شروع	چهارشنبه، 26 آبان 1400، 5:00 عصر
وضعیت	پایان‌یافته
پایان	چهارشنبه، 26 آبان 1400، 7:00 عصر
زمان صرف شده	1 ساعت 59 دقیقه
جمع نمره	4.35 از 5.00
نمره	**8.70** از 10.00 (**87**٪)

سؤال 1

نادرست

نمره 0.00 از 0.25

تعریف مدهای متنوع در سیستم‌های کامپیوتری یک مکانیزم صرفا نرم افزاری برای ایجاد protection است

یک گزینه را انتخاب کنید:

- صحیح ✖
- غلط

پاسخ درست گزینهٔ «غلط» است.

سؤال 2

درست

نمره 0.25 از 0.25

در پیاده‌سازی مفسر خط فرمان (Command Line Interpreter) بهتر است دستورات خط فرمان، built-in باشد، یعنی جزئی از پیاده‌سازی خط فرمان باشد.

یک گزینه را انتخاب کنید:

- صحیح
- غلط ✔

پاسخ درست گزینهٔ «غلط» است.

سؤال 3

درست

نمره 0.25 از 0.25

اگر یک برنامه را از خط فرمان shell اجرا کنیم و قبل از اتمام آن، shell را ببنیدم برنامه مورد نظر یتیم می شود

یک گزینه را انتخاب کنید:

- صحیح
- غلط ✖

پاسخ درست گزینهٔ «صحیح» است.

دیدگاه:

سؤال 4

درست

نمره 0.25 از 0.25

طراحی لایه ای سیستم عامل به نسبت طراحی یکپارچه، performance پایینتر و انعطاف پذیری بالاتری را فراهم میکند

یک گزینه را انتخاب کنید:

- صحیح ✔
- غلط

پاسخ درست گزینهٔ «صحیح» است.

سؤال 5

کامل

نمره 1.00 از 1.00

الف) توضیح دهید PCBدر کجا ذخیره می‌شود و چه نقشی در تعویض متن پروسسها دارد؟

ب) آیا جهت تعویض اجرای سرویس روتین اینتراپت به جای پروسسی که قبل از وقوع اینتراپت در CPU در حال اجرا بوده است، از PCB استفاده میشود؟ اگر بله به چه صورت و اگر خیر پ تعویض متن در این مواقع چگونه انجام می‌شود.

soal5.pdf

الف) ۰.۵ نمره

ب) (۰.۵ نمره) وقتی حین اجرای پروسسی، اینتراپت رخ میدهد در حالت کلی پس از اجرای سرویس روتین اینتراپت قصد داریم دوباره به اجرای پروسس قبلی برگردیم بنابراین نیاز نیست تمام اطلاعات pcb آپدیت شوند کافیست اطلاعات مورد نیاز حین اجرای پروسس در استکی push شوند و پس از اجرای سرویس روتین، دوباره این موارد از استک popشود و اجرای پروسس از سر گرفته شود لذا در این حالت تعویض متن بدان معنی که در تعویض متن پروسسها وجود دارد نداریم و کار تعویض متن از پروسس به اینتراپت و برعکس هم سریعتر و با سربار کمتری انجام میگیرد

دیدگاه:

سؤال 6

کامل

نمره 1.70 از 2.00

الف) یک سیستم تعاملی که زمانبندی نوبت گردشی (Round Robin) را به کار می برد بدین صورت عمل می‌کند: پس از کامل شدن یک دور اجرای همه پروسس‌های آماده (time slice ،(ready هر پروسس آماده موجود را برای دور بعدی اجرا محاسبه می‌کند. این time slice برای هر پروسس از تقسیم ماکزیمم زمان انتظار دور قبلی بر تعداد پروسسهای آماده فعلی بدست می‌آید. این سیاست را چگونه می‌توان توجیه کرد؟ یا این سیاست منطقی است؟ بحث کنید (از تحلیل تغییرات پارامترهای زمان بندی مثل متوسط زمان انتظار، درصد بهره وری سیستم، متوسط زمان پاسخ و ... استفاده کنید)

ب) به بهترین نحوی که میتوانید توضیح دهید یا نشان دهید الگوریتم **CFS** که برای زمان بندی در لینوکس استفاده می‌شود عادلانه است. سپس توضیح دهید که آیا ممکن است شرایطی در سیستم ایجاد شود که گرسنگی برای برخی پروسس‌ها با اجرای این الگوریتم به وجود آید؟ (اگر بله، توضیح دهید چه شرایطی و اگر خیر، بنویسید چرا)

ج) زمان ورود و اندازه burst تعدادی پروسس در زیر مشخص شده است. **متوسط زمان انتظار و متوسط زمان پاسخ** را برای هر کدام از الگوریتمهای **Shortest Remaining Time First و Round Robin با کوانتوم ۴** با رسم جزییات گانت چارت و انجام محاسبات لازم بدست آورید.

پروسس	زمان ورود	اندازه burst
p1	1	4
p2	0	10
p3	3	2
p4	4	3

soal6.pdf

الف) (۰.۵ نمره) فقط تا وقتی که تعداد پروسسها نسبتا کم باشد این سیاست منطقی است. وقتی کوانتوم زمانی کوچکتر میشود تا تعداد پروسسهای بیشتری در مدت مشخصی پاسخ داده شوند، دو اتفاق می افتد: ۱) میزان بهره وری CPU کاهش میباید. در یک لحظه ای کوانتوم اونقدر کوچک میشود که اکثر درخواستها در این زمان تکمیل نمیشود و پروسسها باید دفعات زیادی در صف نوبت - چرخشی قرار گیرند پس اگر پروسسها IO bound باشند و تعدادشان هم کم باشد این الگوریتم خوب است از این جهت که تلاش میکند متوسط زمان پاسخ و انتظار را کاهش دهد اما اگر تعداد پروسسها خیلی زیاد شود و پروسسها CPU bound باشند بهره وری کم میشود و متوسط زمان انتظار هم میتواند بالا رود چون دفعات زیادی هر پروسس باید در صف بماند تا دوباره نوبتش شود و burstش پایان یابد البته سیستم مورد سوال یک سیستم تعاملی است که پروسسها در آن IO bound هستند پس تا وقتی که تعداد پروسسها خیلی زیاد نشود این راهکار نسبتا منطقی است

ب) (۰.۵ نمره) از آنجایی که این الگوریتم سعی میکند یک بازه زمانی مشخص را بین پروسسهای موجود سیستم با توجه به اولویتشان تقسیم کند، عادلانه است. میزان سهم هر پروسس در این بازه زمانی بر اساس اولویت و میزان CPU که تا به حال استفاده کردند میباشد. لذا از بین پروسسهای با الویت یکسان، انهایی که تاکنون cpuکمتری گرفته اند سهم بیشتری میگیرند تا میزان cpu ی که گرفته اند کم کم در اندازه همدیگر شود. از طرفی چنین پروسسهایی زودتر از بقیه هم cpuمیگیرند که این هم با توجه به اینکه تا به حال cpu کمتری دریافت کرده اند عادلانه است
گرسنگی ممکن است به وجود اید اگر مرتبا پروسسهای جدیدی به سیستم اضافه شوند که چون runtime آنها صفر است اگر اولویت بالایی هم داشته باشند بللافاصله بعد ورود cpu میگیرند و این کار میتواند با ورود چنین پروسسهایی تکرار شود و پروسسهای قبلی سیستم گرسنه بمانند
ج) انمره

دیدگاه: الف) ۰.۴
ب) ۰.۳ توضحات زیادی که اصلا در سوال پرسیده نشده بود تمرکز را باید روی پاسخدهی به علت عدالت میگذاشتید جواب اصلی در بین توضیحات گم است.
ج)۱

سؤال 7

کامل

نمره 0.90 از 1.00

این کد را به صورتی تکمیل کنید که **دقیقا** اهداف زیر برآورده شود و نتیجه اجرا به صورتی باشد که شرح داده شده است:

میخواهیم کلمه printf را درون ۴ فایل بزرگ جستجو کرده و خطهای حاوی کلمه موردنظر را در ۴ فایل جداگانه ریخته و نهایتا محتوای این ۴ فایل را پس از این که هر ۴ فایل خروجی تکمیل شدند در خروجی استاندارد چاپ کنیم. در این جستجو جهت **راحتی کار**، از grep و **جهت تسریع جستجو** از مالتی پروسسینگ استفاده میکنیم. (فرض کنید تابع cat_file محتوای یک فایل را در خروجی استاندارد چاپ میکند)

```
int main(int argc, char **argv)
{
    pid_t pid[4];
    char fname[4][50];
    char fout[4][50];
    for(int i=0; i<4;i++){
        sprintf(fout[i],"out_%d.txt", i+1);
    }


            execl("/bin/grep", "grep "," printf", fname[i] , " > ", fout[i], NULL);




    for(int i=0; i<4; i++)
        cat_file(fout[i]);

    printf("--end of main program--\n");
    return 0;
}
```

soal7.pdf

```c
int main(int argc, char **argv)
{
    pid_t pid[4];
    char fname[4][50];
    char fout[4][50];
    for(int i=0; i<4;i++){
        sprintf(fout[i],"out_%d.txt", i+1);
    }
    for(int i=0; i<1;i++){
        scanf("%s", fname[i]);
        pid[i] = fork();
        if (pid[i]==0){
            execl("/bin/grep", "grep "," printf", fname[i],NULL);

        }
    }

    for(int i=0; i<4; i++)
        wait(NULL);
    for(int i=0; i<4; i++)
        cat_file(fout[i]);

    printf("--end of main program--\n");
    return 0;
}
```

اگر پروسسهای اضافه ساخته میشود ۰.۳ کم شده
اگر مقدار i برای هر پروسس فرزند حین اجرای exex صحیح نباشد ۰.۱ شده

چهار waitقبل از چاپ نتیجه نیاز است. در صورتی که کلا wait نگذاشته باشید ۰.۳ و در صورتی که بعد چاپ گذاشته باشید ۰.۲ کم شده است.

دیدگاه: والد در حلقه whileآخر گیر میکند. چرا if نگذاشتید؟