



پاییز 1401

### دستور کار جلسه هفتم

۱. هدف از این سوال ارسال ساختارمند اطلاعات از طریق حافظه‌ی مشترک در دو فرآیند است. برای این منظور مثال اول پیش گزارش بخش حافظه مشترک را در نظر بگیرید و کارهای زیر را انجام دهید. (زمان پیشنهادی: ۲۰ دقیقه)

- یک `strcut` در فایل `protocol` ایجاد کنید که شامل یک آرایه‌ی با اندازه‌ی `ARRAY_SIZE` و یک رشته به طول `STR_L` باشد.
- هدف نوشتن دو عنصر از این ساختار به صورت آرایه‌ای در حافظه‌ی مشترک و خواندن آن است.
- سعی کنید برنامه‌ها را به شکلی تغییر دهید که برنامه‌ی `sender` یک آرایه که شامل دو عنصر از ساختار تعریف شده است را ارسال نماید و برنامه‌ی `receiver` آن را دریافت و نمایش دهد.
- برای اعداد آرایه مقدار تصادفی بین ۱ تا ۵ تولید کنید و برای رشته‌ی عنصر اول نام خود و عنصر رشته‌ی دوم نام دوست خود را ارسال نمایید.
- (بررسی) در خط ۱۸ فایل `receiver` سعی کنید فلگ `PROT_WRITE` را اضافه نمایید. خطایی را که دریافت می‌کنید را در مقابل همان خط کامنت نمایید. چرا؟

۲. هدف از این سوال آشنایی بهتر با سیگنال‌ها است. باید برنامه‌ای بنویسید که علاوه بر چاپ پیغام به عنوان کار عادی خود کاری را برای آینده یعنی چند ثانیه بعد تنظیم نماید و اجرا کند. گاهی در برنامه‌ها لازم می‌شود کاری را تنظیم کنید که در آینده انجام شود. (زمان پیشنهادی: ۲۰ دقیقه)

- با تابع `alarm` از طریق `man` آشنا شوید.
- یک برنامه بنویسید که به صورت عادی هر ثانیه پیغام `I am alive!` را چاپ نماید. (میتوانید اینکار را با `sleep` انجام دهید. کار این قسمت شبیه سازی کار عادی برنامه است. در صورتی که این برنامه واقعی بود نیازی به `sleep` نبود).
- با استفاده از تابع `alarm` برنامه‌ای بنویسید که پیغام `I received alarm` را بعد از گذشت ۵ ثانیه از شروع برنامه، چاپ کند.

۳. هدف این سوال تقسیم کار بین چند فرآیند با استفاده از حافظه‌ی مشترک است. (زمان پیشنهادی: ۶۰ دقیقه)

میخواهیم برنامه‌ای بنویسیم که فرآیند‌های فرزند، عملکرد مشخصی با توجه به سیگنال دریافتی از سمت والد خود داشته را دریافت `SIGUSR۲` و اگر `۱ + ۲xi` را دریافت کرد، عنصر به `SIGUSR۱` باشند. به این صورت که اگر پردازش سیگنال تغییر کند `۳ + ۳xi` کرد، عنصر به در صورتی که واقعا عناصر آرایه زیاد باشند و تابع نگاشت شده نیز زمان‌بر باشد انجام این کار توسط چند فرآیند سریعتر از انجام سری توسط یک فرآیند است. البته روش بهتر در این مسائل (`cpu intensive`) روش `multi threading` است که در آینده خواهید دید.

الف. برنامه‌ی بنویسید که:

- برنامه‌ی شما قرار است با استفاده از فرزندان خود کار روی آرایه را انجام دهد.
- بنابراین لازم است یک آرایه به اندازه `ARRAY_SIZE` (۱۰۰ عنصر) در حافظه‌ی مشترک به صورت بی‌نام ایجاد شود.

### کار برنامه‌ی والد

- برنامه‌ی والد باید به تعداد `MAX_CHILD` فرزند ایجاد کند. هر فرزند `id` منحصر به خود را دارد.
- بعد از تعریف فرزندان والد باید آرایه را مقدار دهی اولیه کند. (`a[i] = i`)
- در گام بعد فرآیند والد با ارسال یکی از سیگنال‌ها به فرآیند‌های فرزند به آنها فرمان اجرا میدهد.

i. فرآیند والد به فرزندان با شماره id زوج SIGUSR۲ و به فرزندان با شماره id فرد، SIGUSR۱ را ارسال کند.

- والد باید منتظر پایان کار تمام فرزندان باشد و در نهایت آرایه را چاپ کرده و پایان یابد.

ii. برای انتظار فرزندان می‌توانید از waitpid یا wait استفاده کنید که پردازنده‌ها زامبی نشوند.

iii. برای چاپ عناصر آرایه از tab استفاده کنید که بتوان تمام آرایه را یکجا دید.

### کار برنامه‌های فرزند

- فرزند باید به صورت طبیعی در یک حلقه ( تا زمان دریافت سیگنال یا بینهایت) منتظر سیگنال بماند.

(pause)

- در صورتی که فرزند یکی از سیگنال‌ها را دریافت کرد باید وظیفه خود را انجام دهد. یعنی تعدادی از عناصر آرایه را تغییر داده و در همان جا ذخیره نماید.

- (پیشنهاد و راهنمایی) شما باید در اینجا یک روش تقسیم کار منصفانه ارائه دهید. یعنی بین همه‌ی پردازنده‌ها تقریباً یکسان کار تقسیم شود. همچنین در مواردی که تعداد عناصر آرایه (کارها) بر تعداد فرزندان بخش‌پذیر نبود باز برنامه باید صحیح کار کند. به همین دلیل پیشنهاد می‌شود هر فرزند به این

شکل عمل نماید(فرض کنید ۵ MAX\_CHILD است):

فرزند صفر: اندیس‌های ۰ و ۵ و ۱۰ و ۱۵ و ....

فرزند یک: اندیس‌های ۱ و ۶ و ۱۱ و ۱۶ و ....

فرزند دوم: اندیس‌های ۲ و ۷ و ۱۲ و ۱۷ و ...

فرزند سوم: اندیس‌های ۳ و ۸ و ۱۳ و ۱۸ و ...

فرزند چهارم: اندیس‌های ۴ و ۹ و ۱۴ و ۱۹ و ....

به همین دلیل می‌توانید از حلقه‌ی زیر استفاده کنید:

```
for( int i = child_id; i < ARRAY_SIZE; i += MAX_CHILD )
```

- فرزند بعد از اتمام کار خود باید پایان یابد.

**نکته مهم:** در این سوال چون کار روی آرایه ساده است و تعداد عناصر آرایه نیز پایین است، ممکن است قبل از آماده شدن پردازشی فرزند

سیگنال توسط والد ارسال شود و برنامه به مشکل بخورد. به همین دلیل بعد از حلقه‌ی fork با تابع usleep در حد ۱۰۰ میلی ثانیه فرصت دهید تا پردازنده‌های فرزند درست ساخته شده و به کار طبیعی خود بپردازند.

**نکته مهم ۲:** (نمره اضافی) انجام کارهای در تابع پاسخ سیگنال (هاندلر) ریزه‌کاری‌های فراوانی دارد چرا که در سیستم‌عامل‌ها آن را به نحوی غیر از برنامه عادی می‌دانند. به همین دلیل سعی کنید کار را در بخش کار طبیعی فرزند انجام دهید و تنها با تنظیم یک متغیر عمومی<sup>۱</sup> در تابع واکنش به سیگنال، برنامه‌ی عادی فرزند را متوجه کنید تا کار آرایه‌ها را انجام دهد.

---

<sup>۱</sup> global