به نام خدا

پاسخ تمرین سری چهارم درس سیستم های عامل

پاییز 1403

استاد درس: دکتر زرندی

سوال اول)

در خصوص انواع فرایندها به سوالات زیر پاسخ دهید.

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

به طور کلی هنگامی که فرایندی یک فرایند فرزند ایجاد می‌کند آن فرایند فرزند به منابع خاصی (مثل زمان اجرا رویCPU حافظه فایل ها و دستگاه هایI/O) برای اجرا نیاز دارد. یک فرآیند فرزند می‌تواند منابع مورد نیاز خود را مستقیماً از سیستم عامل گرفته و یا ممکن است به زیر مجموعه ای از منابع فرایند والد محدود شود.

ب) 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؟

فرایند والد ممکن است اجرای فرایندهای فرزند خود را به دلایل مختلفی خاتمه دهد مانند موارد زیر:

- **فرایند فرزند از منابعی که به او اختصاص داده شده بیش از حد استفاده کرده است. (برای تعیین اینکه آیا این اتفاق افتاده است یا خیر فرایند والد می‌بایستی مکانیزمی برای بررسی وضعیت فرزندان خود داشته باشد.)**
- **وظیفه ای که به فرایند فرزند محول شده دیگر نیاز به انجام آن نیست.**
- **فرایند والد در حال پایان اجرا است و سیستم عامل به فرایند فرزندی که اجرای والد آن پایان یافته اجازه ادامه اجرا را نمی دهد.**

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

زمانی که فرایند والد به دستور wait می‌رسد، program counter در همان‌جا می‌ماند و به خط بعدی نمی‌رود تا فرزندان به دستورخروج در کد خود برسند و به سیستم عامل اتمام کار خود را خبر دهند. سپس سیستم عامل به والد خبر می‌دهد تا والد بتواند به خط بعدی برود و ادامه پیدا کند.

سوال دوم)

در خصوص زمانبندها به سوالات زیر پاسخ دهید.

الف) 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.

سیستم عامل‌های مختلف ممکن است انواع مختلفی از زمانبندها را بر اساس اهداف و الزامات طراحی خود پیاده سازی کنند. با این حال مفهوم اساسی داشتن زمانبند بلندمدت، کوتاه مدت و گاهی اوقات میان مدت در بسیاری ازسیستم عامل‌ها یکسان است. البته در سیستم عامل‌های time-sharing زمانبندهای بلندمدت وجود ندارند.

ب) مشخص کنید در هریک از موارد زیر کدام یک از زمانبدها مسئول انجام وظیفه داده شده‌است.

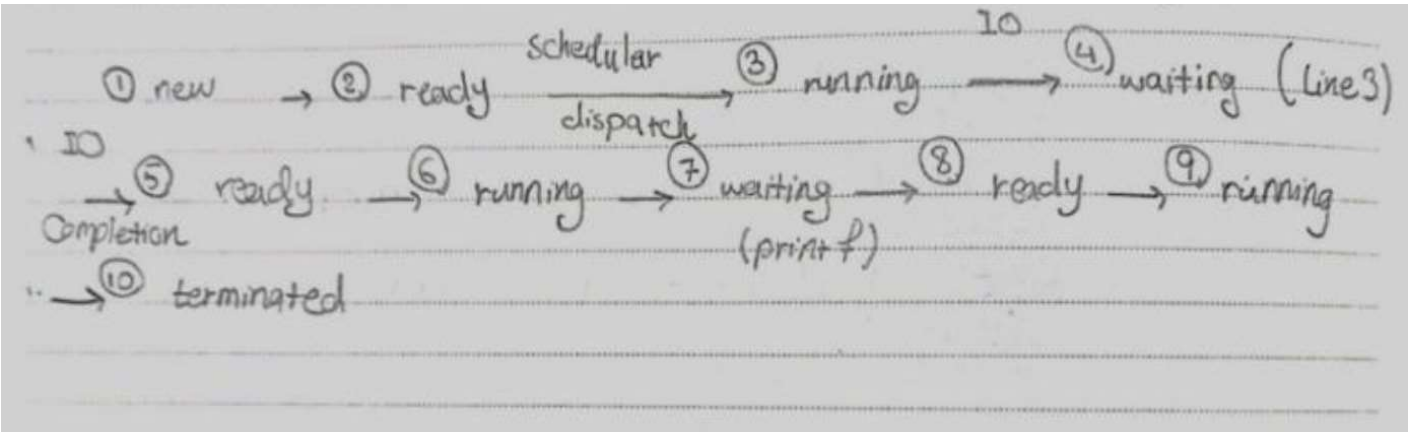
وضعیت	زمانبند مسئول
کنترل تعادل میان I/O bound و CPU bound	**زمانبند بلندمدت**
Swap out میان فرایندها	**زمانبند میان مدت**
تخصیص CPU به یکی از فرایندهای آماده	**زمانبند کوتاه مدت**

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 کد در آن اجرا می‌شود تک پردازنده می‌باشد.

```
int main(){
    int n ;
    scanf("%d", &n) ;
    n *= 10 ;
    printf("%d", n) ;
    return 0 ;
}
```

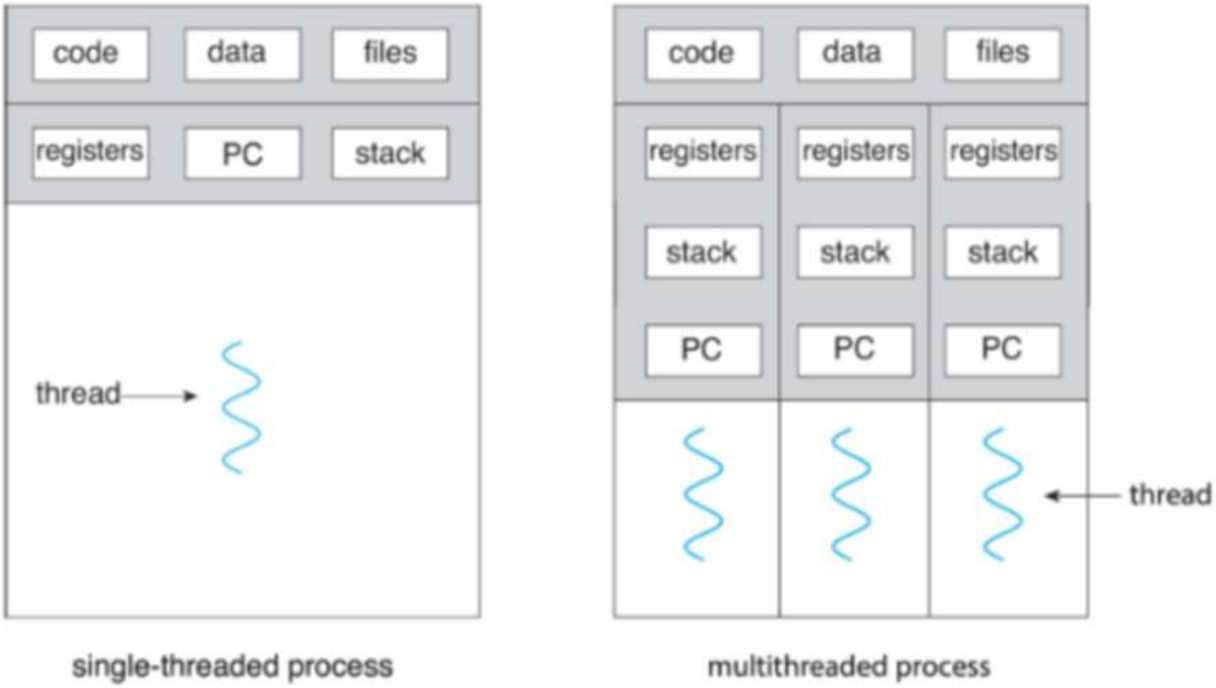
پاسخ:



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 نتیجه گیری کنید که چرا در موارد زیادی استفاده از ریسمان‌ها به جای پردازه‌ها در سیستم می‌تواند سودمند باشد.

در شکل زیر اطلاعات ذخیره شده برای یک پردازه مجزا و تعدادی ریسمان مربوط به یک پردازه نمایش داده شده است.



single-threaded process

multithreaded process

همانطور که در این شکل مشاهده می‌شود آنچه که ریسمان‌های مربوط به یک پردازه را از یکدیگر مجزا می‌سازد مقادیر درون رجیسترها، فضای پشته مربوط به ریسمان و **program counter** برنامه است. بنابراین زمانی که عمل **context switch** میان دو ریسمان مربوط به یک پردازه انجام می‌شود تنها کافی است همین اطلاعات ذخیره و بازیابی شوند.

از طرفی یک پردازه مجزا نسبت به پردازه‌های دیگر به طور کلی فضای حافظه و منابع مجزایی دارد. بنابراین در زمان **context switch** میان دو پردازه علاوه بر موارد اشاره شده در بالا، فضای آدرس برنامه نیز تغییر خواهد کرد و می‌بایستی کل **PCB** برنامه از جمله کل منابع تخصیص داده شده به پردازه مانند لیست فایل‌هایی که باز هستند و **I/O** ها، داده‌های مربوط به مدیریت حافظه و ... نیز ذخیره و بازیابی شوند. می‌دانیم عمل **context switch** خود برای سیستم سربار به همراه دارد، از طرفی با توجه به توضیحات داده شده می‌توان گفت سربار این عمل هنگامی که میان دو ریسمان مربوط به پردازه انجام می‌شود بسیار کمتر از زمانی است که میان دو پردازه مجزا انجام شود. بنابراین استفاده از ریسمان‌ها به جای پردازه‌ها تا جای ممکن می‌تواند به بهبود سرعت عملکرد سیستم کمک کند.

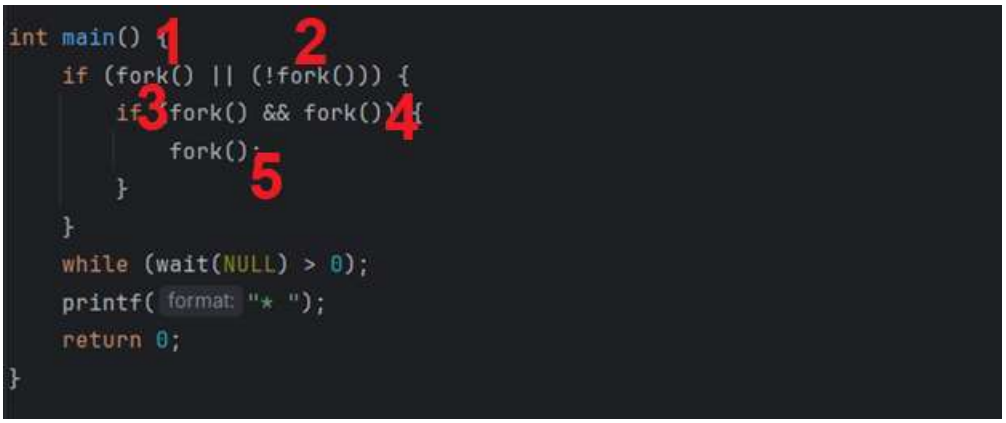
سوال پنجم)

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

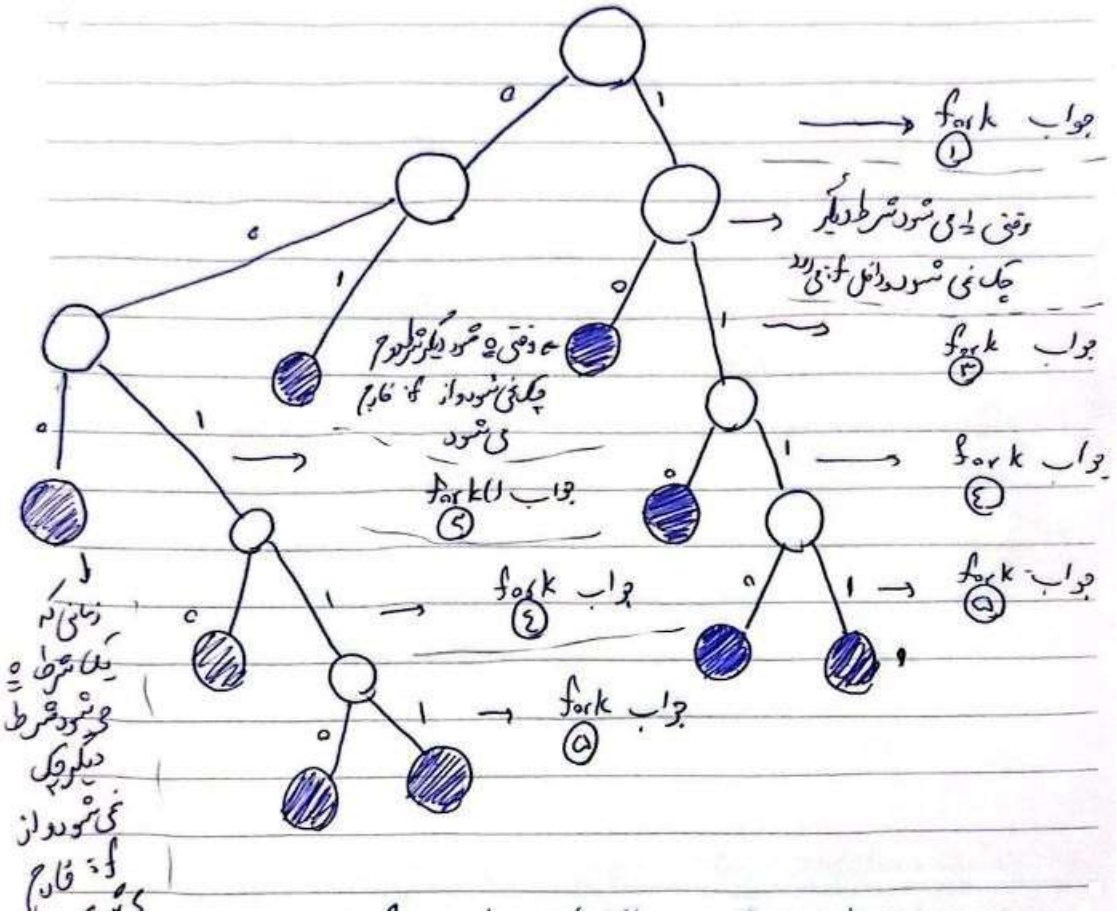
```
int main() {
    if (fork() || (!fork())) {
        if (fork() && fork()) {
            fork();
        }
    }
    while (wait(NULL) > 0);
    printf( format: "* ");
    return 0;
}
```

پاسخ:

تعداد * های چاپ شده در خروجی برابر با 9 خواهد بود. دستورات فورک را به صورت زیر نامگذاری کنید:



درختواره آن به صورت زیر خواهد بود:


جواب fork ①
وقتی این شرط دیگر چک نمی شود وارد آن نمی شود
جواب fork ③
وقتی این شرط دیگر چک نمی شود وارد آن نمی شود
جواب fork ②
جواب fork ④
جواب fork ⑤
جواب fork ⑥
جواب fork ⑤
زمانی که هیچ شرطی چک نمی شود وارد آن نمی شود