دستور کار جلسه چهارم

۱. این تمرین به دنبال آن است که روش اجرای کارها توسط چند فرایند(process) را آموزش دهد. 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 فرایندی انجام گیرد. برنامه‌ی اصلی با اختصاص یک id به این تابع، مشخص می‌کند که کدام بخش از کار را هر فرایند 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 .معمولا seed براساس زمان داده می‌شود تا خاصیت تصادفی بودن را به همراه داشته باشد. اما در برنامه‌های چند فرایندی به دلیل اجرای تقریبا همزمان فرایندهای مختلف اعداد تصادفی یکسانی تولید می‌کنند. در نتیجه رابطه ای بین زمان و pid (مثلا جمع زمان و pid) می تواند مناسب باشد.
- 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 با شمارنده‌ای به اندازه‌ی ۱۰۰۰۰ برابر زمان تأخیر استفاده کنید و دوباره وضعیت استفاده از CPU را بررسی کنید.
سوال امتیازی : زمان تأخیر هر پروسس فرزند (که نماینده‌ی 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 ساخت برنامه‌ای است که محتویات یک دایرکتوری را در دایرکتوری دیگر کپی می‌کند و ضمن انجام، مدت زمان کپی هر فایل را نشان می‌دهد.

در این برنامه فرآیند کپی یک فایل توسط اجرای برنامه‌ی cp و فرآیند محاسبه‌ی زمان کپی و ... توسط برنامه‌ی ما انجام می‌شود.

برنامه‌ی اصلی: این برنامه به ترتیب زیر عمل کند:

- این برنامه آدرس دایرکتوری مبدا و مقصد را به کمک argv دریافت می‌کند.
- برنامه ابتدا لیست فایل‌های دایرکتوری مبدا را بدست می‌آورد.
- سپس در یک حلقه، برای هر فایل یک پروسس فرزند ایجاد شده که در آن دستور cp برای آن فایل اجرا می‌شود.
- پروسس والد منتظر پایان فرزند خود می‌ماند و زمان اجرای فرزند خود را اندازه گرفته و چاپ می‌کند.
- در صورتی که انجام پروسس فرزند (مثلا به دلیل وجود نداشتن دایرکتوری مقصد) با موفقیت انجام نشد، والد باید به کمک status دریافتی، متوجه شده و خطای مناسبی نمایش دهد.

مثال:

```
-------------------------------------------------------------------------------
Date                Time        Execution Time(ms)    FileName
-------------------------------------------------------------------------------
2022-01-01          10:30       267                   file1.txt
-------------------------------------------------------------------------------
2022-01-01          10:32       1300                  file2.png
```

برای بررسی صحت عملکرد برنامه‌ی خود، با استفاده از دستور زیر سه فایل با حجم‌های 100MB، 10MB و 1MB با نام دلخواه در یک دایرکتوری جدید بسازید و سپس آن‌ها را در دایرکتوری دیگری کپی کنید.

```
fallocate -l 20MB fileName
```

خبر خوش : بخشی از کد این برنامه در فایل dirCP.c قرار داده شده‌است و شما صرفا باید کدهای بخش اندازه‌گیری زمان را در تابع main و فرآیند اجرای دستور cp و انتظار و دریافت status توسط والد را در تابع childWork اضافه کنید.