دستورکار جلسه هفتم

۱. هدف از این سوال ارسال ساختارمند اطلاعات از طریق حافظه‌ی مشترک در دو پردازه 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 به صورت آرایه‌ای دو عنصر از ساختار را ارسال کند 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 برنامه ای 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 تابع alarm پیغام I received alarm را ۵ ثانیه بعد از شروع برنامه چاپ نمایید.
- بدیهی است شمردن تعداد پیغام‌ها قسمت b و چاپ پیغام مجاز نیست چرا که در یک برنامه‌ی واقعی این امکان وجود ندارد.

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

میخواهیم تمام عناصر یک آرایه را با یک تابع نگاشت کنیم. یعنی اگر هر عنصر آرایه را x_i در نظر بگیریم به طور مثال می‌خواهیم آن عنصر به $2x_i+1$ تغییر کند. در صورتی که واقعا عناصر آرایه زیاد باشند و تابع نگاشت شده نیز زمان‌بر باشد انجام این‌کار توسط چند پردازه سریعتر از انجام سری توسط یک پردازه 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) را 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 بعد از مقداردهی اولیه با ارسال SIGUSR1 به تمام فرزندان اعلام می‌کند که کار خود را شروع کنند.
- والد باید منتظر پایان کار تمام فرزندان باشد و در نهایت آرایه را چاپ کرده و پایان یابد.
  i. برای انتظار فرزندان می‌توانید از waitpid یا wait استفاده کنید که پردازه‌ها زامبی نشوند.
  ii. برای چاپ عناصر ارایه 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 سیگنال SIGUSR1 را دریافت کرد باید بخشی از کار را انجام دهد. یعنی تعدادی از عناصر آرایه را با تابع 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 پردازه‌ها تقریبا 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 MAX_CHILD ۵ است):

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

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

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

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

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

  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 پردازه‌ی 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 پردازه‌های فرزند درست ساخته شده و به کار طبیعی خود بپردازند.

نکته مهم ۲: (نمره اضافی) انجام کارهای در تابع پاسخ سیگنال (هنلدر) 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 متغییر عمومی[1] در تابع واکنش به سیگنال، برنامه‌ی عادی فرزند را متوجه کنید تا کار آرایه‌ها را انجام دهد.

[1] global