آزمایش ۴

ایجاد و اجرای پردازه‌ها

۱.۴ مقدمه

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

۲.۴ 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

۳.۴ پردازه چیست؟

به عنوان یک تعریف غیر رسمی، پردازه را می‌توان یک تک برنامه در حال اجرا دانست. ممکن است پردازه متعلق به سیستم باشد (مثلاً login) یا توسط کاربر اجرا شده باشد (مثلاً ls یا vim).

هنگامی که در سیستم‌عامل لینوکس یک پردازه ایجاد می‌شود، سیستم‌عامل یک عدد یکتا به آن پردازه می‌دهد. این عدد یکتا را Process ID یا PID می‌نامند. برای دریافت لیست پردازه‌ها به همراه PID آن‌ها از دستور ps استفاده می‌شود.

نکته‌ی مهمی که باید در مورد پردازه‌ها بدانید آن است که پردازه‌ها در سیستم‌عامل لینوکس به عنوان واحدهای اولیه‌ی 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 در فضای آدرس خود و مستقل از پردازه‌های دیگر اجرا می‌کنند.

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

۴.۴ شرح آزمایش

۱.۴.۴ 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 ۱ است؟ به کمک دستور man [process_name] اطلاعاتی در مورد آن کسب کرده و به طور خلاصه وظیفه‌ی این پردازه و نحوه‌ی ساخته‌شدن آن را شرح دهید.

۳. به کمک تابع getpid برنامه‌ای بنویسید که PID خود را در خروجی چاپ کند.

۲.۴.۴ ایجاد یک پردازه‌ی جدید

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

هنگامی که یک پردازه تکثیر می‌شود، پردازه‌ی فرزند و والد دقیقاً مانند هم خواهند بود؛ به غیر از اینکه مقدار PID آن‌ها با هم متفاوت است. کد، داده‌ها و پشته‌ی فرزند، دقیقاً 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 یک کد یک برنامه‌ی اجرای دیگر جایگزین نماید و به این صورت برنامه‌ای غیر از والد خود را اجرا نماید.

۱. به کمک تابع getppid برنامه‌ای بنویسید که PID پردازه‌ی والد خود را چاپ کند. برنامه‌ی 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 <unistd.h>
#include <sys/syscall.h>
int main () {
  long result;
  result = syscall(__NR_mkdir, "testdir", 0777);
  printf("The result is %ld.\n", result);
  return 0;
}
```

۳. برنامه بالا را به گونه‌ای تغییر دهید که نشان دهد حافظه‌ی والد و فرزند از هم مستقل هستند.

۴. برنامه قسمت (۲) را به گونه‌ای تغییر دهید که برای والد و فرزند هرکدام پیام‌های جداگانه‌ای نمایش دهد؛ برای مثال برای فرزند I am the child و برای والد I am the parent را در خروجی چاپ کند (راهنمایی: از خروجی تابع fork استفاده کنید).

۵. به برنامه‌ی قسمت (۲) دو تابع fork دیگر نیز اضافه کنید و بین هرکدام از هاfork یک خروجی (مثلاً After first fork) چاپ کنید و نتیجه را ملاحظه کنید. کد خود را به همراه توضیح خروجی در گزارش بیاورید.

۳.۴.۴ 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 تابه را می‌توانید با دستور man wait ملاحظه کنید. همچنین تابع exit برای خاتمه‌ی اجرای برنامه کاربرد دارد.

۱. برنامه‌ای بنویسید که پردازه‌ی فرزندی 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 پیامی پایان کار فرزند را اعلام کند. برای این کار از تابع wait(NULL) استفاده کنید.

۲. در صورتی که پیش از پایان کار فرزند، والد به اتمام برسد، والد پردازه‌ی فرزند به init تغییر پیدا می‌کند (اصطلاحاً 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 و 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 صفرم یا دستور اجرا کننده‌ی برنامه را نیز باید در لیست آرگومان‌ها قرار بدهید).

۵.۴ فعالیت‌ها

- در مورد گروه‌های پردازه‌ای و دستورات getpgrp و setpgid تحقیق کنید و توضیح مختصری در مورد آن‌ها ارائه دهید.
- برنامه‌ی ساده‌ی زیر را در نظر بگیرید. درخت پردازه‌هایی که این برنامه ایجاد می‌کند را رسم نمایید و خروجی آن را نیز بیان کنید:

```c
#include <stdio.h>
#include <unistd.h>

int main() {
  fork();
  fork();
  printf("Parent Process ID is %d\n", getppid());
  return 0;
}
```

- برنامه‌ی زیر را چند بار اجرا کنید. این برنامه چه چیزی را در سیستم‌عامل نشان می‌دهد؟

```c
#include <stdio.h>
#include <unistd.h>

int main() {
  int i = 0, j = 0, pid, k, x;
  pid = fork();
  if(pid == 0) {
    for (i = 0; i < 20; i++) {
      for (k = 0; k < 10000; k++);
      printf("Child %d\n", i);
    }
  } else {
    for (j = 0; j < 20; j++) {
      for (x = 0; x < 10000; x++);
      printf("Parent %d\n", j);
    }
  }
}
```

- پردازه‌ی Zombie چیست؟