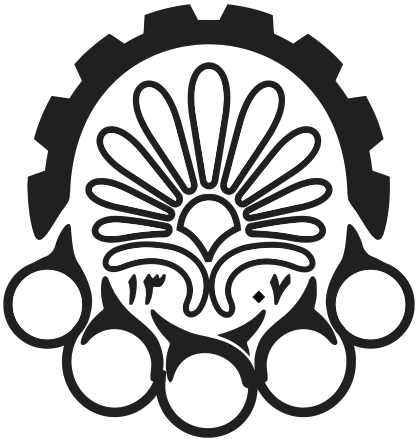
سیستم‌های عامل

دکتر زرندی



دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

رضا آدینه پور ۴۰۲۱۳۱۰۵۵

تمرین سری چهار

۹ آبان ۱۴۰۳

سیستم‌های عامل

تمرین سری چهار

رضا آدینه پور ۴۰۲۱۳۱۰۵۵

دانشکده مهندسی کامپیوتر

سوال اول

در خصوص انواع فرایندها به سوالات زیر پاسخ دهید.

۱. فرایند فرزند چگونه منابع مورد نیاز خود را تامین می‌کند؟ آیا می‌تواند از منابع والد استفاده کند؟

پاسخ

مطابق با توضیحات آقای Silberschatz در صفحه ۱۱۷، فرایند فرزند در هنگام ایجاد توسط سیستم‌عامل می‌تواند منابع خود را به چندین روش تأمین کند:

(آ) **تخصیص منابع از والد:** در بسیاری از سیستم‌عامل‌ها، فرایند فرزند می‌تواند بخشی از منابع فرایند والد را به ارث ببرد. به عنوان مثال، اگر والد دارای فایل‌های باز، حافظه و یا دسترسی به برخی دستگاه‌ها باشد، این منابع ممکن است به فرایند فرزند به ارث برسند.

(ب) **تخصیص منابع جدید:** فرایند فرزند می‌تواند از سیستم‌عامل درخواست منابع جدید کند، مانند حافظه اضافی یا دسترسی به فایل‌ها. این منابع از منابع کلی سیستم اختصاص می‌یابند و مستقل از والد هستند.

(ج) **به ارث بردن حافظه و داده‌ها:** معمولاً در سیستم‌های مبتنی بر UNIX، فرایند فرزند یک کپی از فضای حافظه والد خود را به ارث می‌برد. این به معنای آن است که فرایند فرزند در ابتدا از داده‌ها و متغیرهای والد خود کپی‌ای مستقل دارد.

۲. همانطور که می‌دانید فرایند فرزند ممکن است پیش از اتمام اجرا توسط فرایند والد به پایان برسد. توضیح دهید که فرایند والد به چه دلایلی ممکن است تصمیم بگیرد فرایند فرزند پایان یابد.

پاسخ

فرایند والد ممکن است به دلایل زیر تصمیم بگیرد فرایند فرزند خود را خاتمه دهد:

(آ) **خطا در اجرای فرزند:** اگر فرایند فرزند دچار خطا یا مشکل شود (مثلاً دسترسی غیرمجاز به حافظه)، والد ممکن است تشخیص دهد که بهتر است فرزند را خاتمه دهد.

(ب) **نیاز به بازپس‌گیری منابع:** در صورت نیاز به منابع بیشتر، فرایند والد می‌تواند فرزند را پایان دهد تا منابع مصرفی آن آزاد شوند و به والد یا سایر فرایندها اختصاص یابند.

(ج) **پایان زودتر از موعد والد:** در برخی شرایط، اگر والد تصمیم بگیرد زودتر از موعد پایان یابد، ممکن است فرزند نیز خاتمه پیدا کند، چرا که فرایند فرزند دیگر بدون والد قابل اجرا نیست.

پاسخ

(د) **دستور صریح از والد:** فرایند والد می‌تواند به طور مستقیم و با دستوراتی مانند `kill` یا `terminate` در سیستم‌های مبتنی بر UNIX فرایند فرزند را پایان دهد.

۳. هنگامی که فرایند والد به دستور `wait()` می‌رسد، چه اتفاقی رخ می‌دهد؟

پاسخ

هنگامی که فرایند والد به دستور `wait()` می‌رسد، اتفاقات زیر رخ می‌دهند:

(آ) **توقف اجرای والد:** فرایند والد به حالت Waiting می‌رود و منتظر می‌ماند تا فرایند فرزند خاتمه یابد.

(ب) **انتظار برای پایان فرزند:** سیستم‌عامل اجرای والد را به حالت تعلیق در می‌آورد تا زمانی که فرایند فرزند به پایان برسد.

(ج) **دریافت کد بازگشتی فرزند:** پس از اتمام فرایند فرزند، کد بازگشتی آن به والد منتقل می‌شود. این کد می‌تواند وضعیت خروجی فرایند فرزند را نشان دهد، مثلاً موفقیت یا شکست فرایند.

(د) **بازگشت والد به اجرا:** بعد از اینکه فرزند خاتمه یافت و والد اطلاعات لازم را دریافت کرد، والد دوباره به حالت Ready بازگشته و اجرای آن ادامه می‌یابد.

سوال دوم

در خصوص زمانبندها به سوالات زیر پاسخ دهید.

۱. آیا در همه سیستم‌های عامل از همه انواع زمانبندها موجود می‌باشد؟ اگر پاسخ شما بله است علت لزوم وجود انواع زمانبند را توضیح دهید و اگر پاسخ شما خیر است شرح دهید که کدام یک از زمانبندها می‌توانند نباشند.

پاسخ

خیر، در همه سیستم‌های عامل همه انواع زمانبندها وجود ندارند. سه نوع اصلی زمانبند در سیستم‌های عامل وجود دارد که در ادامه توضیح مختثری درباره هرکدام می‌دهیم:

(آ) **Long-term Scheduler:** این زمانبند مسئول تصمیم‌گیری درباره‌ی اینکه کدام فرایندها باید وارد صف Ready شوند، است. این زمانبند در سیستم‌های Multiprogramming که نیاز به مدیریت تعداد زیادی فرایند دارند، استفاده می‌شود. اما در سیستم‌های تعاملی یا بلادرنگ (مثل بسیاری از سیستم‌های کامپیوترهای شخصی)، زمانبند بلندمدت معمولاً وجود ندارد.

(ب) **Medium-term Scheduler:** این زمانبند مسئول مدیریت جابجایی (Swapping) فرایندها بین حافظه و دیسک است. سیستم‌هایی که از حافظه مجازی یا چندبرنامه‌ای استفاده می‌کنند و نیاز به کنترل دقیق استفاده از حافظه دارند، به زمانبند میان‌مدت نیاز دارند. اما در سیستم‌هایی که حافظه کافی برای اجرای تمامی فرایندهای فعال وجود دارد، زمانبند میان‌مدت ممکن است نباشد.

(ج) **Short-term Scheduler:** این زمانبند مسئول تخصیص CPU به فرایندهای آماده در صف است. زمانبند کوتاه‌مدت در همه سیستم‌های عامل وجود دارد، زیرا در هر سیستم عملیاتی نیاز به تخصیص CPU به فرایندهای در حال اجرا وجود دارد.

بنابراین، در سیستم‌های تعاملی معمولاً زمانبند بلندمدت وجود ندارد، و در سیستم‌هایی با حافظه کافی ممکن است زمانبند میان‌مدت هم وجود نداشته باشد.

۲. مشخص کنید در هر یک از موارد زیر کدام یک از زمانبندها مسئول انجام وظیفه داده شده‌است.

وضعیت	زمانبند مسئول
کنترل تعادل میان I/O bound و CPU bound	Long-term Scheduler
Swap out میان فرایندها	Medium-term Scheduler
تخصیص CPU به یکی از فرایندهای آماده	Short-term Scheduler

سوال سوم

مسیر اجرای کد زیر را در گراف حالت فرایند (Process state) از شروع اجرا تا پایان اجرا مشخص کنید. توجه کنید سیستمی که این قطعه در آن اجرا می‌شود تک پردازنده می‌باشد.

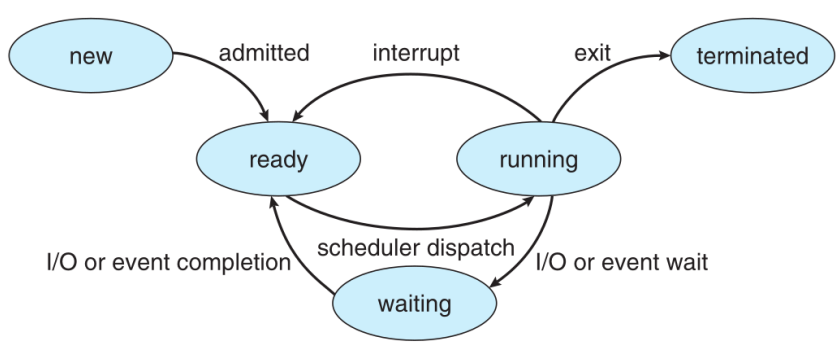
```c
int main()
{
    int n;
    scanf("%d", &n);
    n *= 10;
    printf("%d", n);

    return 0;
}
```

Listing 1: Code of Q3

پاسخ

مطابق با تعریف آقای Silberschatz در صفحه ۱۰۸، گراف حالت فرآیند به صورت زیر است:



شکل ۱: ساختار Process State

طبق این تعریف می‌توان غرایند اجرای این کد را به‌صورت زیر نوشت:

۱. **New:**
فرآیند در وضعیت New شروع می‌شود، زمانی که برنامه (یعنی این کد) ایجاد می‌شود، اما هنوز برای اجرا پذیرفته نشده است. در این مرحله، سیستم‌عامل فرآیند را در حافظه تنظیم کرده و آن را برای اجرا آماده می‌کند.

۲. **Ready:**
پس از ایجاد فرآیند، به وضعیت Ready می‌رویم. و در انتظار تخصیص پردازنده می‌مانیم. این وضعیت نشان می‌دهد که فرآیند در حافظه بارگذاری شده و برای اجرا آماده است، اما منتظر است تا پردازنده به آن اختصاص داده شود.

۳. **Running:**
پس از اینکه زمان‌بند پردازنده این فرآیند را انتخاب کرد، به وضعیت Running وارد می‌شود. در اینجا فرآیند مراحل زیر را طی می‌کند:

ادامه پاسخ ۳

- تابع `scanf()` منتظر ورودی است، بنابراین در حالی که منتظر ورودی کاربر است، فرآیند ممکن است به طور موقت به وضعیت در انتظار Waiting منتقل شود (اگر ورودی زمان‌بر باشد یا به I/O نیاز داشته باشد).
- پس از دریافت ورودی، فرآیند مقدار `n` را در ۱۰ ضرب می‌کند که یک محاسبه است و در این مرحله در وضعیت در حال Running باقی می‌ماند.
- سپس تابع `printf()` برای نمایش نتیجه فراخوانی می‌شود که شامل عملیات I/O است. اگر در عملیات I/O تأخیر باشد، ممکن است به طور موقت دوباره به وضعیت Waiting بازگردد.

۴. **Waiting:**
این وضعیت زمانی اعمال می‌شود که فرآیند نیاز دارد برای یک عملیات I/O (مانند `scanf()` یا `printf()`) منتظر بماند. فرآیند ممکن است به طور موقت در وضعیت Waiting باشد اگر عملیات I/O با تأخیر انجام شود یا منتظر ورودی کاربر باشد، اما پس از اتمام I/O دوباره به آماده یا در حال اجرا بازمی‌گردد.

۵. **Terminated:**
پس از اتمام تمام دستورات (رسیدن به `return 0;`) فرآیند وارد وضعیت Terminated می‌شود، به این معنی که اجرای آن به پایان رسیده و از سیستم خارج می‌شود.

سوال چهارم

در هر عمل تعویض متن (context switch)، میان دو ریسمان متعلق به یک پردازه، چه مواردی باید ذخیره و بازیابی شوند؟ در صورتی که این عمل میان دو پردازه انجام شود چطور؟ با توجه به پاسخ خود نتیجه‌گیری کنید که چرا در موارد زیادی استفاده از ریسمان‌ها به جای پردازه‌ها در سیستم می‌تواند سودمند باشد.

پاسخ

۱. **در ریسمات باید موارد زیر ذخیره و بازیابی شود:**

(آ) ثبات‌های پردازنده (CPU Registers): شامل شمارنده برنامه (Program Counter)، اشاره‌گر پشته (Stack Pointer) و سایر ثبات‌های عمومی.

(ب) پشته ریسمان: هر ریسمان دارای پشته مخصوص به خود است؛ بنابراین، وضعیت پشته باید ذخیره و بازیابی شود.

(ج) اطلاعات وضعیت ریسمان مانند وضعیت اجرای ریسمان، اولویت و اطلاعات زمان‌بندی.

نکته: چون ریسمان‌ها در یک پردازه مشترک هستند، فضای آدرس حافظه، فایل‌های باز و منابع سیستم بین آن‌ها مشترک است و نیازی به تغییر یا مدیریت مجدد آن‌ها در تعویض متن نیست.

۲. **در تعویض متن بین دو پردازه باید:**

(آ) ثبات‌های پردازنده (CPU Registers): مشابه تعویض متن بین ریسمان‌ها.

(ب) فضای آدرس حافظه: باید فضای آدرس حافظه به پردازه جدید تغییر کند؛ این شامل تغییر جداول صفحه (Page Tables) و تنظیمات واحد مدیریت حافظه (MMU) می‌شود.

(ج) منابع پردازه: فایل‌های باز، محیط پردازه، اطلاعات امنیتی و سایر منابع اختصاصی پردازه باید مدیریت شوند.

(د) پشته پردازه: پشته و داده‌های مرتبط با پردازه جدید باید بارگذاری شوند.

نکته: تعویض متن بین پردازه‌ها پیچیده‌تر است زیرا هر پردازه دارای منابع و فضای آدرس حافظه مجزاست.

سوال پنجم

توضیح دهید در خروجی قطعه کد زیر چه تعداد * چاپ خواهد شد؟ همچنین درختواره آن را نیز رسم نمایید.

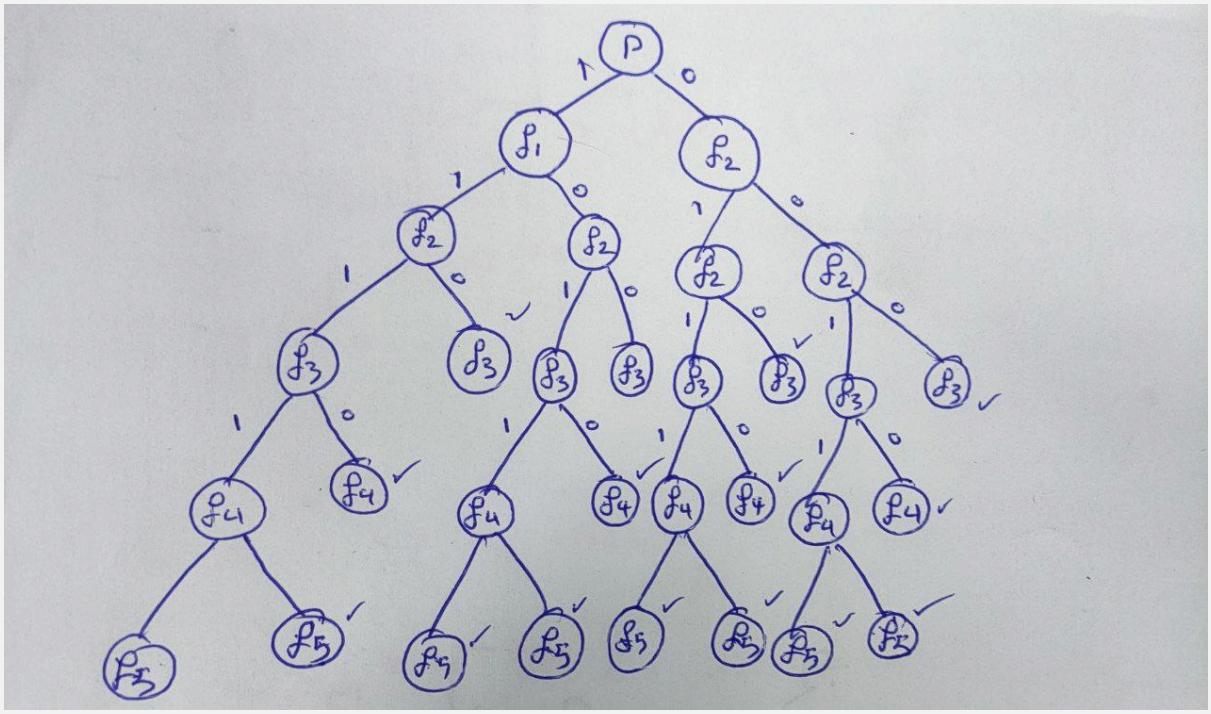
```c
int main()
{
    if (fork() || (!fork()))
    {
        if (fork() && fork())
        {
            fork();
        }
    }
    while (wait(NULL) > 0);
    printf("* ");

    return 0;
}
```

Listing 2: Code of Q3

پاسخ

با توجه به درختواره زیر و جایگاه تابع printf() در کد، ۹ عدد * چاپ خواهد شد و درختواره آن نیز به‌صورت زیر است:



شکل ۲: درختواره این سوال

پاسخ

در این کد، ۱ پردازه والد و ۱۵ پردازه فرزند داریم.
در شرط اول، ۲ پردازه با نام‌های f_1 و f_2 ایجاد می‌شود. در شرط دوم، پردازه f_4 ایجاد می‌شود اما پردازه f_5 فقط در پردازه‌های والد تولید می‌شود تا شرط حلقه ارضا شود و وارد آن شویم و درنهایت هم پردازه f_5 هم برای پردازه‌های والد قبلی خود ایجاد می‌شود.