درس سیستم‌های عامل (دکتر جلیلی)

پاییز ۱۳۸۶

تمرین کامپیوتری ۲

تاریخ تحویل : سه شنبه ۱۳۸۶/۹/۲۰

نکات قابل توجه

۱. تمرین‌های کامپیوتری باید در گروه‌های **دو نفره** انجام شود.

۲. برنامه‌ها باید در محیط لینوکس و با زبان C/C++ نوشته شوند.

۳. استفاده از Makefile برای ساخت برنامه‌ها الزامی است. اطلاعات مربوط به نوشتن Makefile در بخش منابع وب‌سایت درس موجود است.

۴. تمارین باید تا پایان وقت تعیین شده، به آدرس ce424-owner@lists.ce.sharif.edu فرستاده شوند.

- عنوان میل : CA2
- ذکر نام و نام خانوادگی و شماره دانشجویی در متن نامه الزامی است.
- متن برنامه به صورت zip شده و فرمت (family1_family2.zip) پیوست شود.

۵. تنها در صورت بروز مشکل در فرستادن تمارین به آدرس فوق، می‌توانید آن را به آدرس ce424s86@gmail.com بفرستید.

الف. اجرای دستورات خارجی (External Commands) در پوسته (Shell)

همانطور که می‌دانید، پوسته یک دسته دستورات داخلی و یک دسته دستورات خارجی دارد. دستورات داخلی آنهایی هستند که توسط خود پوسته پیاده‌سازی می‌شوند و دستورات خارجی آنهایی‌اند که موجب اجرای مستقیم برنامه‌های اجرایی دیگر می‌شوند. به پوسته‌ی نوشته شده در تمرین کامپیوتری شماره‌ی ۱، امکان اجرای دستورات خارجی را نیز اضافه کنید. پوسته بایستی امکان اجرای دستورات خارجی را در background هم داشته باشد. بدین معنا که وقتی پوسته در حال اجرای یک دستور است، بتواند بدون منتظر ماندن برای اتمام آن، دستور بعدی را نیز اجرا نماید.

Example of executing normally:

```
>> cmd  arg1  arg2  …  argn
>> cp  -r  /root/temp.txt    /home/SHMB/
```

Example of executing in background:

```
>> cmd  arg1  arg2  …  argn &
>> cp  -r  /root/temp.txt    /home/SHMB/ &
```

راهنمایی:

- نسخه‌ی سوم man سیستم عامل لینوکس مربوط به دستور exec (*man 3 exec*).
- فایل man مربوط به دستور fork.
- مطالعه‌ی فصل ۱۱ کتاب Linux-Programming-Unleashed، قسمت *System calls and library function*.

ب. آشنایی با IPC و Signaling

می‌خواهیم یک دستور داخلی به نام *history* به پوسته‌ی تمرین ۱ بیفزاییم که N دستور آخر اجرا شده توسط کاربر را به ما نشان می‌دهد. به دلایلی نمی‌خواهیم که این لیست دستورات در خود پردازه‌ی shell نگهداری شود. به جای آن یک برنامه به نام *logger* باید نوشته شود که این لیست را در خود نگه دارد و سرویس‌هایی برای بازیابی آنها ارائه دهد.

به این منظور، در برنامه‌ی پوسته، پس از هر دستوری که توسط کاربر اجرا می‌شود، به برنامه‌ی *logger* یک سیگنال (signal) بفرستید که آن را از وجود یک درخواست مطلع کند. سپس با استفاده از یکی از روش‌های IPC مثل *pipe, shared memory, message passing, etc.* (از *socket programming* استفاده نکنید)، نام دستور را به *logger* بفرستید. از طرف دیگر، برنامه‌ی *logger* منتظر یک سیگنال می‌ماند (*wait*) و هرگاه که یک سیگنال دریافت کرد با استفاده از IPC، نام دستور فرستاده شده را دریافت می‌کند و در یک ساختمان داده ذخیره می‌کند. برای سادگی این ساختار داده فرض کنید که تعداد دستورات *history* محدود است.

Example:

```
myshell >> ls
.           ..            temp1         temp2
myshell >> cd  temp1
myshell >> history
   error: there is no logger process running in the system.
myshell >> ls
.           ..             shell.cpp    shell.h      logger
myshell >> ./logger
myshell >> chertopert
   error: no such file or directory or command
myshell >> cd  ..
myshell >> ls
.           ..            temp1         temp2
myshell >> history 10
   chertopert
   cd
   ls
   history
myshell >> history 3
   ls
   history
   history
```

راهنمایی:

- چون خود پوسته، برنامه‌ی *logger* را اجرا می‌کند، بنابراین می‌تواند به راحتی PID آن را بدست آورد.
- فایل man مربوط به fifo و signal (*man fifo, man signal*).
- مطالعه‌ی قسمت‌های مربوط به IPC در کتاب Linux-Programming-Unleashed.