دانشکده مهندسی کامپیوتر

دکتر رضا انتظاری ملکی

بهار ۱۴۰۰

Operating System

تمرین عملی سری سوم

Inter Process Communication

تاریخ تحویل : ۳۱ اردیبهشت ۱۴۰۰ ساعت ۲۳:۵۹:۵۹

1- Pipe:

در جلسات قبل دستور زیر را در کلاس های حل تمرین دیده اید:

```
mojtaba@mojtaba-Lenovo-ideapad-320-15IKB:~$ ps -A | grep chrome
 101743 ?        00:19:22 chrome
 101882 ?        00:00:00 chrome
 101883 ?        00:00:00 chrome
 101885 ?        00:00:01 chrome
 101905 ?        00:21:55 chrome
 101911 ?        00:02:27 chrome
 101923 ?        00:00:02 chrome
 120176 ?        00:00:36 chrome
 120541 ?        00:00:00 chrome
 120563 ?        00:00:00 chrome
 127219 ?        00:00:10 chrome
 130023 ?        00:00:07 chrome
```

در این نوع از دستورات در ترمینال لینوکس، از معماری **Pipe and Filter** استفاده می شود.

به این صورت که دستور اول اجرا شده و سپس دستور دوم روی خروجی دستور اول، عملیاتی را انجام میدهد.

در این سوال ما از شما میخواهیم یک برنامه به زبان **C** بنویسید که با استفاده از **pipe** ها دستور زیر را اجرا کند.

ps -A -T | grep chrome

این دستور در واقع لیست **thread** های سیستم را گرفته و سپس در روی آن لیست با توجه به رشته ی **chrome** یک فیلتر اعمال میکند .و برای پیاده سازی باید دو **process** داشته باشید که اولی دستور یک را اجرا و خروجی را در **pipe** قرار دهد و **process** دوم باید روی خروجی دستور اول، دستور اول را اجرا کند.

نکات پیاده سازی:

۱- حتما **syscall** های بر پایه ی **exec** را مرور کنید.

۲- برای سادگی حل سوال میتوانید با استفاده از دستور **man** یا جستجو در اینترنت دستور **dup2** را مطالعه کنید.تا عملیات خواندن و نوشتن روی **pipe** برای شما ساده تر شود.(در غیر این صورت کارتان دشوار خواهد بود.)

use the dup2() system call to duplicate the reading file descriptor of the pipe (fd[0]) onto the standard input file descriptor.

use the dup2() system call to duplicate the writing file descriptor of the pipe (fd[1]) onto the standard output file descriptor.

در نهایت شما باید موارد زیر را تحویل دهید:

۱- کد

۲- یک فایل **pdf** توضیحات شامل:

الف) روند اجرای کد تا رسیدن به خروجی

ب) همچنین توضیحاتی در مورد **file description** ها و ارتباط دقیق آن ها با **pipe**

2- Shared Memory

قرار است دو برنامه داشته باشیم؛ یکی مربوط به نوشتن در **shared memory** و یکی برای خواندن از آن. برای پر کردن **shared memory**، از ساختار **Linked List** ای که در فعالیت کلاسی اول زدید استفاده کنید. (توجه: حتما کدی که راجع به **shared memory** در کلاس گفته‌شد را دوباره نگاه کنید!)

مرحله‌ی اول: یک **command line argument** تعریف کنید که با کمک آن کاربر بتواند سایز لیست مشترک مورد نظر بین دو پروسس را موقع وارد کردن دستور اجرای برنامه به آن پاس بدهد. این پارامتر را در یک ناحیه از مموری به صورت **shared** قرار دهید و در مرحله‌ی اول، کاری کنید که هر دو برنامه‌ی خواندن و نوشتن، این پارامتر را چک کنند و اگر این پارامتر از قبل در برنامه‌ی مقابل پر شده بود (یعنی کاربر موقع اجرای برنامه‌ی مقابل، پارامتر سایز لیست را در **shared memory** گذاشته بود)، به جای این که به ورودی کاربر توجه کند، از همان مقدار استفاده کند. اگر این مقدار از قبل در برنامه‌ی مقابل وارد نشده بود، ورودی کاربر را هم در برنامه‌ی خود لحاظ کند و هم در **shared memory** مربوط به پارامتر سایز لیست قرار بدهد.

مرحله‌ی دوم: فایل هدر ساختار **Linked List** ای که از قبل پیاده‌سازی کردید را طوری ویرایش کنید که هر خانه‌ی آن حاوی یک استرینگ به اسم **value** و یک عدد به اسم **key** باشد.

مرحله‌های مربوط به Write

مرحله‌ی سوم: تابعی بزنید که تمامی دایرکتوری‌های لینوکس شما را پیمایش کند و فقط دایرکتوری برگ‌های آن را ذخیره کند (یعنی تا آخرین عمق فولدر در هر دایرکتوری پیش برود و آدرسی که دیگر در داخل آن فولدری موجود نیست را نگه دارد.) (می‌توانید این مرحله را با **Bash Script** هم انجام دهید.)
فقط به اندازه سایز لیست مشترک ورودی که در مرحله یک گرفتید را در حافظه مشترک ذخیره کنید.

مرحله‌ی چهارم: آدرس‌هایی که ذخیره کرده‌اید را **encrypt** کنید به طوری که همه به استرینگ‌هایی تبدیل شوند که فقط حاوی اعداد و حروف انگلیسی هستند. **Key** ای که به وسیله‌ی آن آدرس‌ها را **encrypt** می‌کنید در مکانی از **shared memory** قرار دهید تا بعدا بتوان محتوا را **decrypt** کرد. (از کدهای آماده برای این کارها می‌توانید کمک بگیرید.)

مرحله‌ی پنجم: دایکتوری‌های **encrypt** شده را داخل پارامتر **value** ی ساختار **Linked List** ای که قبلا پیاده سازی کردید بریزید و برای پارامتر **key** هر خانه، عددی رندوم بگذارید. سپس **node** درست شده را در داخل **Linked List** ای که در **shared memory** هست قرار دهید. (برنامه‌ی **write**)

مرحله‌های مربوط به Read

مرحله‌ی ششم: مطمئن شوید تا الان برنامه‌ی **read** باید از سایز و **struct** مربوط به **node** هر لیست (که در مرحله‌های قبل باید انجام شده‌باشد) و همین‌طور **key** ای که با استفاده از آن محتوا **encrypt** شده‌اند، خبر دارد.

مرحله‌ی هفتم: کد مربوط به خواندن محتوای **Linked List** از **shared memory** را پیاده‌سازی کنید. وقتی برنامه **read** می‌کند، باید محتوایی که می‌خواند را ابتدا **decrypt** کند و سپس برای کاربر چاپ کند.

مرحله‌ی امتیازی: یک **Makefile** برای **build** کردن همه‌ی فایل‌ها درست کنید و یک **Bash Script** بنویسید که اول **Make** کند، سپس ابتدا کد **Write** را اجرا کند و بعد با یک ثانیه تاخیر، کد **Read** را اجرا کند.

نکته: در فایل زیپی که آپلود می‌کنید، حداقل 3 فایل باید باشد:

۱- فایل هدر **Linked List**

۲- کد مربوط به فایل **read**

۳- کد مربوط به فایل **write**