آزمایش ۴ - ایجاد و اجرای پردازه‌ها

۴.۱ مقدمه

در این جلسه از آزمایش خواهیم آموخت که چگونه در سیستم عامل لینوکس می‌توان پردازه‌ی جدید ایجاد و اجرا نمود.

۴.۲ پیش‌نیازها

انتظار می‌رود که دانشجویان با موارد زیر از پیش آشنا باشند:

- برنامه نویسی به زبان C/C++
- دستورات پوسته لینوکس که در جلسات قبل فراگرفته شدند

۴.۳ پردازه چیست؟

به عنوان یک تعریف غیر رسمی، پردازه را می‌توان یک برنامه ی در حال اجرا دانست. ممکن است پردازه متعلق به سیستم باشد (مثلا login) یا توسط کاربر اجرا شده باشد (مثلا `ls`یا `vim`).
هنگامی که در سیستم عامل لینوکس یک پردازه ی جدید ایجاد می‌شود، سیستم عامل یک عدد یکتا به آن پردازه می‌دهد. این عدد یکتا را Process ID یا **PID** می‌نامند. برای دریافت لیست پردازه‌ها به همراه **PID**ی آن‌ها از دستور `ps` استفاده می‌شود.
نکته‌ی مهمی که باید در مورد پردازه‌ها بدانید آن است که پردازه‌ها در سیستم عامل لینوکس به عنوان واحدهای اولیه ی اختصاص منابع به شمار می‌روند. هر پردازه فضای آدرس خاص خود و یک یا چند ریسه در کنترل خود دارد. هر پردازه، یک «برنامه» را اجرا می‌کند. چند پردازه می‌توانند یک برنامه یکسان را اجرا کنند ولی هرکدام از پردازه‌ها یک کپی جداگانه از آن برنامه را درفضای آدرس خود و مستقل از پردازه‌های دیگر اجرا می‌کنند.

پردازه‌ها در یک ساختار سلسله مراتبی قرار می‌گیرند. به جز پردازه ی `init` هر پردازه یک والد دارد. هر پردازه می‌تواند با ایجاد پردازه‌های جدید، پردازه‌های فرزند به وجود بیاورد. ممکن است والد یک پردازه، لزوما ایجاد کننده‌ی آن نباشد. چرا که پس از قطع شدن اجرای پردازه والد اصلی (برای مثال در صورت پایان یافتن آن) والد جدیدی برای پردازه‌های فرزند در حال اجرا در نظر گرفته می‌شود.

۴.۴ شرح آزمایش

۴.۴.۱ مشاهده‌ی پردازه‌های سیستم و PID آن‌ها

۱. به کمک دستور `ps` لیست پردازه‌ها و **PID** آن‌ها را مشاهده می‌کنید.
۲. چه پردازه‌ای دارای **PID** برابر با یک است؟ به کمک دستور `man [process-name]` اطلاعاتی در مورد آن کسب کرده و به طور خلاصه وظیفه ی این پردازه و نحوه ی ساخته شدن آن را شرح دهید.
۳. به کمک تابع `getpid` برنامه‌ای بنویسید که **PID**ی خود را در خروجی چاپ کند.

۴.۴.۲ ایجاد یک پردازه ی جدید

تنها راه ایجاد یک پردازه‌ی جدید در سیستم عامل لینوکس، تکثیر کردن یک پردازه ی موجود در سیستم است. همان‌طور که در بخش قبل دیدید، ابتدا تنها یک پردازه‌ی init در سیستم وجود دارد و در واقع این پردازه جد تمام پردازه‌های دیگر سیستم است.

هنگامی که یک پردازه تکثیر می‌شود، پردازه‌ی فرزند و والد دقیقا مانند هم خواهند بود؛ به غیر از اینکه مقدار PID آن‌ها با هم متفاوت است. کد، داده‌ها و پشته‌ی فرزند، دقیقا از روی والد کپی می‌شود و حتی فرزند از همان نقطه‌ای که والد در حال اجرا بود، اجرای خود را ادامه می‌دهد. با این وجود، پردازه‌ی فرزند می‌تواند کد خود را با کد یک برنامه ی اجرایی دیگر جایگزین نماید و به این صورت برنامه‌ای غیر از والد خود را اجرا نماید.

۱. به کمک تابع `getppid` برنامه‌ای بنویسید که **PID**ی پردازه ی والد خود را چاپ کند. برنامه ی نوشته شده را در ترمینال اجرا کنید؛ پردازه‌ی والد چه پردازه‌ای است؟ نام آن را همراه با توضیح کوتاهی بیان کنید.
۲. برای تکثیر پردازه از تابع `fork` استفاده می‌شود. کد زیر به زبان **C** نوشته شده است. خروجی آن را مشاهده کنید. در مورد اینکه این کد چه کاری انجام می‌دهد توضیح دهید.

```c
#include <stdio.h>
#include <sys/wait.h>
#include <unistd.h>
int main() {
        int ret = fork();
        if (ret == 0) {
                // ...
                return 23;
        } else {
                int rc = 0;
                wait(&rc);
                printf("return code is %d\n", WEXITSTATUS(rc));
        }
        return 0;
}
```

۳. برنامه ی بالا را به گونه‌ای تغییر دهید که نشان دهد حافظه‌ی والد و فرزند از هم مستقل هستند.
٤. برنامه ی قسمت (۲) را به گونه‌ای تغییر دهید که برای والد و فرزند هر کدام پیام‌های جداگانه‌ای نمایش دهد؛ برای مثال برای فرزند عبارت I am the child و برای والد I am the parent را در خروجی چاپ کند (راهنمایی: در مورد مقدار بازگشتی تابع fork در صفحه ی `man fork` مطالعه کنید).
٥. به برنامه‌ی قسمت (۲) دو تابع `fork` دیگر اضافه کنید و بین هر کدام از fork ها یک خروجی (مثلا After first fork) چاپ کنید و نتیجه را ملاحظه کنید. کد خود را به همراه توضیح خروجی در گزارش بیاورید.

۴.۴.۳ اتمام کار پردازه‌ها

گاهی اوقات نیاز است که پردازه‌ی والد تا پایان اجرای پردازه‌ی فرزند منتظر بماند و سپس کار خود را ادامه دهد. برای این کار تابع `wait` مورد استفاده قرار می‌گیرد. جزئیات این تابع را می توانید با دستور `man 2 wait` ملاحظه کنید. همچنین تابع `exit` برای خاتمه‌ی اجرای برنامه کاربرد دارد.

۱. برنامه‌ای بنویسید که پردازه‌ی فرزند را ایجاد کند که این پردازه‌ی فرزند اعداد ۱ تا ۱۰۰ را در خروجی چاپ کند. بعد از پایان کار فرزند، پردازه‌ی والد باید با چاپ پیام پایان کار فرزند را اعلام کند. برای این کار از تابع `wait(NULL)` استفاده کنید (پارامتر اول چیست که مقدار `NULL`برای آن داده شده است؟)

۲. در صورتی که پیش از پایان کار فرزند، والد به اتمام برسد، والد پردازه‌ی فرزند به init تغییر پیدا می‌کند (اصطلاحا گفته می‌شود که پردازه‌ی فرزند توسط آن adopt می‌شود). به کمک استفاده از دستور sleep در فرزند برنامه‌ای بنویسید که این اتفاق را نشان دهد؛ یعنی، PID والد را قبل و بعد از اتمام والد در خروجی به همراه پیامی جهت پایان اجرای والد چاپ کند(راهنمایی: از sleep در بدنه‌ی پردازه‌ی فرزند استفاده کنید).

۴.۴.۴ اجرای فایل

برای اینکه پردازه‌ی فرزند برنامه‌ی دیگری غیر از والد را اجرا کند از دستورات execv, execl, execvp, execlp استفاده می‌شود.

۱. تفاوت‌های این دستورات را بیان کنید.

۲. برنامه‌ای بنویسید که یک پردازه‌ی فرزند ایجاد کند که این پردازه‌ی فرزند دستور `ls -g -h` را اجرا کند. (راهنمایی: آرگومان اول که همان دستور اجرا کننده‌ی برنامه است را نیز باید در لیست آرگومان‌ها قرار دهید.)