دستور کار جلسه پنجم

۱. این تمرین به دنبال آن است که روش اجرای کارها توسط چند پردازه را آموخته شود. به همین دلیل لازم است گام به گام خواسته‌های مطرح شده انجام شوند تا در نهایت به خروجی مطلوب برسید :

گام اول: یک تابع task به ترتیب زیر بنویسید:

```
Void task( int id)
```

این تابع شبیه‌ساز کاری است که قرار است به صورت چند پردازه‌ای انجام گیرد. برنامه‌ی اصلی با اختصاص یک id به این تابع، مشخص می‌کند که کدام بخش از کار را هر پردازه باید انجام دهد. برای شبیه‌سازی این مورد کافی است کارهای زیر را انجام دهید:

- هدف برنامه استفاده از حداکثر ظرفیت پردازنده تحت شرایطی خاص است.
- یک عدد تصادفی در بازه‌ی ۱ تا ۵ ایجاد کنید و به آن مقدار sleep نمایید.
- پس از پایان پیام زیر را چاپ نمایید.

```
Task <id> has been done by child <PID> in <S> seconds
```

- برای ایجاد صحیح اعداد تصادفی نیازمند یک seed دهی مناسب هستیم. مقدار دهی صحیح seed به نحوی که هم از نظر زمانی هم از نظر چند پردازه‌ای اعداد تصادفی باشند امتیاز مثبت خواهد داشت. (معمولا seed براساس زمان داده می‌شود تا خاصیت تصادفی بودن را به همراه داشته باشد. اما در برنامه‌های چند پردازه‌ای به دلیل اجرای تقریبا همزمان پردازه‌های مختلف اعداد تصادفی یکسانی تولید می‌کنند.)
- بهتر از در انتهای تابع task از exit(0) استفاده نمایید تا خیالتان از این بابت که برنامه‌ی فرزند بعد از انجام taskادامه پیدا نمی‌کند راحت شود. همیشه بهتر است در انتهای کار فرزند از این تابع استفاده کنید تا از خطاهای احتمالی کد نویسی خود جلوگیری کنید.

گام دوم: برنامه‌ی اصلی را به شرایط زیر بنویسید:

- برنامه به تعداد MAXCHILD فرزند ایجاد می کند که در آن MAXCHILD عددی است بین 2 تا 10 که لازم است با یک ماکرو تعریف شود. (برای تست‌های خود تعداد را ۵ در نظر بگیرید.)
- هر فرزند باید تابع task را با Id متناظر فرزند اجرا نماید.
- Manual دستور waitpid را بخوانید و با optionهای WNOHANG و WUNTRACED آشنا شوید.(این دو فلگ در این سوال و سوال بعد کاربرد خواهند داشت.)
- والد باید منتظر پایان کار تمام فرزندان باشد و در نهایت با چاپ پیامی خاتمه یابد.

گام سوم: برنامه‌ی اصلی در گام دوم را به نحوی تغییر دهید که خواسته‌ی زیر را نیز انجام دهد.

- فرآیند والد هر 5 ثانیه یکبار وضعیت همه فرزندان خود را بررسی کرده و در صورتی که یکی از آنها پایان یافته باشد، فرزندی دیگر را با همان id جایگزین آن خواهد کرد.
- فرآیند والد تا زمانی ادامه می یابد که یکی از سیگنال های پایان دهنده (برای مثال SIGINT=CTRL+C) را دریافت کند.

(فرض کنید که در یک سیستم دائماً taskهایی برای اجرا درخواست می‌شوند (مانند یک وب سرور). این برنامه نمونه‌ای از برنامه‌ای است که برای انجام task های سیستم پروسس می‌سازد و جهت مدیریت استفاده از منابع سیستم حداکثر تعداد پروسس‌ها را ثابت نگه می‌دارد. بدین منظور به محض اتمام پروسسی آن را با پروسس جدید (task جدید) جایگزین می‌کند. با استفاده از دستور top میزان استفاده از

CPU را چک کنید. آیا از حداکثر ظرفیت CPU استفاده می‌شود؟ sleep حین انتظار CPU مصرف نمی‌کند. به جای sleep از یک حلقه خالی for به اندازه زمان تأخیر sleep استفاده کنید و زمان تأخیر هر پروسس فرزند (که نماینده میزان محاسبات آن یا میزان مصرف CPU توسط اوست) را بالا ببرید، همچنین تعداد ماکزیمم پروسس‌ها را نیز بالا ببرید تا به حالتی برسید که از حداکثر ظرفیت CPU استفاده شود.)

۲. هدف از این تمرین اجرای برنامه‌ها توسط برنامه‌ی دیگر و ثبت لاگ است. به همین دلیل لازم است دو برنامه در این تمرین به کار گرفته شود.

برنامه‌ی اول: برنامه‌ی ساده‌ای با نام app.cpp در اختیار شما است این برنامه با دریافت ورودی از خط فرمان (argv) با چاپ پیغامی به آن مقدار sleep کرده و خاتمه می‌یابد.

- برنامه را با نام app کامپایل نمایید.
- این برنامه یک شبیه‌ساز برای پردازه‌هایی است که می‌خواهید با یک پردازه‌ی دیگر اجرا نمایید.

برنامه‌ی اصلی: برنامه ای بنویسید که به ترتیب زیر عمل کند:

- هدف برنامه ثبت یک log از اجرای برنامه های مختلف است.
- برنامه به صورت یک حلقه‌ی بینهایت باید برنامه‌ی app را اجرا نماید و زمان آن را ثبت کند.
- برنامه در هر اجرا لازم است یک عدد تصادفی r بین ۱ تا ۵ ایجاد کند.
- سپس باید برنامه‌ی app را با ورودی r در فرزند خود اجرا نماید.
- فرآیند والد می‌بایست منتظر پایان اجرای برنامه‌ی فرزند خود باشد و زمان اجرای فرزند را اندازه گرفته و چاپ نماید.
- فرآیند والد تا زمانی ادامه می یابد که یکی از سیگنال های پایان دهنده (برای مثال SIGINT=CTRL+C) را دریافت کند.
- مثال:

```
-----------------------------------------------------------------------------------
Date                    Time        Execution Time(ms)      r
-----------------------------------------------------------------------------------
2022-01-01              10:30       267                     2
-----------------------------------------------------------------------------------
2022-01-01              10:32       1300                    1
```

نکته: پیشنهاد می‌شود در گام اول یک برنامه‌ی والد بنویسید که تنها برنامه‌ی app را یکبار اجرا کند. سپس در گام دوم زمان اجرای آن را چاپ نمایید و در گام آخر برنامه را کامل نمایید.

همچنین پیشنهاد می‌شود تابعی بنویسید که تولید فرزند و سپس کار فرزند یعنی اجرای برنامه‌ی app و کار والد که انتظار برای اجرای کار فرزند هست، را در آن انجام دهید.. سپس زمان اجرای تابع اصلی (که لزوما در برنامه‌ی والد هست) را اندازه بگیرید.