دانشگاه صنعتی اصفهان

دانشکده مهندسی برق و کامپیوتر

تکلیف اول درس سیستم عامل

نیمسال تحصیلی پاییز-۱۴۰۰

مدرّس: دکتر محمّدرضا حیدرپور

دستیاران آموزشی:

مجید فرهادی - دانیال مهرآیین - محمّد نعیمی

نحوه تحویل:
پاسخ های خود به همراه برنامه های نوشته شده را در قالب یک فایل PDF در سامانه بارگذاری کنید. استفاده از LaTeX اختیاری بوده و ۴۰ نمره اضافی دربرخواهد داشت. می توانید برای آشنایی با دستورات LaTeX از این قالب آماده شده استفاده کنید.

۱. به سوالات زیر به صورت کوتاه پاسخ دهید. (۸۰ نمره)

(آ) تفاوت User Context و Kernel Context چیست؟

(ب) در سیستم عامل، چرا به جای User Stack از Kernel Stack برای رسیدگی به وقفه ها استفاده می شود؟

(ج) در روند اجرای یک System Call، از Trap-Table و Syscall-Table چه استفاده ای می شود؟

(د) در هنگام Context Swtich، به چه علت ذخیره برخی از رجیسترها به عهده سخت افزار گذاشته می شود؟

(ه) با توجه به الگوریتم های زمان بندی FIFO، PSJF، Lottery، CFS و MLFQ اولیه، در کدام یک از این الگوریتم ها امکان Starvation وجود دارد؟

(و) در الگوریتم MLFQ اولیه برای جلوگیری از Gaming چه کار می توان کرد؟

(ز) در الگوریتم RR، بزرگ یا کوچک کردن Time Slice چه مزایا و معایبی می تواند داشته باشد؟

---

۲. با توجه به کد زیر، با احتساب والد در نهایت چند پروسس بر حسب n خواهیم داشت؟ دلیل خود را بیان کنید. (فرض کنید n توانی از ۲ باشد.) (۲۰ نمره)

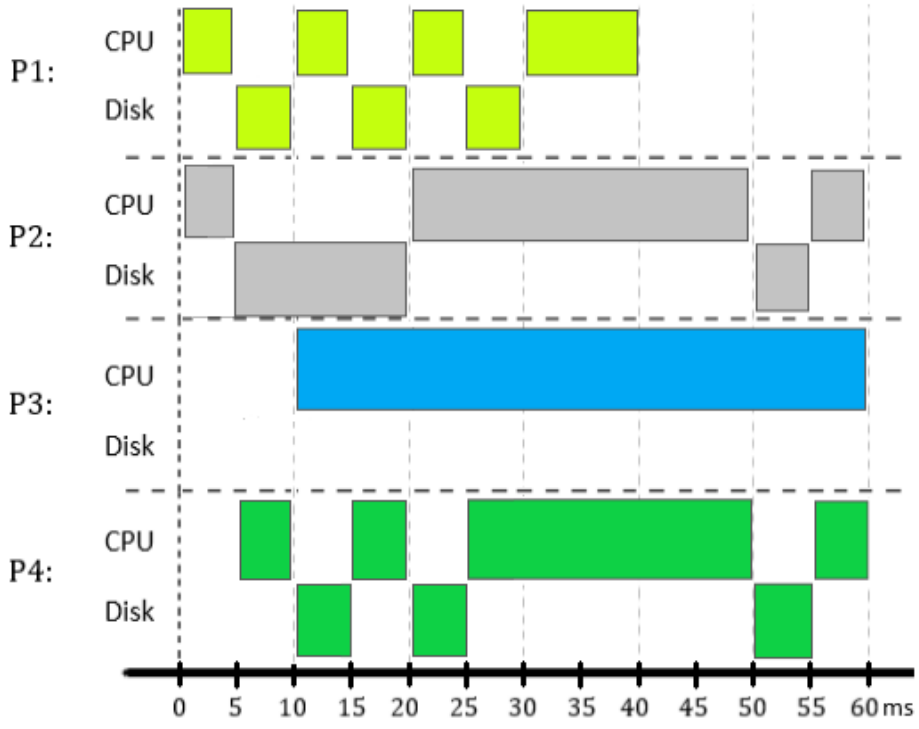
```
for (int i = 1; i <= n; i *= 2)
{
        fork();
}
```

۳. با توجه به شکل-۱ و الگوریتم های زمانبندی MLFQ و CFS به سوالات زیر پاسخ دهید. (۸۰ نمره)

(آ) اگر از الگوریتم MLFQ برای زمانبندی استفاده شود، ترتیب اجرای پروسس ها را ترسیم کنید. Time Slice مربوط به الگوریتم برابر با 10ms است و از ۴ صف اولویت در این الگوریتم استفاده می شود و مدت زمان Priority Boost مقداری بزرگ بوده که در این مسئله تاثیرگذار نمی باشد. همچنین در این الگوریتم از قانون Anti-Gaming استفاده نمی شود. (نکته: به پروسس های هر یک از صف های اولویت الگوریتم MLFQ، با عملکردی مانند الگوریتم RR سرویس دهی می شود؛ یعنی پروسس اول در ابتدای یک صف قرار گرفته و پروسس های بعدی به انتهای صف اضافه می شوند و به این پروسس ها به صورت چرخشی سرویس داده می شود. همچنین در لحظه صفر، ابتدا پروسس اول انتخاب شده و سپس به پروسس دوم رسیدگی می شود.)

(ب) در صورتی که برای این پروسس ها توسط زمانبند CFS اولیه (بدون استفاده از قابلیت های Adaptive Quantum و جریمه مربوط به پروسس های Sleep) با Quantum زمانی 10ms برنامه ریزی شوند، ترتیب اجرای این پروسس ها را با جزئیات مربوط به Virtual Runtime آن ها ترسیم کنید.

برای جواب به این سوال می توانید از دو شکل قرار داده شده در فایل تکلیف استفاده نمایید.



شکل ۱: نمودار زمانی مربوط به روند اجرای پروسس ها

۴. یک هکر قصد دارد تا با اجرای یک برنامه، کنترل CPU را از سیستم عامل گرفته و بدون هیچ محدودیتی از آن استفاده کند. برای این کار یک روش پیشنهاد دهید و روش خود را با استفاده از دستورات اسمبلی معماری x86 در یک برنامه به زبان C پیاده سازی کنید. با اجرای این برنامه، چه چیزی مشاهده می شود؟ آن را توجیه کنید. (۲۰ نمره)

---

۵. الگوریتم زیر برای هر عدد مثبت n به صورت زیر عمل می کند: (۵۰ نمره)

$$n = \begin{cases} n/2 & if\ n\ is\ even \\ n \times 3 + 1 & if\ n\ is\ odd \end{cases}$$

با اجرای مکرر این الگوریتم، به ازای هر عدد طبیعی و مثبت به عدد ۱ خواهیم رسید. برای مثال برای ورودی n=35 دنباله ای به صورت زیر ایجاد می شود:

35,106,53,160,80,40,20,10,5,16,8,4,2,1

با توجه به توضیحات داده شده، برنامه ای بنویسید که ورودی n را دریافت کند و با استفاده از دستور fork() دنباله مربوط به ورودی n را ایجاد کند. لازم است که عملیات به روز رسانی در فرزند و چاپ دنباله در والد انجام شود. (در حل این سوال از مقدار برگشتی فرزند به والد استفاده کنید. همچنین فرض کنید که اعداد دنباله از ۲۵۵ بیشتر نخواهند شد.)

(اختیاری): چرا نیاز است تا اعداد دنباله بیشتر از ۲۵۵ نشوند؟ در این مورد تحقیق کنید. (۲۰ نمره)

(اختیاری): این محدودیت را با استفاده از Shared Memory برطرف نمایید. (۳۰ نمره)

۶. با توجه به Orphan Process ها و Zombie Process ها، به سوالات زیر پاسخ دهید.

(آ) در مورد این پروسس ها و نحوه شناسایی آن ها تحقیق کنید و به صورت مختصر در مورد آن ها توضیح دهید.

(ب) هر یک از برنامه های ۱ و ۲، کدام یک از پروسس های تعریف شده در بالا را ایجاد می کنند؟ دلیل خود را بیان کنید و سپس ادعای خود را با استفاده از دستورات "ps -el" و "grep" مورد بررسی قرار دهید و نتیجه آن را در پاسخ نامه خود قرار دهید.

برنامه ۱:

```
#include<stdio.h>
#include <unistd.h>
int main()
{
        int rc = fork();
        if (rc == 0)
        {
                sleep(1);
                printf("Child: pid = %d, ppid = %d\n",
                        getpid(), getppid());
                sleep(2);
                printf("Child: pid = %d, ppid = %d\n",
                        getpid(), getppid());
                sleep(100);
        }
        else
        {
                printf("Parent: pid = %d, ppid = %d\n",
                        getpid(), getppid());
                sleep(2);
        }
        return 0;
}
```

برنامه ۲:

```
#include<stdio.h>
#include <unistd.h>
int main()
{
        int rc = fork();
        if (rc == 0)
        {
                sleep(1);
                printf("Child: pid = %d, ppid = %d\n",
                        getpid(), getppid());
        }
        else
        {
                printf("Parent: pid = %d, ppid = %d\n",
                        getpid(), getppid());
                sleep(100);
        }
        return 0;
}
```

---

۷. در این سوال قصد داریم تا روند اجرای پروسس های مختلف و حالات مختلف آن ها را بررسی کنیم. ابتدا فایل اجرایی مربوط به شبیه سازی را از اینجا دانلود کرده و سپس با توجه به توضیحات داده شده در این فایل، به سوالات ۶ و ۷ موجود در انتهای آن پاسخ دهید. همچنین می توانید برای آشنایی با طرز استفاده از این شبیه ساز به راهنمای شبیه ساز مراجعه کنید. تصویر مربوط به نتیجه شبیه سازی را به همراه تحلیل خود از نتیجه آن را در پاسخ نامه خود قرار دهید. (۳۰ نمره)

موفق باشید.