پرسش ۱

نوع فرآیند هایی که در هر سیستم اجرا میشود، هرروز در حال تغییر است. پس مدلی برای تعیین زمان اجرای فرآیند ها وجود ندارد. با این حال تعداد فراخوانی های پردازشگر و دستگاه های ورودی و خروجی قابل مدلسازی است و میتوان توزیعی از آن بدست آورد. همچنین توزیع زمان ورود فرآیند ها به سیستم نیز قابل مدلسازی است. با استفاده از این دو میتوان کمیت های متفاوتی نظیر زمان انتظار را برای الگوریتم های مختلف زمانبندی محاسبه کرد. برای مثال، اگر n طول صف فرآیند های در انتظار برای پردازش باشد و W را میانگین زمان انتظار فرآیند های موجود در صف در نظر گرفته و در نهایت، Lمیانگین زمان ورود فرآیند ها به صف باشد، انتظار میرود در زمان W که یک فرآیند، پردازش میشود L×W فرآیندی جدید وارد صف میگردد. اگر سیستم در حالت پایدار خود باشد، آنگاه تعداد فرآیند های ورودی به صف با تعداد فرآیند ای خروجی از آن برابر است، لذا $n = L \times W$:این تساوی که به Little s formula معروف بوده برای هر الگوریتم زمانبندی و هر توزیع L قابل تطبیق است که با استفاده از آن میتوان با داشتن یکی از دو کمیت، سومی را محاسبه نمود.

پرسش ۲

یکی از چالش‌های زمانبندی زمانی به وجود می‌آید که یه فرایند با اولویت بالا خواستار خواندن یا تغییر یک منبع کرنل سیستم عامل باشد. در صورتی که این منبع در اختیار یک فرایند با اولویت پایین تر است. از آن جا که منابع کرنل محافظت شده هستند، فرایند با اولویت بالا قادر به دسترسی نیست و تا پایان کار فرایند کم‌اولویت با منبع، باید صبر کند. وضعیت زمانی وخیم تر میشود که فرایند کم اولویت تحت اختیار فرایندی با اولویت بالاتر باشد. لذا فرایندی که منتظر استفاده از منبع است، تا اتمام کار دو فرایند دیگر باید منتظر بماند. این پدیده، وارونگی اولویت نام دارد و در سیستم های دارای بیش از دو اولویت اتفاق میافتد. یک راه حل برای این مشکل، تقلیل اولویت ها به ۲ است؛ که در سیستم عامل های چند منظوره به صرفه نیست. رویکرد دیگر، پروتکل ارث بری اولویت نام دارد، به این صورت که تمام فرایند های دارای دسترسی به یک منبع که مورد درخواست فرایندی با اولویت بالاتر است، به صورت موقت، به اولویت این فرایند، ترفیع اولویت داده میشوند، لذا در مثال فوق، فرایند L به اولویت H ترفیع مییابد و M نمیتواند که اجرای آن را تحت اختیار بگیرد. پس از اتمام اجرای L منبع مورد درخواست، به جای M به دست رسی H در میآید و اولیت L به حالت قبل برمیگردد.

پرسش ۵

ایجاد فرایند در سیستم های یونیکس با استفاده از دستور سیستمی fork انجام میپذیرد. هنگامی که یک فرایند درخواست fork میدهد، سیستم عامل مراحل زیر را طی میکند: ابتدا یک خانه از جدول فرایندها را به فرایند جدید اختصاص میدهد، یک شماره منحصر به فرد به فرایند جدید اختصاص میدهد، یکی کپی از فرایند پدر بدون حافظه ی اشتراکی میسازد، تمام شمارنده های فایل های پدر را بعلاوه یک میکند تا یک فرایند جدید نیز آنها را در اختیار داشته باشد، فرایند جدید را در حالت آماده اجرا قرار میدهد، شماره اختصاصی فرایند جدید را به پدر داده و مقدار ۰ به فرایند فرزند میدهد. وقتی این اعمال در هسته سیستم اجرا شد، یکی از سه کار زیر انجام میشود: ۱ -پردازش در فرایند پدر ادامه مییابد ۲ -فرایند جدید، شروع به اجرا میکند ۳ -فرایند پدر و فرزند هردو در حالت آماده مانده و پردازش به فرایند دیگری اختصاص مییابد.

پرسش ٦

در **لینوکس** ترکیبی از سه زمان بندی SCHED_FIFO، SCHED_RR و SCHED_NORMAL است. که به ترتیب صف بلادرنگ(real-time) ، round-robin بلادرنگ و دیگر الگوریتم های غیر بلادرنگ (non-real-time) استفاده میشود. که اولویت اجرا با الگوریتم های بلادرنگ است. برای این که بتواند این مسئله را هندل کند به کلاس های بلادرنگ عددی بین ۰ تا ۹۹ را نسبت میدهد و به کلاس‌های غیربلادرنگ عددی بین ۱۰۰ تا ۱۳۹ را میدهد که هر چه قدر اندازه اعداد کوچکتر باشد اولویت اجرا بالاتر است.

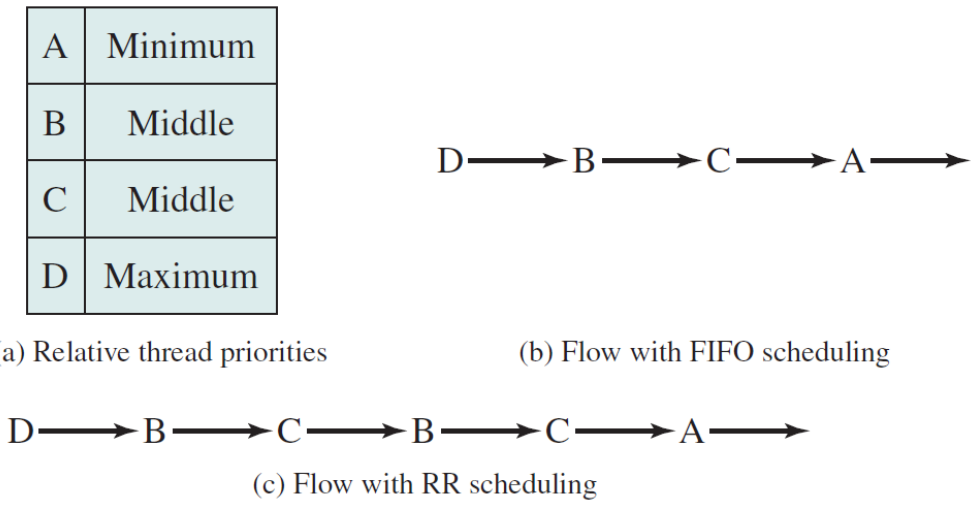
برای روش صف قوانین زیر اجرا میشود:

۱) در غیر از سه حالتی که در ادامه می‌آوریم سیستم نمیتواند وقفه ای در فرآیند ایجاد کند؛ یک. FIFO دیگری با الویت بالاتری آماده باشد. دو.FIFO در حال اجرا به خاطر I/O متوقف شود. سه. FIFO در حال اجرا به صورت داوطلبانه CPU را رها کند.

۲) هنگامی که وقفه‌ای رخ میدهد FIFO در حال اجرا از CPU خارج شده به صف اولویت وارد میشود و متناسب با اولویت خود در صف قرار میگیرد.

۳) چنانچه FIFO ای آماده اجرا شود. و از FIFO در حال اجرا اولویت بیشتری داشته باشد؛ در آن وقت FIFO در حال اجرا بالاترین اولویت FIFO آماده اجرا را میگیرد.

برای روش round-robin هم قوانینی مشابه صف وجود دارد. به غیر از این که یک تایم اسلایس ها به ماجرا اضافه میشوند تصویر زیر نمونه از انجام فرآیندهای لحظه ای به دو روش round-robin و FIFO است.



Example of Linux Real-Time Scheduling

ضمنا SCHED_NORMAL در فرایندهای غیر بلادرنگ (non-real-time) استفاده میشود که در هر سیستمی میتواند متفاوت باشد.

در **ویندوز**، طراحی به گونه ای است که در حد امکان پاسخگوی نیاز های یک کاربر واحد یا یک سرور باشد.زمانبندی در ویندوز به صورت round-robin منعطف است. اولویت زمانبندی به دو گروه تقسیم میشود: یک. بلادرنگ و دو. متغیر. هر کدام از این گروه ها به ۱۶ سطح مختلف تقسیم میشوند. threadهای مورد نیاز و فوری در دسته لحظه ای قرار میگیرند چون ویندوز از یک برنامه زمانبندی پیشگیرانه مبتنی بر اولویت استفاده میکند. threadهای بلادرنگ از سایر threadها اولویت بیشتری دارند. هنگامی که یک thread آماده به کار میشود و اولویت اجرایی آن از thread در حال اجرا بالاتر است؛ موضوع با اولیت پایین تر مجددا اولویت بندی شده و پردازنده به thread با اولویت بالاتر داده میشود.

نکته مهم این است که اولویت thread های بلادرنگ ثابت است و قابل تغییر نیست. اما اولویت در threadهای متغیر همه threadها مقدار اولویت اولیه ای دارند که در طول عمر آن thread ممکن است افزایش پیدا کند. و برای هندل کردن این موضوع از صف اولویت دار استفاده میشود.

پرسش ۸

دو نوع سیستم بلادرنگ وجود دارد:

Soft Real Time Systems :گارانتی ای برای زمان برنامه ریزی یک فرایند حیاتی بی‌درنگ وجود ندارد.

Hard Real Time Systems : این نوع سیستم ها فرآیندها قبل از به پایان رسیدن زمانشان میبایستی سرویس دهی شده باشند.

الگوریتم های مربوط به زمان بندی:

Priority-Based Scheduling:ابتدا به هر فرایند اولویتی داده میشود؛ و شروع میکند به اجرای بالاترین اولویت. و چنانچه قابلیت پس‌گرفتن CPU از فرایند فراهم باشد، چنانچه فرآیندی اولویت بالاتری به خود بگیرد؛ آن فرایند در مرحله اجرا قرار میگیرد. حال سوال پیش می‌آید که چه بلایی سر فرایندهایی با اولویت برابر می‌آید؟ جواب این است که فرایندهایی با اولویت برابر به صورت time sharing و با استفاده از round-robin اجرا میشوند.

Rate-Monotonic Scheduling: در این روش هر فرآیندی که وارد سیستم میشود، زمانی برای پایان دارد (deadline). ضمنا مطابق به دوره تناوب انجام فرآیند به آن اولویتی داده می‌شود و هر فرآیندی که دوره تناوب بیشتری دارد، اولویت کمتری دارد؛ فی‌الواقع با دوره تناوب و اولویت با هم رابطه عکس دارند. چنانچه فرایندی با اولویت بالاتر وارد شود؛ CPU به فرآیند با اولویت بالاتر داده میشود. در این الگوریتم به این صورت است که فرآیندی که بیشتر پردازنده میخواهد بیشتر پردازنده میگیرد (اولویت بالاتری دارد).

Earliest-Deadline-First Scheduling : در این روش هم مانند فرآیند بالا ابتدا به هر فرایند deadline داده میشود. و در این روش به هر فرایندی که deadline آن نزدیک‌تر است اولویت بالاتری داده میشود. ضمنا در این مدل اولویت ها قابلیت تغییر به صورت داینامیک دارد.

Proportional Share Scheduling : در این روش ابتدا سهم های زمانی که سیستم عامل میتواند به همه فرایند ها اختصاص دهد را حساب میکنیم. و سهم هر فرایند نیز مشخص میشود؛ در واقع میدانیم هر فرایند چه درصدی از زمان پردازنده در اختیار هر فرایند خواهد بود. این نکته باعث میشود که هنگامی که میخواهیم یک فرآیند را وارد کنیم، قبل از آن بررسی کنیم که آیا به اندازه سهم زمانی آن فرآیند سهم زمانی موجود است؟ تنها اگر موجود بود این فرآیند را اضافه کنیم.

POSIX Real Time Scheduling : در این روش چندین برنامه‌ریزی متفاوت داریم میگیرد دو نوع SCHED_FIFO و SCHED_RR برای فرایند های بلادرنگ و SCHED_OTHERS برای فرایند های غیر بلادرنگ اختصاص یافته است(همانند لینوکس در سوال قبلی) در FIFO فرایندهایی با اولویت بالاتر اجرا میشود و Time-sharing اتفاق نمی‌افتد.(یعنی فرایندهای هم اولویت همزمان اجرا نمیشوند) اما در RR فرایندهای هم اولیت به صورت Time-sharing اجرا میشود. و برای فرایند های غیر بلادرنگ در هر سیستم سیاستی متفاوت در پیش گرفته میشود.

پرسش ۱۰

Foreground application: برنامه‌ای که هنگام اجرا بر صفحه نمایش نشان داده میشود و کاربر از اجرای آن باخبر میگردد.

Background application: برنامه هایی که اجرا میشوند ولی به در صفحه نمایش نشان داده نمیشوند و کاربر به صورت عادی اجرای آن را نمیبیند.

IOS: در این سیستم عامل در ابتدای امر امکان اجرای چند برنامه به صورت همزمان وجود نداشت (و فقط خود سیستم عامل و یک برنامه دیگر قابل اجرا بود) و با اجرای یک برنامه به صورت Foreground application دیگر برنامه ها در حالت suspended قرار میگرفتند. اما با ارائه IOS 4 امکان اجرای چند فرآیندِ همزمان به طور محدودی ایجاد شد. به این صورت که برنامه های محدودی این امکان را داشتند و یک برنامه به صورت Foreground application و دیگر برنامه ها به صورت Background application اجرا میشدند. با ارائه نسخه‌های بعدی این سیستم عامل، امکان چند فرآیندی پیشرفت کرد به طوری که در نسخه های جدید امکان اجرای چند برنامه به صورت Foreground application وجود دارد.

Android: این سیستم عامل از ابتدا دارای قابلیت multitasking بود و محدودیتی برای Background application ها نداشت. این برنامه میتوانند از سرویس ها استفاده کنند که برنامه ایست که در background اجرا میشود و این امکان را حتی به برنامه های suspended میدهد که از منابع استفاده کنند و برخی از فعالیت های خود را انجام دهند.

پرسش ۱۱

هنگامی که یک فرآیند در کامپیوتر تمام میشود؛ در state خاصی میرود که به آن termination میگویند. و به این معنااست که این فرآیند دیگر نیازی به CPU ندارد. از طرفی هر فرآیند میتواند تعدادی فرآیند فرزند داشته باشد. سیستم عامل اجازه نمیدهد که هیچ فرآیندی بدون پدر بماند و چنانچه فرآیند پدر terminate شود؛ فرآیندهای فرزند آن نیز terminate میشوند به این نوع تمام شدن یک فرآیند cascadeing termination گفته میشود.