جلسه‌ی ششم — مدیریت پردازه‌ها

در این جلسه با توابع و فراخوانی‌های سیستمی برای مدیریت پردازه‌ها در سیستم عامل‌های مشابه یونیکس آشنا خواهید شد. این بخش به سه بخش تقسیم شده است. بخش اول ساختن یک پـردازه‌ی جدیـد را شـرح می‌دهد، بخش دوم اجرای یک برنامه که در فایل سیستم قرار دارد را بیان می‌کند و بخش سوم شیوه‌ی انتظار در یک پردازه برای پردازه‌های فرزند آن را توصیف می‌نماید.

ایجاد یک پردازه

با فراخوانـی سیستمی «()fork»، سیستم عامـل پردازه‌ی جدیدی ایـجاد می‌کنـد که یـک کپـی از پردازنـده‌ی فراخوانی کننده می‌باشد. بنابراین پس از اجرای این فراخوانـی سیستمی دو پردازه‌ی پدر و فرزند هـر دو اجرای خود را با برگشتن از تابع «()fork» ادامه می‌دهند. بنابراین، در مثـال زیر عبـارت قبل از فراخوانی سیستمـی «()fork» یک بار و عبارت پس از آن دو بار (یک بار در پردازه‌ی پدر و یک بار در پردازه‌ی فرزند) اجرا می‌گردد (تابع «()getpid» شماره‌ی پردازه‌ی فراخوانی کننده را برگشت می‌دهد).

```
printf("%d: before forking\n", getpid());
fork();
printf("%d: after forking\n", getpid());
```

مقدار برگشت داده شده توسط «()fork» در پردازه‌ی پدر و فرزند متفاوت است و با استفاده از آن می‌توان به راحتی تشخیص داد که عبارت بعد، در کدام پردازه در حال اجرا است: این فراخوانی در پردازه‌ی پدر مقدار «PID» پردازه‌ی فرزند (مقداری بزرگ‌تر از صفر) و در پردازه‌ی فرزند صفر را بر می‌گرداند:

```
printf("Before fork syscall\n");
if (fork())                              /*          ایجاد یک پردازه‌ی جدید   */
      printf("The parent process\n");
else
      printf("The child process\n");
```

در صورتی که در اجرای این فراخوانی سیستمی مشکلی رخ دهد (مثلا به دلیل کمبود حافـظه، سیستم عامـل

نتواند پردازه‌ی جدیدی ایجاد نماید) این تابع مقدار منفی یک را بر می‌گرداند.

اجرای یک برنامه

برای اجرای یک برنامه که در فایل سیستم وجود دارد می‌توان یکی از توابع خانواده‌ی «exec()» را فراخوانی نمود. یکی از این توابع، تابع «execvp()» می‌باشد. ورودی اول این تابع، آدرس برنامه‌ی مورد نظر و ورودی دوم آن آرایه‌ای است که پارامترهایی که به پردازه‌ی ایجاد شده فرستاده می‌شوند (ورودی‌های فرستاده شده به تابع «main()» در یک برنامه) را مشخص می‌کند. این آرایه باید با یک «NULL» خاتمه پذیرد. به صورت قراردادی، در درایه‌ی صفرم این آرایه، آدرس برنامه تکرار می‌شود.

```
char *argv[] = {"ls", "/home", NULL};
execvp("ls", argv);                    /*        اجرای دستور «ls /home»   */
```

پس از این فراخوانی قسمت‌های کد و داده‌ی پردازه از بین می‌روند و با مقدار مناسب برای پردازه‌ی جدید جایگزین می‌گردند. بنابراین در صورت موفقیت‌آمیز بودن این فراخوانی هیچ یک از عبارت‌های پس از این فراخوانی اجرا نمی‌شوند. در صورت رخداد خطا (برای مثال، موجود نبودن برنامه‌ی مشخص شده) این فراخوانی مقدار منفی یک را بر می‌گرداند.

انتظار برای اتمام پردازه‌ها

فراخوانی سیستمی «wait()» منتظر خواهد بود تا یکی از پردازه‌های فرزند پردازه‌ی فراخوانی کننده خاتمه یابد. مقدار برگشت داده شده از این تابع، شماره‌ی «PID» پردازه‌ی خاتمه یافته است و اطلاعاتی در مورد خاتمه‌ی این پردازه (از جمله مقدار کد برگشتی آن) در متغیری که آدرس آن به این تابع فرستاده می‌شود قرار می‌گیرد. در مثال زیر، شیوه‌ی استفاده از «wait()» نمایش داده شده است.

```
pid = wait(&status);                /*        انتظار برای خاتمه‌ی یک پردازه‌ی فرزند   */

printf("pid %d exited with return code %d\n",
       pid, WEXITSTATUS(status));
```

همان طور که نشان داده شده است، با استفاده از ماکروی[1] «WEXITSTATUS» می‌توان کد برگشتی یک برنامه را از مقداری که این فراخوانی سیستمی در متغیر «status» قرار می‌دهد، استخراج نمود.

تمرین ششم

برنامه‌ای به نام «ex6.c» در شاخه‌ی «ex5» بنویسید که برنامه‌ای که داده می‌شود را اجرا کند، منتظر پایان این برنامه بماند و کد برگشتی آن را چاپ کند. برنامه‌ای که باید اجرا شود و پارامترهای آن به صورت پارامتر به برنامه‌ی «ex6» فرستاده می‌شوند. در مثال زیر برنامه‌ی «ls» با پارامتر اول «1.txt» اجرا می‌گردد:

```
$ ./ex6 ls 1.txt
```

در صورتی که کد برگشتی برنامه غیر صفر بود، اجرای برنامه باید پس از یک ثانیه تکرار شود. این کار باید تا زمانی ادامه پیدا کند که برنامه مقدار صفر را برگرداند. برای تأخیر، می‌توانید تابع «sleep()» را فراخوانی کنید.

گام‌های پیشنهادی برای انجام این تمرین:

۱ ایجاد فایل «ex6.c» و ترجمه‌ی آن

۲ ایجاد یک پردازه‌ی جدید با فراخوانی «fork()» و بررسی آن با چاپ پیغام‌هایی

۳ انتظار برای اتمام پردازه‌ی فرزند در پردازه‌ی پدر با فراخوانی «wait()» و چاپ کد برگشتی آن

۴ اجرای یک برنامه در پردازه‌ی فرزند با فراخوانی «exec()»

۵ تکرار ایجاد پردازه‌ی فرزند در پردازه‌ی پدر در صورت دریافت کد بازگشتی غیر صفر

1 Macro