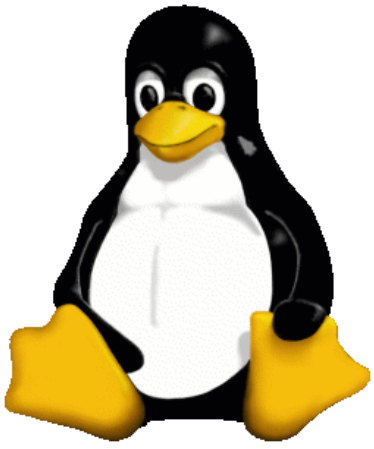
تهیه

دانشگاه صنعتی شریف
دانشکده مهندسی کامپیوتر

دستورکار آزمایشگاه سیستم‌های عامل

جلسه چهارم: ایجاد و اجرای پردازه‌ها



بهار ۱۳۹۳

مقدمه

در این جلسه از آزمایشگاه خواهیم آموخت که چگونه در سیستم‌عامل لینوکس می‌توان پردازه‌ها را ایجاد و اجرا نمود.

پیش‌نیازها

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

- برنامه‌نویسی به زبان C/C++
- دستورات پوسته‌ی لینوکس که در جلسات قبل فرا گرفته شده‌اند.

پردازه چیست؟

به عنوان یک تعریف غیررسمی، 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 به اختصار 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 برنامه‌ی یکسان را اجرا کنند ولی هر کدام از پردازه‌ها یک کپی جداگانه از آن برنامه را در فضای آدرس خود و مستقل از پردازه‌های دیگر اجرا می‌کنند.

پردازها در یک ساختار سلسله‌مراتبی قرار می‌گیرند. هر پردازه یک پردازه‌ی «والد» دارد که آن را ایجاد کرده است. پردازه‌هایی که یک پردازه ایجاد کرده‌است، پردازه‌های «فرزند» آن نامیده می‌شوند.

شرح آزمایش

الف) مشاهده‌ی پردازه‌های سیستم و PID آن‌ها

۱. به کمک دستور ps لیست پردازه‌ها و PID آن‌ها را مشاهده کنید.

۲. چه پردازه‌ای دارای PID برابر با 1 است؟ به کمک دستور man [process_name] اطلاعاتی در مورد آن کسب کرده و به طور خلاصه وظیفه‌ی این پردازه و نحوه‌ی ساخته شدن آن را شرح دهید.
۳. به کمک تابع getpid برنامه‌ای بنویسید که PID خود را در خروجی چاپ کند.

ب) ایجاد یک پردازه‌ی جدید

تنها راه ایجاد یک پردازه‌ی جدید در سیستم‌عامل لینوکس، تکثیر کردن یک پردازه‌ی 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 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 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 پردازه‌ی والد خود را چاپ کند. برنامه‌ی نوشته شده را در ترمینال اجرا کنید؛ پردازه‌ی والد چه پردازه‌ای است؟ نام آن را به همراه 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:

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

int main() {
      printf("Hello World!\n");
      fork();
      printf("I am after forking\n");
      printf("\tI am process %d.\n", getpid());
      return 0;
}
```

۳. برنامه بالا را به‌گونه‌ای تغییر دهید که نشان دهد حافظه‌ی والد و فرزند از هم مستقل هستند.
۴. برنامه‌ی قسمت (۲) را به گونه‌ای تغییر دهید که برای والد و فرزند هر کدام 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 هر کدام از fork ها یک خروجی (مثلاً After first fork) چاپ کنید و نتیجه را ملاحظه کنید. کد خود را به همراه توضیح خروجی در گزارش بیاورید.

پ) اتمام کار پردازه‌ها

گاهی اوقات نیاز است که پردازه‌ی والد تا پایان اجرای پردازه‌ی فرزند منتظر بماند و سپس به کار خود ادامه دهد. برای این‌کار تابع wait مورد استفاده قرار می‌گیرد. جزئیات این تابع 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 استفاده کنید.
۲. در صورتی که پیش از پایان کار فرزند، والد به اتمام برسد، والد پردازه‌ی فرزند به init تغییر 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).

ت) 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 نماید.

✓ 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 کنید و خروجی آن را نیز بیان کنید:

```
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 سیستم عامل نشان می‌دهد؟

```
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؟