

جلسه نهم — سیگنال‌ها

در این جلسه با سیگنال‌ها در یونیکس و شیوه‌ی دریافت و ایجاد آنها آشنا خواهید شد. همچنین، شیوه‌ی مدیریت دسترسی به فایل‌ها به صورت فاصله معرفی می‌گردد.

سیگنال‌ها

سیستم عامل می‌تواند با استفاده از سیگنال، پردازنده‌ها را از رخدادهای خارجی مطلع سازد. سیگنال‌ها برای اهداف مختلفی استفاده می‌شوند: گاهی برای اطلاع از گذشت زمان مشخص (مثل فرافوانی سیستمی `alarm()`، گاهی برای گزارش اشکال در اجرای پردازنده (مثل اشکال در دسترسی به حافظه)، گاهی برای ارتباط بین پردازنده‌ها و گاهی برای اطلاع پردازنده از درخواست‌های خارجی (مثل درخواست اتمام پردازنده).

سیستم عامل برای هر سیگنال (که با یک عدد مشخص می‌شود) به صورت پیش‌فرض عمل خاصی را در هر پردازنده انجام می‌دهد (این عملیات پیش‌فرض در صفحه‌ی راهنمای «`signal`» شرح داده شده‌اند). هر پردازنده می‌تواند عملی که باید بعد از رخداد هر سیگنال (با چند استثنا) انجام شود را تغییر دهد. یکی از راه‌های انجام این کار، استفاده از فرافوانی سیستمی `signal()` می‌باشد. فرافوانی سیستمی `signal()` دو ورودی دریافت می‌کند: ورودی اول شماره‌ی سیگنال و ورودی دوم عملی که باید پس از رخداد سیگنال انجام شود را مشخص می‌کنند.

```
signal(SIGINT, SIG_DFL);
signal(SIGINT, SIG_IGN);
signal(SIGINT, func);

void func(int signo)
{
    printf("Signal %d\n", signo);
}
```

همان طور که در این مثال مشاهده می‌شود، شماره‌ی سیگنال‌ها توسط ماکروهای¹ (که در فایل «signal.h» تعریف شده‌اند) مشخص می‌شود. عملی که باید پس از رخدادهای سیگنال انجام شود با چند ماکرو یا یک تابع مشخص می‌شود. ماکروی «» عمل پیش‌فرض را مشخص می‌کند و ماکروی «» به این مفهوم است که سیگنال باید نادیده گرفته شود. در صورتی که یک تابع به عنوان ورودی دوم به signal() داده شود، در صورت رخداد سیگنال تابع مشخص شده فراخوانی می‌گردد. در مثال قبل این سه حالت نمایش داده شده‌اند.

یکی از سیگنال‌هایی که بسیاری از برنامه‌ها رفتار آن را تغییر می‌دهند سیگنال است: در صورتی که کاربر (با استفاده از کلیدهای کنترل و C) درخواست فاتمی یک برنامه را داشته باشد این سیگنال به برنامه فرستاده می‌شود که به صورت پیش‌فرض موجب فاتمی آن می‌گردد. سیگنال پس از اتمام هر یک از فرزندان پردازش به آن ارسال می‌گردد. سیگنال‌های و برای تعامل بین پردازش‌ها استفاده می‌شود (یک پردازش بتواند پردازش دیگری را از اتفافی آگاه کند).

یک پردازش نیز می‌تواند از سیستم عامل درخواست کند تا سیگنالی به پردازش دیگری فرستاده شود. این کار توسط فراخوانی سیستمی kill() انجام می‌شود. این فراخوانی سیستمی دو ورودی دریافت می‌کند: ورودی اول شماره‌ی پردازش دریافت کننده‌ی سیگنال و ورودی دوم شماره‌ی سیگنال می‌باشد.

```
kill(pid, SIGUSR1);
```

مدیریت پردازش‌ها در پوسته

با استفاده از دستور «ps» می‌توان فهرست پردازش‌های در حال اجرا را مشاهده نمود. دستور «ps aux» فهرست همه‌ی پردازش‌های در حال اجرا، صاحب هر پردازش و شماره‌ی آن را نمایش می‌دهد. همچنین دستور «pstree» سافتار درختی پردازش‌ها را به صورت گرافیکی نمایش می‌دهد.

```
$ ps aux
$ pstree -phcU
```

برای فرستادن یک سیگنال از پوسته، می‌توان از دستور «kill» استفاده نمود. در دستور «kill»، می‌توان نام یا شماره‌ی سیگنال را مشخص نمود. نام سیگنال (پس از حذف SIG از شروع آن)

1 Macros

باید پس از «s» قرار گیرد (مثل «sTERM») و شماره‌ی سیگنال باید بعد از علامت «-» مشخص می‌شود (مثل «2-«).

```
$ kill -sUSR1 pid
```

مدیریت دسترسی‌ها

در سیستم‌های عامل چند کاربره^۱ لازم است مکانیزمی برای محافظت از فایل‌ها و به اشتراک گذاشتن آنها وجود داشته باشد. همان طور که در درس سیستم عامل اشاره می‌گردد، برای هر فایل (یا شافه) در یونیکس یک کاربر به عنوان صاحب آن مشخص می‌گردد. صاحب هر فایل می‌تواند گروه و میزان دسترسی افراد مختلف به آن فایل را مشخص نماید. برای دسترسی به فایل‌ها، افراد به سه دسته تقسیم می‌شوند: صاحب فایل، اعضای گروه فایل و سایر افراد. به ازای هر یک از این دسته‌ها، صاحب فایل می‌تواند مشخص کند که این افراد اجازه‌ی خواندن، نوشتن یا اجرای فایل را دارند یا فیر. معمولاً اجازه‌ی دسترسی‌ها به یک فایل را با یک عدد سه رقمی در مبنا‌ی هشت (مثل ۶۴۴) نمایش می‌دهند. هر رقم در این نمایش دسترسی یکی از این دسته‌ها را مشخص می‌کند. در «»، دسترسی صاحب فایل، دسترسی اعضای گروه فایل و دسترسی سایر افراد را مشخص می‌نماید. در هر یک از این رقم‌ها، بیت کم‌ارزش توانایی اجرا، بیت بعدی توانایی نوشتن و پر ارزش‌ترین بیت توانایی خواندن را نشان می‌دهد. برای مثال، در صورتی که دسترسی یک فایل «۶۴۰» باشد یعنی صاحب فایل می‌تواند از فایل بخواند و بر روی آن بنویسد، اعضای گروه فایل فقط می‌توانند آن را بخوانند و سایر افراد اجازه‌ی هیچ یک از این عملیات را ندارند. پارامتر -ا- دستور «ls» صاحب، گروه و دسترسی افراد را برای فایل‌ها نمایش می‌دهد.

```
$ ls -l
-rw-r--r-- 1 user users 143 Dec 4 14:10 Makefile
-rwxr-xr-x 1 user users 7609 Nov 9 23:57 ex8
-rw----- 1 user users 412 Nov 9 23:57 ex8.c
-rw-r--r-- 1 user users 2232 Nov 9 23:57 ex8.o
```

صاحب، گروه و دسترسی افراد را می‌توان با دستورات زیر تعیین نمود (راه‌های فاصله‌تری برای مشخص کردن اجازه‌ی دسترسی افراد برای دستور «chmod» وجود دارد؛ به صفحه‌ی راهنمای آن

1 Multi-user

مراجعة شود).

```
$ chown user path  
$ chgrp users path  
$ chmod 644 path
```

تمرین نهم

برای انجام این تمرین فایل `oslab9.c` را دریافت کنید. این برنامه یک ریسمان می‌سازد و در آن مقدار متغیر «found» را به روز می‌رساند. این برنامه را به صورتی تغییر دهید که پس از دریافت سیگنال «SIGUSR1»، آفرین مقدار متغیر «found» را چاپ کند و در صورت دریافت سیگنال «SIGINT»، پس از چاپ مقدار «found» خاتمه یابد.

گام‌های پیشنهادی برای انجام این تمرین:

- ۱ دریافت و ترجمه‌ی فایل «`oslab9.c`»
- ۲ نوشتن تابعی برای دریافت سیگنال «SIGUSR1» و فراخوانی «`signal()`» برای ثبت آن
- ۳ آزمایش درستی دریافت سیگنال «SIGUSR1» با استفاده از دستور «kill» از پوسته
- ۴ تکرار مرحله‌ی دو برای سیگنال «SIGINT» و فروج پس از دریافت آن
- ۵ آزمایش درستی دریافت سیگنال «SIGINT» با فشار دادن دکمه‌های کنترل و «C»