

جلسه هفتم — انتقال داده با لوله

در این جلسه با شیوهی مدیریت فایل‌های باز پردازنده‌ها در یونیکس و استفاده از لوله برای انتقال اطلاعات بین آنها آشنا خواهیم شد. در یونیکس علاوه بر فایل‌های ذخیره شده در دیسک، بسیاری از منابع موجود در سیستم عامل (از جمله اتصالات شبکه، لوله‌ها و بسیاری از قطعات سخت‌افزاری از جمله کارت‌های صوتی، دیسک‌ها و حافظه‌ی کارت‌های گرافیکی) نیز توسط فایل قابل دسترسی هستند.

شناسه‌های فایل

۱ در یونیکس هر فایل باز (Open file) پردازنده با یک عدد مشخص می‌شود؛ به این عدد شناسه‌ی فایل (File descriptor) گفته می‌شود. تابع `open()` یک فایل را در فایل سیستم باز می‌کند و به آن یک شناسه تخصیص می‌دهد. در ادامه کد زیر، مقدار شناسه‌ی ایجاد شده را چاپ کنید.

```
fd = open("test.txt", O_RDWR | O_CREAT, 0644);   ایجاد شناسه‌ی فایل  
close(fd);                                       بستن یک شناسه‌ی فایل
```

۲ تابع `write()` بایت‌های داده شده را (که توسط یک اشاره‌گر و تعداد بایت‌ها مشخص می‌شود) به یک فایل می‌نویسد. عدد برگردانده شده توسط تابع `write()` تعداد بایت‌های نوشته شده در شناسه‌ی فایل داده شده را مشخص می‌کند. در صورتی که فضای رخ دهد، عددی منفی از این تابع برگشت داده می‌شود.

```
nw = write(fd, "Hello\n", 6);           نوشتن یک رشته در یک شناسه‌ی فایل
printf("wrote %d bytes\n", nw);
```

۳ به صورت مشابه می‌توان با تابع `read()` از یک فایل خواند.

```
char buf[128];
nr = read(fd, buf, 128);
```

۴ به صورت قراردادی، فایل شماره‌ی صفر به ورودی استاندارد (`stdin`) در کتابخانه‌ی استاندارد زبان C، فایل شماره‌ی یک به خروجی استاندارد (`stdout`) و فایل شماره‌ی دو به خروجی خطا (`stderr`) اختصاص می‌یابند. رشته‌ای را به شناسه‌ی فایل یک بنویسید.

۵ برای راحتی بیشتر، با استفاده از تابع `fdopen()` می‌توان با کمک توابع کتابخانه‌ی استاندارد زبان C مثل `fprintf()` و `fscanf()` به یک شناسه‌ی فایل نوشت.

```
FILE *fp = fdopen(fd, "w");
fprintf(fp, "Hello\n");
fclose(fp);
```

۶ پارامتر دوم `fdopen()` نوع نوشتن به فایل را مشخص می‌کند: برای نمونه، «r» برای خواندن و «w» برای نوشتن.

۷ برنامه‌ای بنویسید که با استفاده از شناسه‌های فایل، یک خط از ورودی استاندارد بخواند و آن را در فایلی به اسم line.txt بنویسد.

استفاده از لوله

۸ با فرافوانی تابع pipe()، سیستم عامل یک لوله می‌سازد. هر لوله دو سر دارد؛ هر چه در سر نوشتن به لوله نوشته شود، می‌تواند از سر خواندن خوانده شود. این تابع یک آرایه با اندازی دو می‌گیرد و شناسه‌ی فایل دو سر لوله را در این آرایه قرار می‌دهد.

```
int fds[2];  
pipe(fds);  
printf("descriptors: %d, %d", fds[0], fds[1]);
```

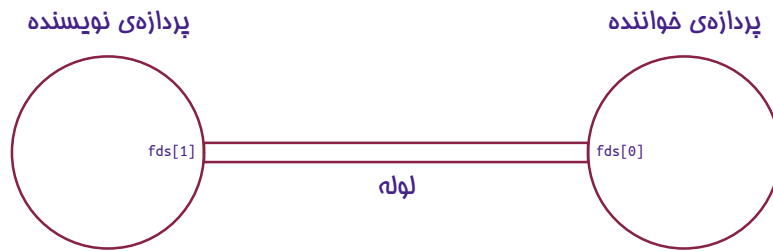
ایجاد لوله و ذخیره‌ی شناسه‌ی دو سر آن در fds[]

۹ با استفاده از یک لوله می‌توان داده‌هایی را بین دو پردازش انتقال داد. معمولاً پس از فرافوانی تابع pipe()، با تابع fork() پردازش جدیدی ساخته می‌شود. سپس یکی از این پردازش‌ها از سر نوشتن لوله داده‌ها را می‌نویسد و پردازش دیگر از سر خواندن لوله، داده‌ها را می‌خواند.

```
char buf[100];  
pipe(fds);  
if (fork()) {  
    close(fds[0]);  
    write(fds[1], "Hello\n", 6);  
} else {  
    close(fds[1]);  
    read(fds[0], buf, 100);  
}
```

ایجاد لوله
ایجاد پردازش جدید
بستن سر خواندن
نوشتن به لوله
بستن سر نوشتن
خواندن از لوله

۱۰ در شکل زیر ارتباط دو پردازش با استفاده از لوله نشان داده شده است.



۱۱ برنامه‌ای بنویسید که یک پردازش ایجاد کند. پردازش اصلی یک خط از ورودی استاندارد بخواند و آن را با لوله به پردازش فرزند بفرستد. پردازش فرزند یک خط از آن لوله بخواند و آن را چاپ کند.