دستورکار جلسه ششم

۱. هدف از این سوال مدیریت یک برنامه‌ی چند پردازه‌ای با مدیریت خط لوله است. به همین منظور کار فرزندان با sleep شبیه‌سازی می‌شود. هر فرزند باید مقداری تصادفی ایجاد کند و به آن میزان sleep کرده و آن را برای والد در پایپ بنویسد. والد باید تمام مقادیر ارسالی از فرزندان را جمع کرده تا به مقداری ثابت برسد و در نهایت کار خودش و فرزندان را پایان دهد.

بنابراین برنامه ای بنویسید که در آن:

- برنامه‌ی شما ابتدا یک متغییر عمومی با نام total=0 تعریف خواهد کرد.
- در گام بعد باید یک پایپ بسازد. (بین والد و تمام فرزندان فقط یک پایپ وجود خواهد داشت.)
- والد به تعداد MAX_CHILD فرزند ایجاد می‌کند و فرزندان به تابع کار خود منتقل خواهند شد.(برای تست مقدار را ۵ در نظر بگیرید)
- پس از ساختن فرزندان والد باید مدام منتظر خواندن مقادیر از پایپ باشد.
- والد مقدار ارسال شده توسط هر فرزند را دریافت کرده و با total جمع می کند.

هر گاه مقدار total از MAXIMUM (برای مثال MAXIMUM=100) بیشتر شود، والد همه فرزندان را از بین می‌برد و خود نیز به پایان می رسد. (راهنمایی سؤال ۱: برای از بین بردن فرزندان از kill(0, SIGKILL) استفاده کنید)

وظیفه‌ی فرزندان:

- هر فرزند در یک حلقه اجرا بینهایت اجرا می‌شود.
- فرزند در هر بار اجرا یک مقدار تصادفی بین ۱ تا ۵ ایجاد خواهد کرد و سپس به آن میزان sleep می‌کند.
- سپس فرزند عدد تصادفی ایجاد شده در گام قبل را روی پایپ می‌نویسد و به ابتدای حلقه برخواهد گشت.

نکات تحویل: هنگام تحویل به سوالات زیر پاسخ دهید:

الف) فرض کنید تعداد فرزندان و حجم داده‌ی انتقالی زیاد باشد. استفاده از یک پایپ برای تمام فرزندان چه خطری ممکن است ایجاد کند؟(با مطالعه‌ی بخش هفت manual به این مهم پی خواهید برد)

برای حل این مشکل چه پیشنهادی دارید؟ به صورت شفاهی توضیح دهید برای حل مشکل چگونه کد خود را باید تغییر دهید.

ب) آیا امکان ارتباط بین فرزندان توسط پایپ هست؟ چگونه؟

۲. هدف از این سوال استفاده از named pipeها است.

یکی از موارد استفاده از named pipeها ارتباط بین چند برنامه‌ی مجزا است. به طور مثال فرض کنید یک برنامه برای انجام کاری سنگین مقادیر را به برنامه‌ی دیگری که قوی‌تر است می‌دهد تا خروجی را دریافت کند. در این سوال قرار است این مورد شبیه‌سازی شود. به همین دلیل لازم است دو برنامه بنویسید.

برنامه‌ی server:

مسیر خط لوله را در ماکروی FIFO_PATH ذخیره کنید.

برنامه ابتدا باید خط لوله(fifo) را بسازد.

برنامه در یک حلقه‌ی بینهایت باید منتظر ارسال مقادیر از برنامه‌ی کلاینت باشد.

در ابتدای حلقه لازم است برنامه خط لوله را به صورت خواندن باز کند و منتظر خواندن دستور از پایپ باشد. دستور یا m است یا p به معنای ضرب کردن یا جمع کردن. بنابراین سرور قرار است با دریافت دو عدد و دستور آن‌ها در یک دیگر ضرب کرده یا اولی را به توان دومی برساند.

در گام بعد برنامه باید یک عدد به عنوان عملوند اول و عدد دوم را به عنوان عملوند دوم دریافت کند و پایپ را ببیند.

پس از انجام عملیات لازم و بدست آوردن نتیجه لازم است برنامه یکبار دیگر پایپ را در حالت نوشتن باز کند و پاسخ را در آن بنویسد و در نهایت پایپ را ببندد.

برنامه‌ی کلاینت:

برنامه کلاینت که بعد از برنامه‌ی سرور اجرا خواهد شد در یک حلقه‌ی بینهایت اجرا شود.

در ابتدا با دریافت مقادیر و دستور از کاربر خط لوله را در حالت فقط نوشتن باز کرده و ابتدا دستور مربوط، سپس مقدار اول و در مرحله‌ی سوم مقدار دوم را در پایپ می‌نویسد. سپس پایپ را بسته و دوباره در حالت خواندن باز می‌کند و پاسخ را به صورت یک عدد(long) دریافت می‌کند و به کاربر نمایش می‌دهد.

نکته: همانطور که مشخص است ارسال دستور و دو مقدار با سه نوشتن و خواندن محقق می‌شود. در صورتی که این مورد را با یک struct پوشش دهید نمره اضافه دریافت خواهید کرد. یعنی ساختار شما شامل یک کاراکتر به عنوان دستور و دو عدد خواهد بود.