به نام خدا

دانشکده مهندسی کامپیوتر

درس سیستم‌های عامل
نیم‌سال دوم ۰۳-۰۴
استاد: دکتر جلیلی

تمرین شماره ۲

---

۱. قطعه کد زیر را در نظر بگیرید:

```
pid_t pid;
pid = fork();
if (pid == 0) {
    fork();
    thread_create(...);
}
fork();
```

الف. با اجرای این قطعه‌کد، چه تعداد پردازه[1] مجزا ساخته می‌شود؟

ب. با احرای این قطعه‌کد، چه تعداد ریسه[2] مجزا ساخته می‌شود؟

۲. لینوکس بین پردازه‌ها و ریسه‌ها تمایز قائل نمی‌شود. در عوض، لینوکس با هر دو به یک شکل رفتار می‌کند و به یک عملیات پردازشی اجازه می‌دهد بسته به مجموعه پرچم‌های ارسال شده به فراخوان سیستمی ()clone بیشتر شبیه یک پردازه یا یک ریسه باشد. با این حال، سایر سیستم‌عامل‌ها، مانند ویندوز، با پردازه‌ها و ریسه‌ها به طور متفاوتی رفتار می‌کنند. به طور معمول، چنین سیستم‌هایی از یک نشانه گذاری استفاده می‌کنند که در آن ساختار داده برای یک پردازه شامل اشاره‌گرهایی به ریسه‌های جداگانه‌ای است که به آن پردازه تعلق دارند. این دو رویکرد برای مدل‌سازی پردازه‌ها و ریسه‌ها در داخل هسته را با یکدیگر مقایسه کنید.

۳. یک پردازنده چند هسته‌ای و یک برنامه چند ریسه‌ای نوشته شده با استفاده از مدل چند_به_چند تخصیص ریسه‌ها [3] را در نظر بگیرید. فرض کنید تعداد ریسه‌های سطح کاربر در برنامه بیشتر از تعداد هسته‌های پردازشی در سیستم است. سناریوهای زیر را از نظر کارایی[4] بررسی کنید.

الف. تعداد ریسه‌های کرنل تخصیص داده شده به برنامه کمتر از تعداد هسته‌های پردازشی باشد.

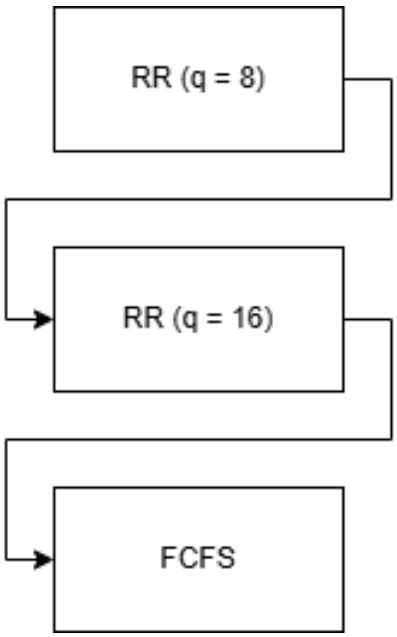
ب. تعداد ریسه‌های کرنل تخصیص داده شده به برنامه برابر با تعداد هسته‌های پردازشی باشد.

ج. تعداد ریسه‌های کرنل تخصیص داده شده به برنامه بیشتر از تعداد هسته‌های پردازشی اما کمتر از تعداد ریسه‌های سطح کاربر باشد.

۴. درستی و یا نادرستی هر کدام از موارد زیر را با ذکر دلیل بیان کنید:

---

[1] Process
[2] Thread
[3] Many-to-many threading model
[4] Performance



شکل ۱: یک نمونه MLFQ

الف. پس از صدا زدن ()exit توسط ریسه اصلی یک پردازه، سایر ریسه‌ها برای اجرا تا پایان زمان خود وقت دارند.

ب. در مدل چند_به_یک تخصیص ریسه‌ها، تمام ریسه‌های سطح کاربر به یک ریسه سطح هسته نگاشت می‌شوند و باعث می‌شود حتی در سیستم‌های چند پردازنده، تنها یک ریسه در هر لحظه اجرا وجود داشته باشد. این مدل بهترین کارایی را در برنامه‌های با پردازش زیاد و محدود به پردازنده مرکزی دارد.

ج. در مدل یک_به_یک تخصیص ریسه‌ها، هر ریسه سطح کاربر به یک ریسه سطح هسته نگاشت می‌شود و امکان اجرای هم‌زمان چند ریسه بر روی چند هسته فراهم می‌شود، اما این مدل به دلیل سربار زیاد در ایجاد ریسه‌های هسته‌ای، برای برنامه‌های با تعداد ریسه زیاد بهینه نیست.

۵. به سوالات زیر پاسخ دهید.

الف. نحوه کار زمان‌بندی اولویت ثابت[5] را شرح دهید.

ب. چگونه می‌توان الگوریتم EDF را با استفاده از یک زمان‌بند اولویت ثابت شبیه‌سازی کرد؟

ج. در قسمت ب در چه زمان‌هایی نیاز به محاسبه مجدد اولویت‌ها می‌شود؟

د. چگونه می‌توان الگوریتم FIFO را با استفاده از الگوریتم RR شبیه‌سازی کرد؟

ه. با توجه به MLFQ شکل ۱ و با فرض اینکه پردازه‌ها به طور پیوسته وارد این زمان‌بند می‌شوند:

- دو حالت را مثال بزنید که اجرای تمامی پردازه‌ها، محدود به لایه‌ی اول باقی بماند.
- فرض کنید پردازه شماره ۱، به طور پیوسته و هر ۱۰ ثانیه یک بار به زمان‌بند وارد می‌شود. مراحل دو بار اجرا شدن این پردازه را توضیح دهید.

۶. فرض کنید در سیستمی پنج پردازه در انتظار اجرا هستند. زمان اجرای پیش‌بینی شده برای آن‌ها به ترتیب (از چپ به راست) x ، ۵، ۳، ۶، ۹ است. ترتیب اجرای این پردازه‌ها در سه حالت زیر چگونه باشد تا میانگین زمان برگشت[6] آن‌ها به حداقل برسد؟

[5] Strict priority
[6] Turnaround time

الف. $۶ < x < ۹$

ب. $۳ < x < ۵$

ج. $۹ < x$

۷. یک الگوریتم زمان‌بندی مداخله‌جویانه[7] را در نظر بگیرید که بر اساس تغییر پویای اولویت‌ها عمل می‌کند. اعداد اولویت بزرگتر به معنای اولویت بالاتر است. هنگامی که یک پردازه در انتظار پردازنده است (در صف آماده، اما در حال اجرا نیست)، اولویت آن با نرخ α تغییر می‌کند، و هنگامی که در حال اجرا است، اولویت آن با نرخ β تغییرمی‌کند. تمام پردازه‌ها هنگام ورود به صف آماده، اولویت ۰ دریافت می‌کنند. پارامترهای α و β را می‌توان به گونه‌ای تنظیم کرد که الگوریتم‌های زمان‌بندی مختلفی به دست آید.

الف. الگوریتمی که از $\beta > \alpha > 0$ حاصل می‌شود چیست؟

ب. الگوریتمی که از $\alpha < \beta < 0$ حاصل می‌شود چیست؟

۸. فرض کنید یک سیستم SMP دارای صف‌های اجرای خصوصی برای هر پردازنده است. هنگامی که یک پردازه جدید ایجاد می‌شود، می‌تواند در همان صف پردازه والد یا یک صف جداگانه قرار گیرد.

الف. مزایای قرار دادن پردازه جدید در همان صف پردازه والد چیست؟

ب. مزایای قرار دادن پردازه جدید در یک صف متفاوت چیست؟

۹. یک سیستم کنترل صنعتی حساس به زمان را در نظر بگیرید که پردازه‌ها مختلفی با محدودیت‌های زمانی متفاوت دارد. این سیستم شامل سه نوع پردازه است:

الف. **پردازه‌های بحرانی:** این پردازه‌ها باید در یک بازه زمانی نو سختگیرانه به پایان برسند. عدم اتمام به موقع این پردازه‌ها می‌تواند منجر به نقص فاجعه‌بار در سیستم شود. مثال: خواندن سنسورهای ایمنی و فعال‌سازی مکانیزم‌های اضطراری.

ب. **پردازه‌های مهم:** این پردازه‌ها دارای محدودیت‌های زمانی ملایم‌تری هستند. از دست دادن مهلت انجام این پردازه‌ها مطلوب نیست اما منجر به نقص فاجعه‌بار نمی‌شود، بلکه ممکن است کیفیت عملکرد سیستم را کاهش دهد. مثال: به‌روزرسانی وضعیت نمایشگر اپراتور.

ج. **پردازه‌های غیرضروری:** این پردازه‌ها محدودیت زمانی خاصی ندارند و می‌توانند با تاخیر اجرا شوند. مثال: ثبت داده‌های عملکرد سیستم برای تجزیه و تحلیل‌های بعدی.

سیستم از یک زمان‌بند اولویت ثابت با الگوریتم نرخ یکنواخت[8] برای تخصیص اولویت به پردازه‌ها استفاده می‌کند. فرض کنید دوره‌های تناوب پردازه‌ها بحرانی کوتاه‌تر از پردازه‌ها مهم و پردازه‌ها مهم کوتاه‌تر از پردازه‌ها غیرضروری است، که مطابق با تخصیص اولویت RM است (دوره‌ی تناوب کوتاه‌تر، اولویت بالاتر).

حال فرض کنید یک رویداد غیرمنتظره در سیستم رخ می‌دهد که باعث می‌شود یکی از پردازه‌ها غیرضروری به طور ناگهانی بار محاسباتی بسیار سنگینی پیدا کند و زمان اجرای آن به طور قابل توجهی افزایش یابد.

با توجه به ساختار زمان‌بندی و انواع پردازه‌ها در این سیستم، تحلیل کنید که افزایش ناگهانی زمان اجرای پردازه غیرضروری چه تاثیری بر قابلیت زمان‌بندی پردازه‌ها بحرانی و مهم خواهد داشت؟ آیا ممکن است این وضعیت منجر به نقض محدودیت‌های زمانی پردازه‌ها بحرانی شود؟ استدلال خود را با در نظر گرفتن مفاهیم مربوط به تداخل پردازه‌ها[9] و استفاده از پردازنده مرکزی در سیستم‌های بلادرنگ[10] بیان کنید. همچنین، پیشنهاد دهید که چگونه می‌توان با استفاده از مکانیزم‌های

[7] Preemptive
[8] Rate-monotonic (RM)
[9] Task Interference
[10] Real-time

موجود در سیستم عامل‌های بلادرنگ (مانند مکانیسم‌های محدودسازی منابع یا تغییر پویای اولویت) از بروز چنین مشکلاتی جلوگیری کرد یا تاثیر آن‌ها را کاهش داد.

۱۰. دستور nice در لینوکس و همچنین سایر سیستم‌عامل‌های یونیکس برای تنظیم مقدار "nice" یک پردازه استفاده می‌شود. توضیح دهید که چرا برخی از سیستم‌ها ممکن است به هر کاربری اجازه دهند تا مقدار "nice" یک پردازه را بزرگتر یا مساوی ۰ تعیین کند، اما فقط به کاربر root اجازه دهند تا مقادیر "nice" کمتر از ۰ را تعیین کند.