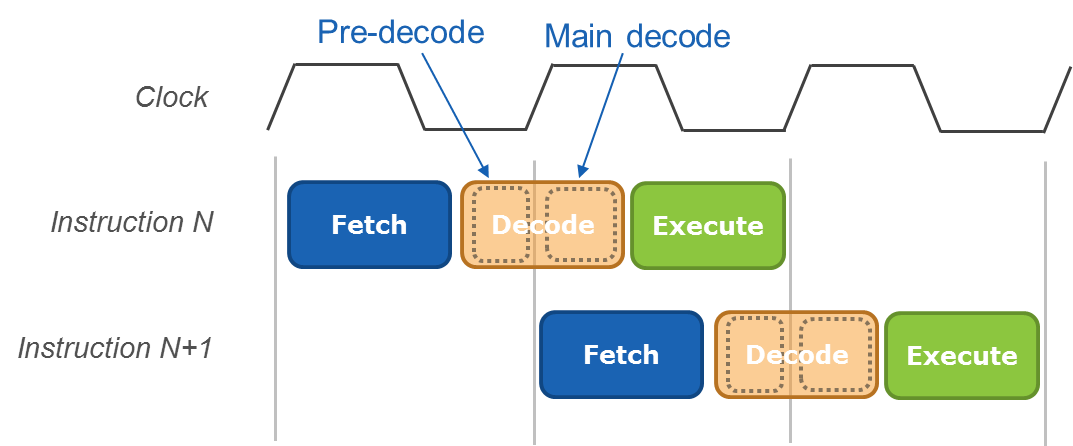
این بخش آخرین وضعیت اجرای هر دستور عمل را نشان می دهد. که می توان به پرچم وقفه Flag Interrupt اشاره نمود.

لازم به ذکر است که هر CPU دائما در حال انجام سه وظیفه مهم، واکشی(Fetch)، رمزگشایی(Decoding)، اجرا(Execute)است که اگر وقفه ای موجب شود CPU برای لحظه توقف کند ادامه این وظایف را به بخش PSW می سپارد.

**Pipeline چیست؟**

به معنای واقعی، بدون هیچ در گیری ذهنی، یعنی اجرای یک فرایند در ادامه فرایند دیگر. بر فرض مثال یک فرایند در حال اجراست و الان در مرحله رمزگشایی قرار دارد،در همین مرحله فراینده دیگری هم آغاز می شود و این مسئله تا انتها ادامه خواهد داشت

برای درک بهتر به تصویر زیر توجه کنید



نکته قابل توجهی که وجود دارد این است که انجام فرایند های ورودی-خروجی به تنهایی برای CPU بسیار وقت گیر است به همین دلیل این وظیفه بر عهده Controller ها خواهد بود.

**تعریف Controller:**

کنترلر یک قطعه سخت افزاریست که که دستور عمل ها را هدایت می کند.

**بخش های Controller**

بخش های اصلی یک Controller در زیر آمده است

**Command**

این بخش همان طور که از اسمش پیداست وظیفه دریافت دستور را از CPU دارد

**Status**

این بخش وظیفه بررسی وضعیت انجام دستور را بر عهده دارد

**Buffer**

**تعریف Driver**

Driver نرافزار یا افزونه ای است که به سیستم عامل کمک می کند تا از سخت افزار به صورت بهینه استفاده کند. یا دستوری صادر کند یا وضعیتی را بررسی.

**میانگین زمان دسترسی به حافظه**

**فرمول:**

**System** **Bus گذرگاه های سیستمی**

به طور کلی سیستم باس به اتصال اجزای سریع ومهم ماننده CPU، بانک حافظه و سیستم ورودی-خروجی به یکدیگر گفته می شود.

**انواع سیستم باس**

**Bus Address**

**Bus Data**

**Base Controller**

**Interrupt یا وقفه**

رویدادیست خارج از CPU که به CPU کمک می کند تا از اتفاق های سخت افزاری و نرم افزاری مطلع شود.

برای مثال می توان به عمل چاپ اسناد توسط پرینتر اشاره نمود

CPU داده را برای چاپ به پرینتر ارسال کرده و بقیه وضعیت را به Controller می سپارد به محض اینکه داده با موفقیت توسط پرینتر چاپ شد، یک وقفه (Interrupt) به سمت CPU ارسال کرده تا عمل را موفقیت آمیزبیان کند تا CPU به دنبال بقیه عملیات برود.

**Interrupt vector جدول وفقه**

کلا جدولیست که در آن شماره وقفه به همراه ادرس وضعیت دستورالعمل در آن قرار می گیرد.

**انواع وفقه**

**سخت افزاری** : اشکالات فیزیکی مانند خرابی دیسک سخت یا اشکل در Jump Bios.

**نرم افزاری** : اشکالاتی که مربوط به برنامه و سیستم عامل آن است.

**دو حالت برای اجرای برنامه وجود دارد:**

**حالت هسته** که به آن monitor یا مد هسته گفته میشود

**حالت کاربر**

**در کنار آن تعریف دو حالت هسته و شل تعریف می شود:**

Core mode

Shell mode

**انواع منابع از نظر انحصاری بودن**

منابع انحصاری یا non-Preemptiveمنابعی هستند که نمی توان آنها را از یک پردازه قبل از اتمام کار گرفت

منابع غیر انحصاری Preemptiveمنابعی هستند که آنها را میتوان از یک پردازه قبل از اتمام آن گرفت**.**

**تعریف PIC Programmable Interrupt Controller**

یک قطعه سخت افزاری است که وظیفه آن اولویت بندی کردن وقفه ها میباشد.

برای مثال سی پی یو دستوراتی مانند پرینت را به پرینتر میفرستد و در قسمت بافر کنترلر پرینتر رفته و از آن به قسمت PIC میرود و در مرحله بعد برای پاسخ به سی پی یو به سمتCPU میرود.

**تاریخچه سیستم های عامل از ابتدا**

کلا سیستم های رایانه ای نسل های مختلفی داشتند که آنها را با هم یه بررسی می کنیم.

**اولین نسل لامپ های خلاء**

نسل بدون سیستم عاملی بودند که با استفاده از لامپ های خلاء کار می کردند، ورودی داده از طریق تخته های مداری به سیستم داده می شد، زبان برنامه نوسیشان نزدیک به سخت افزار بود ،و با استفاده از همان لامپ های خلاء خروجی نمایش داده یعنی خروجی به صورت لامپ های روشن یا خاموش نشان داده می شد.

**دومین نسل پیدایش ترازیستور ها**

در این نسل ترانزیستور ها اختراع شدند، سیستم های عامل نوشته شدند و زبان های برنامه نوسی مثل فورترن بوجود آمدند و در ضمن آن تخته های مداری بروز شدند و کارت پانچ ها جایگزین شدند و از طریق آنها داده های ورودی نوشته می شدند. مهمترین سیستمی که در این نسل بسیار استفاده می شد سیستم های دسته ای یا Batch OS بودند.

سیستم های دسته ای سیستم هایی هستند که یک دسته کارا به صورت همزمان وارد ، پردازش و با هم نمایش می دادند.

**مهمترین هدف سیستم عامل هدر نرفتن زمان و منابع است.**

**مهمترین منبع در سیستم عامل CPU است.**

تمام داده ها در این سیستم ها بر پایه JCL یعنی Job Control Language نوشته می شدند

در صورتی که سی پی یو نتواند یک JCL را پیدا کند سی پی یو به دنبال JCL های بعدی می رود.

در این سیستم کلا سیستم عامل تک منظوره بوده است.

چند برنامگی Multi Programmingیعنی وارد کردن چند کار یا چند برنامه در سیستم های دسته ای به منظور استفاده بهینه از CPU

**معایب و مشکلات نسل دوم**

هدر رفتن وقت کاربر

هدر منابع

نبود ارتباط افلاین کاربر با پی سی

تک منظوره بودن آن

**سومین نسل پیدایش مدار های مجتمع**

در نسل سوم مدار های مجتمع آمدند که به وسیله آنها سی پی یو ها پیشرفت کردند که سی پی یو هایی مانند مدل پنتیوم وارد بازار شدند اندازه کوچکتر و سرعت بیشتر از ویژگی های بارز آنها بود.در این نسل دیسک ها و دستگاه های ورودی-خروجی، اختراع شدند .

**تعریف Spooling**

انجام همزمان عملیات ورودی-خروجی با پردازنده را Spooling می گویند.

Online Spooling

Offline Spooling

نسل دوم از نوع offline spooling بودند که تمام داده های ورودی در کارت پانچ نوشته و پردازش می شدند

و در نهایت به وسیله همان کارت پانچ ها نمایش داده می شدند.

اما نسل سوم از نوع online spooling بودند که داده ها را بروی دسیک می نوشتند و پردازش می شدند و در نهایت به وسیله خود آنها نمایش داده می شد.

**تعریف Responsive Time و Turnaround Time**

به ورود داده تا زمان نمایش اولین خروجی Responsive Time گفته می شود.

به وروده داده تا زمان نمایش آخرین خروجی Turnaround Time گفته می شود.

نکته مهم این است که هر سی پی یو قادر به تقسیم زمان به واحد های کوچکی به نام کوانتوم است که در حد یک میلی ثانیه هستند و همچنین میتوانند از یک پردازه که درون حافظه است سویچ کنند.

سیستم عامل مسئول برقراری عدالت و حفاظت بین پردازه ها می باشد زیر در حافظه تعداد زیادی پردازه وجود دارد باید از انها ومحافظت کند تا روی هم اثر منفی نگذارند.

**تقسیم منابع از نظر میزان استفاده سی پی یو و دستگاه ورودی-خروجی**

I/o Limited خیلی از عملیات بر روی ورودی/خروجی ها انجام میشه

CPU Limited یعنی خیلی از عملیات روی پردازنده انجام میشه

**چهارمین نسل، نسل میکروکامپیوتر**

ظهور کامپیوتر های شخصی

سیستم عامل های قدرتمند

و تعامل بیشتر کاربر با سیستم عامل

**انواع سیستم عامل ها:**

Server

Personal

Distributed

Embedded

Virtual

**انواع سیستم عامل ها:**

**سیستم عامل های شبکه ای:**

سیستم هایی هستند که چندین کامپیوتر به هم متصل میشوند تا با هم بتوانند اشتراک منابع و ارتباط با یکدیگر داشته باشند.

سیستم های شبکه ای سیستم هایی هستند که چندین و چند دستگاه به یکدیگر متصل شده اند و کاربر متوجه آن است هنگامی که دارد با سیستم عامل های شبکه ای کار میکند چندین دستگاه پشت پرده وجود دارد.

**سیستم های توضیعی Distributed System :**

سیستم های توضیعی سیستم هایی هستند که که در پشت پرده آنها چندین و چند سیستم وجود دارد اما کاربرزمانی که با آنها کار میکند متوجه ان نیست که چندین و چند دستگاه حضور دارد و فکر میکند که دارد با یک سیستم واحد کار خود را انجام می دهد.

**سیستم های سریس دهنده – سرویس گیرنده:**

سیستم هایی هستند همان طور که می دانید به آن Domain گفته میشود. که یک دستگاهی به عنوان سرویس دهنده یا Server وجود دارد که به وسیله یک اتصال دهنده چندتاییHob یا Switch چندین سیستم به عنوان سرویس گیرنده Client به آن متصل شده تا بتوانند منابع مورد نیاز خود را از سرور درخواست نماییند و سرور نیز منابع مورد نیاز هر مشتری را برایش ارسال کند.

**سیستم های توکار یا Embedded :**

همانطور که از اسمش پیداست سیستم های توکاری هستند که برای یک بار نوشته می شنوند دیگر توسط خودر شرکت سازنده توسعه داده نمی شوند. مهم ترین مثال این سیستم ها وجود سیستم کروز کنترل در خودرو های امروزی است.

**سیستم های بلادرنگ، بی درنگ یا Real Time :**

سیستم هایی هستند که میبایست کار خودشان را در زمان تعیین شده انجام دهند در غیر این صورت اگر سیستمی نتوانست کار خود را در بازه زمانی مشخص شده انجام دهد دوحالت رخ می دهد.

**حالت نرم Soft :** که ممکن است یک مشکل نرم افزاری رخ دهد مثلا سیستم هنگ کند

**حالت سخت Hard :** معمولا موجب مشکلات سخت افزاری بزرگ میشوند مثلا اتش سوزی یکی از دستگاه ها.

قبل از تعریف سیستم های Main Frame می بایست با مفهوم انجام کار به صورت دسته ای آشنا باشیم

سیستم های دسته ای سیستم هایی هستند که یه سری کار را بدون دخالت کاربر انجام می دهند.

**سیستم های Main Frame**

سیستم هایی که توانایی انجام کار های دسته ای را دارند، به صورت مستقیم با ورودی/خروجی ها در تعامل می باشند، انجام پردازش های سنگین اما حجم کم و قادر به عمل اشتراک زمانی هستند.

**حسگر ها:**

حسگر ها مانند یک سیستم واقعی هستند اما با پردازش ضعیف، قابلیت های کمتر، حافظه های کوچک تر و کم حجم ترتا انرژی محدود و کمتر را مصرف کنند.

**سیستم های دم دستی Handled system:**

مهم ترین مثال آنها همان تلفن همراه تبلت ها و غیره هستند که یه سری از برنامه ها را هندل میکنند

**Kernel کرنل :**

کرنل به زبان فارسی یعنی هسته. کرنل ویظفه مدیریت منابع سخت افزاری و سیستمی را برعهده دارد.

**فراخوانی سیستمی System Call**

زمانی که یه نرم افزار بخواهد به منابع سیستمی و سخت افزاری دسترسی داشته باشد، این درخواست را از سیستم عامل میکند که به طور کلی به آن سیستم کال گفته میشود.

**ساختار سیستم عامل ها:**

یکپارچگی Monolithic :

می توان گفت هر سیستم عامل می تواند شامل چندین زیرمجموعه یا پروسیجر باشد و هر زیر مجموعه میتواند چند زیرمجموعه دیگر داشته باشد وهر زیر مجموعه می تواند جزِئی از کرنل باشد، اگر مشکلی در یک زیر مجموعه ایجاد شود کل سیستمم به هم میریزد.

ساختار لایه ای سیستم عامل

هر سیستم عامل 5 لایه دارد:

لایه مدیریت ارتباط بین پردازه ها

لایه مدریت حافظه

لایه مدریت ورودی خروجی ها

لایه مدیریت ارتباط و تعامل کاربر

لایه کاربردی نمایش

ماشین های مجازی یا Virtual Machine:

هر سیستم عامل می تواند دو یا چند سیستم عامل داشته باشد، نرم افزاری باید وجود داشته باشد تا این سیستم عامل های مجازی را مدیریت کند هر سیستم عامل مجازی خیال می کند که کل سخت افزار را بر عهده دارد.

یه جدولی به نام جدول ماشین مجازی وجود دارد که تعداد و نام هر ماشین مجازی در ان ثبت میشود

Exokernel

در این مرحله هر سیستم عامل مجازی سخت افزار مشخصی را بر عهده می گیرد به عبارت دیگر

در این قسمت مشخص می شود که هر سیستم عامل مجازی چه مقداری از سخت افزار را بر عهده داشته باشد

Kernel Micro میکروکرنل:

میکروکرنل وظیفه کم کردن وظایف کرنل را برعهده دارد و آن ها را در مد کاربر اجرا میکند

**فصل دوم**

**پردازه Process**پردازه واحد برنامه های درحال اجرا یا آماده به اجرا در سیستم عامل است.معمولا برای انجام هر کاری یک پردازه صورت میگیرد و برای اجرای آن برنامه، پردازه مخصوص به آن صورت میگیرد.برای مثال هنگام نوشتن یک فایل بر روی دیسک پردازه مخصوص به آن صورت میگیرد یا مثلا پردازه چاپ خروجی به وسیله پرینتر.

NOTE : هر پردازه از سه قسمت اصلی **Code**، **Data** ،**Stack** تشکیل شده است.

سیستم عامل یک جدول برای هر پردازه دارد:Process Table Or PT

به هر رکورد این جدول PCB یا Process Control Block گفته می شود.

**نمودار وضعیت فرایند:**

دیتا وارد یک فرایند یا پردازه شده سپس آماده اجرا میگردد وبعد از آن اجرا میشود و در نهایت پایان میابد

در این میان ممکن است یکی از مراحل تعلیق شود

چند حالت برای خروج از پردازه وجود دارد:

خروج عادی

خطای برنامه نویسی غیرمهلک

خطای مهلک

کشته شدن توسط پردازه های دیگر

**زمانبند سی پی یو**

در این قسمت مشخص می شود کدام پردازه برای انجام کار انتخاب شود یا به عبارتی کدام برنامه ها از قسمت آماده باش به سمت اجرا بروند.

**زمانبند حافظه**

مشخص می کند که کدام بخش در حالت تعلیق قرار گیرد

**زمانبند کار**

مشخص میکند که بسته ها از new به ready بروند

**Content Switching، تعویض زمینه، تعویض متن**

در علوم کامپیوتر و در بحث سیستم عامل زمانی از تعویض متن استفاده میکنند که بخواهند وضعیت یک پردازه را ذخیره کندد و از آن در پردازش های بعدی از ادامه آن استفاده کنند که به سی پی یو قابلیت اشتراک زمانی را میدهند و این یکی از ارکان های بحث چند برنامگی است.

خب مراحل آن به این صورت است:

ابتدا قطعه سخت افزاری بنام تایمر سیگنال وفقه را به PIC میفرستد و از انجا که PIC وظیفه اولویت دهی به وقفه ها را دارد، وقفه ای که اولویت بالاتری دارد را انتخاب می کند. PIC از CPU درخواست وقفه میکند و CPU دستورالعمل جاری را به اتمام میرساند، CPU رجیستر های مهم از قبل PC و PSW را در استک فرایند جاری ذخیره می کند، CPU به PIC میگه من کارمو انجام دادم و یه وقفه به نام Interrupt Ack میفرستند، PIC شماره وقفه را در DataBus ذخیره کرده و CPU شماره وفقه را از DataBus می خواند، CPU به خانه ایکسم بردار وقفه میرود و ISR مربوط به تایمر را برمیدارد و در نهایت CPU این عمل را در PC قرار می دهد و از حالت هسته به مد کاربر میرود

**مفهوم نخ یا Thread :**

به طور کلی نخ یا Thread به کوچک ترین قسمت یک پردازه گفته می شود که باعث میشود برنامه نویسی راحتری داشته باشیم و برنامان ماژولاریتی بالاتری داشته باشد.

**مزایای کلی نخ یا Thread :**

1. در حالت تک هسته با اجرای همروند نخ ها سرعت خواندن و نوشتن افزایش پیدا میکند تا اگر مثلا یک نخ مسدود شد پردازنده به دنبال نخ بعدی برود.
2. در حالت چند هسته هر نخ بر روی پردازنده خودش به صورت مجزا انجام میشود که باعث افزایش عملکرد و سرعت خواندن و نوشتن آن میشود.
3. برنامه نویسی راحت تر و خوناتر میشود.
4. برنامه همراه با ماژولاریتی بالاتی خواهد شد.

نخ در دوحالت تک هسته به صورت شبه موازی و در حالت چند هسته به صورت موازی صورت میگرد

انواع نخ ها از نظر سطح کابر و سطح هسته و ترکیب هر دو آنها :

نخ های سطح کاربر در سمت مد کاربر انجام میشوند و چون در سمت کاربر هستند CPU از وجود انها اطلاعی ندارد

نخ های سطح هسته در مد هسته انجام میشوند و خود هسته با انها به صورت مستقیم در ارتباط است

نخ های ترکیبی در واقع همانطور که از نامشان پیداست ترکیبی از مد کاربر و مد هسته هستند.

مزایای نخ های سطح کاربر

1. ایجاد، خذف، تعویض و مدیریت نخ ها در سطح کاربر به سرعت انجام میشود

معایب نخ های سطح کاربر

1. چون نخ ها در مد کاربر هستند مد هسته از انها اطلاعی ندارد و اگر یک نخ مسدود شود کل پردازه مسدود خواهد شد

مزایای نخ سطح هسته

1. نخ ها چون در سطح هسته هستند اگر یکی از نخ ها مسدود شود پردازه بدنبال نخ های بعدی میرود

معایب

1. ایجاد، تغییر و تعویض، حذف و مدیریت نخ ها در سطح هسته به کندی انجام میشود

سطح ترکیبی نخ ها چون ترکیبی از هر دو سطح هستند هم ایجاد و تعویض و حذف سریع تری دارند هم اینکه اگر یکی از نخ ها مسدود شود کل پردازه مسدود نخواهد شد

**چرا از الگوریتم های زمانبدی استفاده میکنیم؟**

زمانی از الگوریتم های زمانبدی استفاده میکنیم که بخواهیم بین پردازه ها عدالت و سیاست (حالا بستگی دارد چه سیاستی رو استفاده میکنیم) رعایت شود.

**الگوریتم های زمانبد CPU**

**الگوریتم FCFS (First Come First Service)**

الگوریتم اولین ورود اولین سرویس :

این الگوریتم به صورت انحصاری می باشد یعنی آنها را نمی توان از یک پردازه قبل از خاته کار گرفت.

به مثال زیر توجه کنید

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P5 | P4 | P3 | P2 | P1 | Process |
| 5 | 4 | 3 | 2 | 0 | Arrival Time |
| 3 | 4 | 1 | 2 | 10 | Execute Time |

**این گانت زمان سرویس دهی را مشخص می کند !**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P5 | P4 | P3 | P2 | P1 |
| 20 | 17 | 13 | 12 | 0-10 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| WAT | TAT | Complete Time | Execute Time | Arrival Time | Process |
| 0 | 10 | 10 | 10 | 0 | P1 |
| 8 | 10 | 12 | 2 | 2 | P2 |
| 9 | 10 | 13 | 1 | 3 | P3 |
| 9 | 13 | 17 | 4 | 4 | P4 |
| 12 | 15 | 20 | 3 | 5 | P5 |
| 7.6 | 11.6 | “ | “ | “ | AVG |

اطلاعات استخراج شده از جدول بالا :

1. Process هر پردازه ای که وارد میشود.
2. Arrival Time زمان رسیدن یا ورود هر پردازه.
3. Execute Time زمان اجرا و سرویس دهی به هر پردازه.
4. Complete Time زمانی که هر پردازه به صورت کامل اجرا میشود.
5. Total Arrival Time میانگین زمان ورود هر پردازه به سی پی یو.
6. Wait Time میانگین زمان صبر و انتظار هر پردازه.

جریان از این قرار است که هرسرویسی که داخل فرایندی می شود با زمان سرویس فرایندی بعدی جمع می شود. این در واقع نشان میدهد که هر سرویس جدید با سرویس قبلی جمع میگردد(زمان سرویس بعدی)

نکته بعدی این است که جدول بالا میتواند به صورت تصادفی و بدون نظم باشد و کسی که قرار است این مسئله را با الگوریتم FCFS حل کند بایستی جدول را بر اساس زمان ورود یا ARRIVAL TIME مرتب کند

خب مسئله به اینجا ختم نمی شود بلکه ما باید میانگین زمان انتظار را بدست اوریم

سرویس تایمی که در گانت بالا نوشته ایم برای بدست آوردن میانگین زمان انتظار مورد استفاده قرار می گیرند. بازم تاکیید می کنم که زمان سرویس دهی را بایستی خودتان بدست آورید !

فرمول: زمان انتظار در الگوریتم FCFS

میانگین زمان انتظار= زمان ورود پردازه – سرویس تایم بدست آمده در گانت بالا

P1=0-0=0 P4=13-4=9

P2=10-2=8 P5=17-5=12

P3=12-3=9

Avg=(0+8+9+9+12)/5=7.6

میانگین زمان انتظار از رابطه بالا بدست آمد که به شرح آن میپردازیم

ابتدا زمان سرویس گانت را با زمان ورود کم میکنیم به ترتیب بالا

بعد تمام جواب های بدست آمده را باهم جمع میکنیم و تقسیم بر تعداد میکنیم

مسئله

چون این بخش نیاز به تاکید خاصی ندارد و یه راحل ساده را برایش بیان کردیم برای اینکه خودتان را محک بزنید مسئله زیر را حل کنید

راهنمایی: ابتدا سرویس تایم را بدست آورید و بعد میانگین از طریق آن.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P4 | P3 | P2 | P1 | Process |
| 3 | 2 | 1 | 0 | Arrival Time |
| 6 | 8 | 3 | 5 | Execute Time |

**الگوریتم SJF**

**Shortest Job First الگوریتم کوتاه ترین کار باید اول انجام شود**

این الگوریتم به صورت غیر انحصاری است یعنی آنها را نمیتوان از یک پردازه قبل از اتمام کار گرفت

خب همون طور که از اسم این الگوریتم معلوم است داره میگه آقا کوتاه ترین کار باید اول انجام شود

بریم باهم همون جدول قبلی رو از طریق این الگوریتم انجام بدیم.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P5 | P4 | P3 | P2 | P1 | Process |
| 5 | 4 | 3 | 2 | 0 | Arrival Time |
| 3 | 4 | 1 | 2 | 10 | Execute Time |

دو نکته برای حل این مسئله وجود دارد:

نکته اول این است که اگر زمان رسیدن داشتیم((Arrival Time، از آنجایی شروع میکنیم که زمان رسیدن اولمان صفر باشد بابقی دیگر به AT یا Arrival Time یا زمان رسیدن در این الگورتیم زمانبندی، نگاه نمی کنیم و با توجه به این الگوریتم ( کمترین کار اول) آن پردازه ای که کمترین کار را دارد می نویسیم.

نکته دوم اینکه اگر زمان رسیدن یا Arrival Time نداشتیم، با توجه به حل این نوع الگرویتم (کمترین کار اول) آن پردازه ای که کمترین زمان اجرایی را دارد مینویسیم و به همین ترتیب تا آخرین.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P1 | P4 | P5 | P2 | P3 | Process |
| 0 | 4 | 5 | 2 | 3 | Arrival Time |
| 10 | 4 | 3 | 2 | 1 | Execute Time |

اگر جدول این چنین بود این طوری مینویسیم:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Wait Total | Total Arrival Time | Complete Time | Execute Time | Arrival Time | Process |
| 0 | 10 | 10 | 10 | 0 | P0 |
| 9 | 11 | 13 | 2 | 2 | P1 |
| 7 | 8 | 11 | 1 | 3 | P2 |
| 12 | 16 | 20 | 4 | 4 | P3 |
| 8 | 11 | 16  “ | 3  “ | 5  “ | P4 |
| 7.2 | 11.2 | Avg |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P3** | **P4** | **P1** | **P2** | **P0** |
| **20** | **16** | **13** | **11** | **0-10** |

توصیف جدول:

اولین ستون درباره زمان رسیده هر پردازه توضیح می دهد (پس زمان ورود داریم)

دومین ستون زمان اجرایی هر پردازه است (که معمولا صورت مسئله داده میشود)

سومین ستون باتوجه به گانت رسم شده هر پردازه (زمان کامل شده)آن در ردیف پردازه خودش نوشته می شود.

چهارمین ستون میانگین زمان رسیدنTotal Arrival Time=CT-AT

پنجمین ستون میانگین صبر هر پردازه WT=TAT – ET

مسئله 1) :

باتوجه به جدول زیر میانگین زمان انتظار و میانگین زمان رسیدن را با استفاده از الگوریتم SJF بدست آورید

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| WT | TAT | CT | ET | AT | Process |
| 4 | 5 | 7 | 1 | 2 | P1 |
| 10 | 15 | 16 | 5 | 1 | P2 |
| 3 | 4 | 8 | 1 | 4 | P3 |
| 0 | 6 | 6 | 6 | 0 | P4 |
| 6 | 9 | 11 | 3 | 2 | P5 |

گانت به صورت زیر است:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P2 | P5 | P3 | P1 | P4 |
| 16 | 11 | 8 | 7 | 0-6 |

میانگین زمان انتظار (4+10+3+0+6)=~5

میانگین زمان رسیدن =~ 6.2

مسئله 2) :

داده های پیش فرض مسئله با قرمز نوشته شده است...

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| WT | TAT | CT | ET | AT | Process |
| 0 | 8 | 8 | 8 | 0 | P0 |
| 14 | 19 | 20 | 5 | 1 | P1 |
| 7 | 9 | 11 | 2 | 2 | P2 |
| 17 | 22 | 25 | 5 | 3 | P3 |
| 7 | 11 | 15 | 4 | 4 | P4 |
| 3 | 4 | 9  “ | 1  “ | 5  “ | P5 |
| 8 | ~12 | AVG |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| “ | “ | P3 | P1 | P4 | P2 | P5 | P0 |
| “ | “ | 25 | 20 | 15 | 11 | 9 | 0-8 |

در تمام این مسائل ما زمان رسیدن را داشتیم که از کوچک ترین آنها شروع کردیم.

اما ممکن است در بعضی از مسائل ما Arrival Time را نداشته باشیم

مسئله 3) :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Wait Time | Total Arrival Time | Complete Time | Execute Time | Arrival Time | Process |
|  |  | 9 | 6 | 3 | P1 |
|  |  | 24 | 8 | 16 | P2 |
|  |  | 16 | 7 | 9 | P3 |
|  |  | 3  “ | 3  “ | 0  “ | P4 |
|  |  | AVG |

گانت : اول پردازه ای که کارش از همه کمتر است!

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| “” | “” | P2 | P3 | P1 | P4 |
| “” | “” | 24 | 16 | 9 | 0-3 |

برای بدست آوردن زمان ورود جدولی که زمان ورود را ندارد بایستی عمل تفاضل بین CT با ET انجام گیرد.

خب بعد از این که فرایند تفاضل بین Ct با Et صورت گرفت دیدیم که مسئله دوباره ماننده قبل شده است، در این قسمت از مسئله با یاد گرفتیم که اگر صورت مسئله اطلاعات کمی دهد یعنی فقط زمان اجرا را به ما دهد، ما بازم میتوانیم با بدست آرودن CT و بعد از ترسیم گانت، خواهیم توانست بازم Arrival Time را بدست آوریم...

مسئله 4 ) : داده های مسئله با قرمز نوشته شده است...

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| WT | TAT | CT | ET | AT | Process |
| 10 | 20 | 29 | 10 | 9 | P1 |
| 0 | 2 | 2 | 2 | 0 | P2 |
| 0 | 4 | 6 | 4 | 2 | P3 |
| 0 | 8 | 19 | 8 | 11 | P4 |
| 0 | 5  “” | 11  “” | 5  “” | 6  “” | P5 |
| 2 | AVG |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| “” | P1 | P4 | P5 | P3 | P2 |
| “” | 29 | 19 | 11 | 6 | 0-2 |

**الگوریتم نوبت چرخشی یا Round Robin**

این الگوریتم به حسب یک بره زمانی به نام کوانتوم کار میکند که معمولا در صورت مسئله اعلام میکنند که بره زمانی (کوآنتوم) مورد نظر چقدر است، این الگوریتم انحصاری است، گرسنگی ندارد، به صورت تئوری میتوان اینگونه نوشت که، ابتدا به بره زمانی یا کوآنتوم مسئله نگاه میکنیم و بعد به جدول مورد نظر که هر زمان ورود هم زمان اجرای هر یک در آن نوشته شده است، بعد از آن بر اساس ورود هر فرد مانند الگوریتم FIFO یا FCFS عمل میکنیم، در یک مسئله آن را توضیم میدم

**مسئله راند روبین 1):**

سه پردازه باهم در زمان صفر وارد پردازش میشوند هر کوآنتوم هم برابر با 3 است:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Wait Time | Complete Time | Execute Time | Arrival Time | Process |
| 13-9-0=4 | 13 | 9 | 0 | P1 |
| 5-2-0=3 | 5 | 2 | 0 | P2 |
| 7-2-0=4 | 7 | 2 | 0 | P3 |
| 4 | “ | “ | “ | AVG |

ماجرا این است که در این مسئله همه در زمان صفر وارد شدن و می بینیم که پردازه اول 9 تا زمان برای اجرا نیاز دارد پس با هم بررسی میکنیم: ما چقدر کوآنتوم داریم؟ 3 بره زمانی، چقدر قراره اجرا بشه؟ در پردازش اول 9 تا.. خب ما تا سخت 3 تا میتونیم زمان بدیم در بیشتر باید این پردازه در چرخه قرار گیرد و بعد از هر پردازه ای که کارش تمام شد دوباره از کوآنتوم کم می کنیم، ودر نهایت مشخص میکنیم که در چه بره زمانی ای پردازه به طور کامل اجرا شده تا بتوانیم در واقع میانگین زمان انتظار را بدست آوریم

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P1 | P1 | P3 | P2 | P1 |
| 13 | 10 | 7 | 5 | 0-3 |

خب حالا می رسیم به حل چند تمرین :

تمرین دوم :

ابتدا همان جدول معروف را به عنوان سوال خواهیم دید:

در این مسئله کوآنتوم (بره زمانی) برابر با **یک** است...

هر پردازه ای که به صورت کامل اجرا شود من در جدول آنرا به صورت بُــلد نشان داده ام.

زمان رسیدن حائز اهمیت است، باید به ترتیب ورود پردازه انجام شود

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Waiting Time | Total Arrival Time | Complete Time | Execute Time | Arrival Time | Process |
| 10 | 20 | 20 | 10 | 0 | P1 |
| 3 | 5 | 7 | 2 | 2 | P2 |
| -1 | 0 | 3 | 1 | 3 | P3 |
| 6 | 10 | 14 | 4 | 4 | P4 |
| 4 | 7 | 12  “ | 3  “ | 5  “ | P5 |
| 4.4 | 9.6 | Avg |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **P1** | P1 | P1 | P1 | P1 | P1 | **P4** | P1 | **P5** | P4 | P1 | P5 | P4 | **P2** | P1 | P5 | P4 | **P3** | P2 | P1 |
| **20** | 19 | 18 | 17 | 16 | 15 | **14** | 13 | **12** | 11 | 10 | 9 | 8 | **7** | 6 | 5 | 4 | **3** | 2 | 0-1 |

مثالی بسیار ساده برای حل مسائل راند روبین: شما یه نانوایی رو فرض کن که شاتر به هر نفر ( مانند این مسئله ) فقط میتواند یک نان بدهد، یکی میاد جلو میگه من ده تا نان میخوام شاتر میگه تو یدونتو بگیر برو ته صف تا دوباره نوبتت بشه تا ده تات کامل بشه، اون فرد میره ته صف، یکی دیگه میاد جلو میگه من 2 تا نان میخوام شاتر میگه باشه این یدونه برو ته صف تا کامل بشی، همین ترتیب تا اخر پیش میرود تا جایی دیگر کسی نیازی نداشته باشد و همه به نانشان رسیده باشند، این الگوریتم باعث میشود هیچ پردازه ای گرسنه نماند و عدالت بین آنها باقی می ماند.

**الگوریتم HRRN بالاترین نرخ پاسخ Highest Ratio Response Next**

این الگوریتم انحصاری است

گرسنگی ندارد

بر اساس رابطه پیش میرود که زمان عملیاتی می برد

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Waiting Time | Total Arrival Time | Complete Time | Execute Time | Arrival Time | Process |
| 0 | - | *10* | 10 | 0 | P1 |
| 8 | - | 12 | 2 | 2 | P2 |
| 9 | - | 13 | 1 | 3 | P3 |
| 12 | - | 20 | 4 | 4 | P4 |
| 8 | - | 16 | 3 | 5 | P5 |
| 7.4 | - | Avg |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P4** | **P5** | **P3** | **P2** | **P1** |
| **20** | **16** | **13** | **12** | **0-10** |

**در اولین مرحله خواهیم داشت:**

RR 1= W = 12 – 3 = 9 => W + S / S => 9 + 1 /1 = 10 Check True

RR 2 = W= 12 – 4 = 8 => W + S /S => 8 + 4 / 4 = 3

RR 3 = W= 12 – 5 = 7 => W + S / S => 7 + 3 / 3 = 3.3

**مرحله دوم**

RR 1 = W = 13 – 4 = 9 => W + S / S => 9 + 4 / 4 = 3.25

RR 2 = W = 13 – 5 = 8 => W + S / S => 8 + 3 / 3 = 3.6 Check True

می خوایم متوجه موضوع بشیم :

اقا جریان از این قرار که وقتی به ما مسئله ای دادن که با روش HRRN حل کنید خب اول باید یه گانت رسم کنیم بعد از رسم گانتمون از ابتدای جدول تا مورد دوم آن یعنی ردیف یک و دو رو با توجه به زمان سرویس و انجام آن در گانت مینویسیم. عددی که به وجود می آید را باید با روش W + S / S با ردیف های باقی ماننده محاسبه کنیم و موردی را از حاصل این نتیجه انتخاب می کنیم که بیشترین نسب پاسخ را داشته ( یعنی عددی بیشتره رو انتخاب میکنیم) بعد از انتخاب بیشترین، برای ردیف های بعدی به ترتیب این روش را پیاده میکنیم.

**الگوریتم SRTF یا کوتاه ترین زمان باقی مانده Shortest Remaining Time First**

**فصل ششم**

**بن بست و گرسنگی**

**تعریف بن بست؟**

وقتی چند فرایند و پردازه با هم در ارتباط هستند و برای رسیدن به منابع سیستمی با هم رقابت می کنند تا یکی از دیگری سریع تر برسند پدیده ای بنام بن بست بوجود می آید.

نکته: برای بن بست هیچ راه حل درست حسابی و کارآمدی وجود ندارد.

ساده ترین نوع مثال در بن بست در سیستم عامل این است که وقتی دو یا چند فرایند به صورت مجزا و همزمان بخواهد از منابع فرایند دیگری که خود آن در حال استفاده از آن منابع می باشد و تا زمانی که آنرا تمام نکند منابع را رها نخواهد کرد.

**منابع نقش بسیارزیادی در تعیین بن بست دارند.**

**تا به اینجا چندین منابع را از هر نظر یاد گرفتیم پس باهم یه دوره ای داشته باشیم:**

منابع یک دسته سخت افزاری که قابل دیدن و فیزیکی هستن مانند پرینتر

منابع یک دسته دیگر که به صورت منطقی وجود دارند مانند فایل ها پرونده ها وغیره

منابع انحصاری و غیر انحصاری

جدید..:

منابع یک بار مصرف که نمی توان آنها را در مراتب بعدی دوباره استفاده کرد

منابع قابل استفاده مجدد.

منابع قابل مصرف مجدد منابع هستند که اگر تنها توسط یک فرایند از آنها استفاده کند و در نهایت استفاده از ان به اتمام برسد دیگر تمام نمی شوند می توان از آنها در فرایند های دیگری استفاده نمود. این بدین معناست که این منابع از بین نمی روند بلکه پس از اتمام کار بر روی آنها توسط یک پردازه ا فرایند، در انتها آزاد میشوند.

**مثال هایی از منابع قابل استفاده مجدد:**

کانال های ورودی خروجی، حافظه اصلی و ثانوی، سختمان های داده و دیتابیس ها از این قبیل اند.

**بن بست در منابع قابل استفاده مجدد زمانی رخ میدهد که یک فرایند منبعی را نگهدارد و منبع دیگری را درخواست نماید.**

**مثالی از منباع قابل استفاده مجدد:**

**فرض کنید حافظه ای که به تمام فرایند ها اختصاص میابد 200 کیلوبایت باشد به همین منظور مشخص کنید فرایند زیر بنبست دارند یا خیر؟**

|  |  |
| --- | --- |
| **P2** | **P1** |
| **Request 70kb** | **Request 80kb** |
| **Request 80kb** | **Request 60kb** |

حل مسئله:

همانطوری که در صورت مسئله آمده است به کل فرایند ها حافظه 200 کیلوبایتی اختصاص میابد

اگر فرایند اول اجرا شود و فرایند دوم در همان مرحله اول اجرا شود به خودی خود 150 کیلوبایت را اشغال می کنند، اما در مرحله دوم 140 کیلوبایت کل این دو فرایند روی هم 290 کیلوبایت را اشغال میکنند، پس این عمل باعث ایجاد بن بست میشود

**منابع یکبار مصرف**

منابعی هستند که اگر فرایند از آنها استفاده کند تمام می شوند و دیگر در مراتب بعدی قابل استفاده نیستند.

منابع یکبار مصرف مانند تمام وقفه ها، پیام ها، علامت ها،اطلاعات بافر ورودی و خروجی است.

در صورتی که در آنها بن بستی اتفاق افتد شناسایی آن بسیار مشکل است.

عواملی که باعث ایجاد بن بست میشوند:

انحصار متقابل:

هر پردازه یا فرایند در هر لحظه فقط میتواند از یک منبع استفاده کند.

نگهداشت و انتظار:

اگر پردازه ای منبع جدیدی را درخواست کند و منبع قبلی را آزاد نکند باعث اینجاد بن بست میشوند.

انحصاری بودن:(قبضه نکردن)

نمیتوان آنها را از یک فرایند تا قبل از اتمام کار بروی آنها گرفت.

انتظار چرخشی:

مانند یه حلقه هستند، هر ذنجیره می تواند یک منبع را در خود نگهدارد تا در مراحل بعدی از آن استفاده شود.

**برای جلوگیری از بن بست باید یکی از این 4 شرط بالا نقض شود.**

**نقض انحصار متقابل:**

این شرط را نمی توان نقض کرد چرا که در بیشتر واقع یک پردازه فقط میتواند در هر زمان از یک منبع استفاده کند، برای مثال یک پرینتر فقط میتواند یک پردازه را انجام دهد.

**نقض نگهداشت و انتظار:**

یک پردازه زمانی میتواند منبع جدیدی را دریافت کند که منبع قبلی را ازاد کرده باشد و سپس منبع جدید را دریافت نماید.در غیر این صورت بن بست ایجاد می شود

معایب این روش این است که یک منبع انقدر باید صبر کند تا پردازه منبع قبلی را ازاد کند

به همین خاطر منبع صبر کننده بیکار میماند.

**نقض انحصاری بودن:**

تا زمانی که یک فرایند منبعی را تمام نکرد نمیتواند در خواست منبع جدیدی را داشته باشد.

**نقض انتظار چرخشی:**

در صورتی که انتظار به صورت حلقه ای باشد همواره با بن بست رو به رو خواهیم بود، پس به همین دلیل می بایست منابع را به صورت خطی منظم کنیم، که بزرگترین عیب آن کند شدن فرایند است.

**راه های اجتناب از بن بست:**

در این روش بیشتر سعی در پیشبینی داریم:

1. عدم شروع فرایندی که درخواست هایش موجب بن بست میشود.
2. عدم پاسخ به فرایند هایی که تخصیص منابع به آنها موجب بن بست میشود.(الگوریتم بنکرز)

الگوریتم بنکرز Bankers

زمانی از این الگوریتم استفاده میکنیم که بخواهیم عمل عدم پاسخ به درخواست فرایندی که اخصاص منبع به آن موجب بن بست می شود.

صورت مسئله به شکل زیر است :

آیا حالت زیر امن است ؟

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Resource | | Max | | | | Allocate | | | | Process |
| Qty | Names | U | T | S | R | U | T | S | R | Res |
| 6 | R | 1 | 1 | 1 | 4 | 1 | 1 | 0 | 3 | A |
| 3 | S | 2 | 1 | 2 | 0 | 0 | 0 | 1 | 0 | B |
| 4 | T | 0 | 1 | 2 | 4 | 0 | 1 | 1 | 1 | C |
| 2 | U | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | D |
|  | | 0 | 1 | 1 | 2 | 0 | 0 | 0 | 0 | E |

ابتدا باید از طریق همین جدول تعداد نیاز هر یک از منابع را با محاسبه Need=(Max - Allocate) بدست آوریم.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Need = Max - Allocate | | | | | | Max | | | | Allocate | | | | Process |
| Sum | U | | T | S | R | U | T | S | R | U | T | S | R | Res |
| 2 | 0 | | 0 | 1 | 1 | 1 | 1 | 1 | 4 | 1 | 1 | 0 | 3 | A |
| 4 | 2 | | 1 | 1 | 0 | 2 | 1 | 2 | 0 | 0 | 0 | 1 | 0 | B |
| 4 | 0 | | 0 | 1 | 3 | 0 | 1 | 2 | 4 | 0 | 1 | 1 | 1 | C |
| 1 | 0 | | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | D |
| 4 | | 0 | 1 | 1 | 2 | 0 | 1 | 1 | 2 | 0 | 0 | 0 | 0 | E |

و بعد تمام مقادیر هر منبع را با هم جمع و از Resource کم میکنیم که دارایی هامون بدست بیاد

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | Resource | | | | Available | Qty | Names | | 1 | 6 | R | | 0 | 3 | S | | 2 | 4 | T | | 0 | 2 | U | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Allocate | | | | Process | | U | T | S | R | Res | | 1 | 1 | 0 | 3 | A | | 0 | 0 | 1 | 0 | B | | 0 | 1 | 1 | 1 | C | | 1 | 0 | 1 | 1 | D | | 0 | 0 | 0 | 0 | E | | 2 | 2 | 3 | 5 | Sum | |

بعد از مشخص کردن تمام دارایی هامون از منابع به صورت زیر پیش می رویم.

اونی که نیاز کمتری داره یعنی فرایند D را برای شروع انتخاب می کنیم، اگر جمع نیاز اون سطر از جمع کل دارایی هامون کوچکتر مساوی دارایی هامون بود مقدار Allocate را به Available اضافه میکنیم. این مرحله تا اخر پیش می رود، اگر دربین آنها مقداری بود که از دارایی هامون بیشتر میشد اعلام می کنیم که این مسئله امن نیست.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Need = Max - Allocate | | | | | | Allocate | | | | Process |
| Sum | U | | T | S | R | U | T | S | R | Res |
| 2 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | 3 | A |
| 4 | 2 | | 1 | 1 | 0 | 0 | 0 | 1 | 0 | B |
| 4 | 0 | | 0 | 1 | 3 | 0 | 1 | 1 | 1 | C |
| 1 | 0 | | 1 | 0 | 0 | 1 | 0 | 1 | 1 | D |
| 4 | | 0 | 1 | 1 | 2 | 0 | 0 | 0 | 0 | E |

نکته: در الگوریتم بانکداران چون ما سعی در پیشبینی در آینده را داریم، نمی توان مطمئن شویم که این راهبرد کاملا درست یا غلط است، بعضی وقتا درست ، بعضی وقتا غلط عمل میکند.

**معایب الگوریتم بانکداران:**

حداکثر منابع باید از قبل مشخص شده باشد.

هر فرایند باید به صورت مستقل عمل کند.

منابع تخصیصی باید به صورت ثابت باشد.

فرایندی که منبعی را در درست دارد نمیتواند خارج شود.

**کشف بن بست:**

یکی از مهمترین وظایف سیستم عامل، کشف خطا میباشد، که مشخص میکند آیا فرایندی انجام شود؟ یا آیا منبعی تخصیص یابد؟ یا خیر!

**راهبردهای کشف خطا:**

اگر بن بستی رخ داد، تمام فرایند ها متوقف میشوند، فرایند ها به نقطه شروع خودشان بر میگردند، منابع به صورت انحصاری، و قبضه شدن پی در پی تخصیص میابد.

راهبرد مجتمعی برای بن بست وجود ندارد چرا که هر کدام از آنها مزایا و معایب خودشان را دارند،

بهترین راهبرد این است، که عمل مرتب سازی خطی صورت گیرد، منابع در گروه های مختلف تقسیم شود.

دو نکته مهم در خصوص قبضه کردن، بایستی ذکر شود:

1. اگر فرایندی منبعی را در اختیار داشته باشد و تازمانی که کار برروی آن را به اتمام نرسانده باشد، وآنرا آزاد نکرده باشد، هیچ حقی ندارد که منبعی را درخواست کند.
2. اگر فرایند 1 منبعی را درخواست کند که آن منبع در دست فرایند 2 باشد CPU فرایند 2 راقبضه کرده و منبع آنرا به فرایند 1 میدهد.

**تفاوت بین، اجتناب از بن بست، پیشگیری از بن بست و کشف بن بست**

در پیشگیری از بن بست سعی میشود که یکی از 4 شرط بروز بن بست، نقض شود، یا بصورت مستقیم از انتظار مدور جلوگیری شود، یا به صورت غیر مستقیم از سه شرط گفته شده جلوگیری شود.

در اجتناب از بن بست دو شرط بیان میشود که

1. عدم شروع به فرایندی که پاسخ به درخواست هایش موجب بن بست می شود
2. عدم پاسخ به فرایند هایی که تخصیص منابع به آنها موجب بن بست خواهد شد.

در کشف بن بست بیشتر سعی میشود، که اجازه دهد کدام فرایند صورت گیرد و چه منابعی اختصاص دهد،

و به صورت مستقیم در حال بررسی عدم بروز انتظار مدرو هست، در این صورت از تمام فرایند ها جلوگیری میشود تا ابتدا انتظار مدور به ترتیب خطی تغییر حالت دهد.

**فصل هفتم مدیریت حافظه**

چرا نیاز به مدیریت حافظه داریم؟

حفاظت، اشتراک، جابجایی، سازمان منطقی، سازمان فیزیکی، از جمله اهداف مدیریت حافظه اند.

حافظه به چند دسته تقسیم میشود؟

حافظه به 4 دسته تقسیم می شود:

حافظه ایستا

حافظه پویا

جابجایی

سیستم های رفاقتی

**حافظه به دو دسته تقسیم میشود:**

1. قسمتی برای خود سیستم عامل
2. قسمتی برای برنامه های تحت اجرای کاربر

نکته: برای اجرای چندین برنامه در یک سیستم عامل خود بخش کاربر هم باید به چندین قسمت تقسیم بشود تا سیستم عامل بتواند تعداد زیادی از برنامه ها در خودش جای دهد.

**مدیریت حافظه:**

مدیریت حافظه یکی از مهمترین وظایف سیستم عامل می باشد، که وظیفه مدیریت و تقسیم بندی و بخشبندی حافظه به زیر بخش هایی را بر عهده دارد.

مهمترین دلیل مدیریت حافظه مربوط به **جابه جایی** آن است.

برنامه نویس نمیداند برنامه که نوشته است و اکنون درحال اجراست، در کجای حافظه قرار میگیرد.

ممکن است برنامه درحال اجرای برنامه نویس ابتدا برروی دیسک اجرا شود و بعد در یک جایی از حافظه منتقل شود.

**حفاظت:**

یک فرایند نمیتواند به اطلاعات فرایند دیگری دسترسی داشته باشد.

نیاز های امنیتی-محافظتی توسط سخت افزار برآورده میشود نه توسط نرم افزار.

**اشتراک گذاری:**

اشتراک گذاری باعث میشود که پردازه ها یا فرایند ها در هرجایی که هستند به حافظه دسترسی داشته باشند.

**پارگی یا پراکندگی داخلی:**

در واقع فضای خالی در داخل پارتیشن را پارگی داخلی گویند.

**پارگی خارجی:**

بعد از اینکه هر پردازه ای کارش را انجام داد و از لیست خارج شد به فضای باقی مانده پارگی خارجی گویند.

**الگوریتم جایگذاری بخش استاتیک**

الگوریتمی که بدو دسته تقسیم میشود:

بخش مساوی: بدیل مساوی بودن تمام بخش ها مهم نیست که کدام بخش برای جایگذاری انتخاب میشود

بخش نامساوی: قسمتی کوچکی که میتوان هر فرایند را در آن جایگذاری نمود.

این الگوریتم درک راحتی دارد:

به این مثال توجه کنید:

سیستم عامل

8M

سیستم عامل

8M

**P2 : 1M**

**P2 : 7M**

**P3 : 8M**

8M

8M

8M

**P1 : 5M**

**3M**

**الگوریتم جایگذاری بخش بندی پویا:**

الگوریتم بخشبندی پویا به 4 دسته تقسیم میشود:

First Fit اولین برازش

Next Fit درپی برازش

Best Fit بهترین برازش

Worst Fit بدترین برازش

در First Fit پردازه ها را در اولین مراجعه جایگذاری میکنیم به ترتیب ادامه دارد.

در Next Fit پردازه ها در اولین جایی که قرار بگیرند از آن به بعد جایگذاری میکنیم.

در Best Fit حافظه ای که کمترین پارگی را دارد برای جای گذاری انتخاب میکنیم.

در Worst Fit حافظه ای که بیشترین پارگی را دارد برای جایگذاری انتخاب میکنیم.

**بهترین برازش از هر نظر First Fit است چرا که از اول شروع به پیمایش میکند و از همان ابتدا جایگذاری را انجام می دهد.**

مثال )

4 پردازه با اندازه های مختلف، میخواهند وارد حافظه شوند انها را با 4 الگوریتم بالا جایگذاری کنید.

**First Fit :**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Value | process | | 40MB | P1 | | 10MB | P2 | | 50MB | P3 | | 5MB | P4 | |  |  | | 4 | 3 | 2 | 1 | MB/Level |
| 5 | 10 | 10 | 20-10 | 20 |
| 10 | 10 | 10 | 10 | 10 |
| 10 | 10 | 10 | 10 | 50-40 |
| 30 | 30 | 30 | 30 | 30 |
| 20 | 20 | 70-50 | 70 | 70 |
| 5 | 5 | 5 | 5 | 5 |

**Next Fit :**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Value | process | | 40MB | P1 | | 10MB | P2 | | 50MB | P3 | | 5MB | P4 | |  |  | | 4 | 3 | 2 | 1 | MB/Level |
| 20 |  |  | 20 | 20 |
| 10 |  |  | 10 | 10 |
| 0 |  | 0 | 10-10 | 50-40 |
| 30 |  | 30 | 30 | 30 |
| 15 | 20 | 70-50 | 70 | 70 |
| 5 | 5 | 5 | 5 | 5 |

**Best Fit**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Value | process | | 40MB | P1 | | 10MB | P2 | | 50MB | P3 | | 5MB | P4 | |  |  | | 4 | 3 | 2 | 1 | MB/Level |
| 20 | 20 | 20 | 20 | 20 |
| 0 | 0 | 0 | 10-10 | 10 |
| 10 | 10 | 10 | 10 | 50-40 |
| 30 | 30 | 30 | 30 | 30 |
| 20 | 20 | 70-50 | 70 | 70 |
| 0 | 5 | 5 | 5 | 5 |

**Worst Fit**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Value | process | | 40MB | P1 | | 10MB | P2 | | 50MB | P3 | | 5MB | P4 | |  |  | | 4 | 3 | 2 | 1 | MB/Level |
| 20 |  | 20 | 20 | 20 |
| 10 |  | 10 | 10 | 10 |
| 35 | Fault P3 | 40 | 50-10 | 50 |
| 30 |  | 30 | 30 | 30 |
| 30 |  | 30 | 30 | 70-40 |
| 5 |  | 5 | 5 | 5 |

**تفاوت آدرس دهی منطقی، نسبی، فیزیکی:**

آدرس دهی منطقی درواقع همان ادرس ارجاع شده یا لینک شده است.

ادرس دهی نسبی یک حالت و نمونه خاصی از ادرس دهی منطقی است مثلا به مکان معینی لینک شده.

آدرس دهی فیزیکی در واقع ادرس حقیقی و واقعی هست که واقع در آنجا حضور دارد.

برنامه های کاربر هر کدام به قسمت های مساوی و با اندازه یکسانی تقسیم میشوند که به هر قطعه تقسیم شده صفحه گفته میشود، با هر بار تقسیم بندی، حافظه هم به همان اندازه مساوی تقسیم می شود که به هر بخش تقسیم شده حافظه قاب گفته میشود، هر صفحه اندازه قاب مخصوص به خودش است.

از طرفی دیگر برنامه های کاربر هم به قطعاتی با اندازه های مختلف تقسیم می شوند که هیچ لزومی ندارد که تمام قطعات اندازه یکسانی با یکدیگر داشته باشند، که به این عمل قطعه بندی گفته می شود.این قطعه بندی مانند قطعه بندی بخش پویا هستند.