سیستم عامل - بهار ۱۴۰۴

استاد: دکتر مهدی کارگهی

پروژه ۲: مبانی برنامه‌نویسی پایپ

طراحان: آرین فیروزی - سروش اصفهانیان

اهداف پروژه

هدف اصلی این پروژه آشنایی بیشتر با لوله[1] در سیستم عامل است. شما احتمالا با دستورات پایپ در زبان برنامه‌نویسی Bash در رابط‌های خط فرمان[2] آشنا هستید. در این پروژه هدف این است که در مرحله اول، یک مفسر بسیار محدود در قالب یک خط فرمان پیاده کنید که تعدادی از دستورات مربوط به پایپ را اجرا کند. در مرحله دوم، از شما خواسته می‌شود که عملکرد میان unnamed pipe و named pipe را با یک‌دیگر مقایسه کنید.

قسمت اول: پیاده‌سازی خط فرمان

آشنایی با دستورات Bash

در زبان برنامه‌نویسی Bash با استفاده از دستورات مختلفی می‌توان پایپ‌ها را پیاده‌سازی کرد. سه نمونه از این دستورات در ادامه به اختصار توضیح داده شده است.

دستور پایپ

دستور پایپ در واقع بین دو دستور قرار می‌گیرد و خروجی دستور اول را به ورودی دستور بعدی انتقال می‌دهد. به عنوان مثال دستور زیر:

```
cat example.txt | grep "pipe"
```

باعث می‌شود که متن داخل فایل example توسط دستور cat خوانده شود و سپس دستور grep را بر روی مقدار خروجی، یعنی محتویات فایل، اجرا می‌کند. در نتیجه این دستور خطوطی از فایل example.txt که کلمه "pipe" در آن ذکر شده را نشان می‌دهد.

---

[1] pipe
[2] command-line interface

دستور های Redirection

این دستورات خروجی دستور قبلی را به یک فایل منتقل می‌کنند. به عنوان مثال دستور زیر:

```
cat example.txt > example2
```

باعث خواهد شد که محتویات داخل فایل example.txt در فایل example2 ذخیره شود.

پایپ در لینوکس، مانند بسیاری از موجودیت‌های دیگر، به عنوان یک فایل شناخته می‌شود. در نتیجه، می‌توان با استفاده از همین دستور، به جای فایل عادی دستور را به یک named pipe انتقال داد. به عنوان مثال دستورات زیر را در نظر بگیرید:

```
mkfifo named_pipe
cat example.txt > named_pipe
cat < named_pipe
```

در نتیجهٔ اجرای این دستورات، فایل در پایپ نوشته شده و توسط خط سوم قابل خواندن خواهد بود.

شرح عملکرد سیستم

در این پروژه از شما خواسته می‌شود که یک خط فرمان ساده پیاده‌سازی کنید که سه دستور ذکر شده را به همراه دستوراتی که در ادامه ذکر می‌شوند پیاده‌سازی کند. در پیاده‌سازی دستورات redirection نیازی به پشتیبانی از نوشتن در فایل نیست و تنها پیاده‌سازی نوشتن و خواندن در named pipe کافی است.

خط فرمان پیاده سازی شده توسط شما به صورت پیش فرض تنها یک named pipe دارد و نیازی به دستور ایجاد آن نیست. این لوله یک اسم ثابت دارد و در صورتی که هنگام parse کردن دستورات واژه np مشاهده شد، فرض بر آن است که دستور خواندن یا نوشتن بر روی همان لوله باید اجرا شود.

برای پیاده سازی unnamed pipe باید مشابه سیستم عامل لینوکس، پردازه پدر fork شود و یک unnamed pipe بین دو پردازه ایجاد شود. خروجی دستور اول از طریق این پایپ به ورودی دستور دوم وصل شده و دو دستور پشت سر هم اجرا می‌شوند.

ترتیب اجرای دستورات از چپ به راست بوده و نیازی به پیاده‌سازی اولویت‌بندی (مانند پرانتز گذاری) نیست.

به علت دشواری کار با رشته[3] ها در زبان c++ میتوانید از هر کتابخانه ای به این منظور استفاده کنید.

[3] string

دستورات در یک حلقه اجرا می‌شوند و در صورتی که کاربر واژه exit را در خط فرمان بنویسد برنامه خاتمه پیدا می‌کند.

محدودیتی در تعداد دستورات وجود ندارد و کاربر باید بتواند هر تعداد دستور را در یک خط پشت سر هم اجرا کند. توجه داشته باشید که دستورات ممکن است امکان اجرای پشت سر هم نداشته باشند، در نتیجه باید دستورات را با مکانیسم مدیریت خطای مناسب اجرا کنید. توجه کنید که فرض بر این است که دستورات و آرگومانهای آن‌ها به تنهایی درست نوشته می‌شوند و تنها در توالی آنها که مربوط به pipe است ممکن است خطا رخ دهد.

دستورات مورد نیاز در خط فرمان

1. دستور echo:

این دستور یک رشته به عنوان ورودی گرفته و آن را در stdout سیستم چاپ می‌کند.

2. دستور print_n:

این دستور یک آرگومان عددی دریافت کرده و به تعداد آن واژه ثابت "hello" را به شکل زیر چاپ می‌کند. برای مثال دستور:

```
print_n 2
```

خروجی زیر را چاپ می‌کند:

```
hello, hello
```

3. دستور len:

این دستور یک رشته را به عنوان ورودی گرفته و طول آن را چاپ می‌کند.

نمونه‌ای از اجرای برنامه به صورت زیر است:

```
print_n 3 | echo | len | echo
15
print_n 2 > np
print_n 1 > np
echo < np
hello, hello
echo < np | len
5
exit
```

قسمت دوم: تحلیل و مقایسه عملکرد انواع Pipe

در این قسمت، ابتدا به چند سوال تشریحی در مورد نحوه عملکرد هر کدام از انواع pipe جواب می‌دهید و سپس به صورت عملی به انجام مقایسه عملکرد این دو نوع pipe می‌پردازید.

سوالات تشریحی:

در مورد پایپ‌های نام‌دار[4] و بی‌نام[5] و ساز و کار هر کدام تحقیق کنید و به سوالات زیر پاسخ دهید:

1. توضیح دهید که در سیستم‌عامل لینوکس، انتقال داده بین فرایندها از طریق پایپ‌های نام‌دار و بی‌نام چگونه انجام می‌شود و هر نوع پایپ از چه مکانیزمی بدین منظور استفاده می‌کند؟ تفاوت اصلی در نحوهٔ ایجاد، محل نگهداری داده‌ها، و نحوهٔ اتصال فرایندها چیست؟
2. ویژگی‌های عملکردی پایپ‌های نام‌دار و بی‌نام را مقایسه کنید. وجود یا تفاوت در مقدارِ کدام عوامل در سطح سیستم مانند زمان بهره‌وری از CPU، فراخوانی‌های سیستمی، دسترسی به فایل‌سیستم و...، باعث تفاوت در عملکرد آن‌ها می‌شود و چگونه؟
3. در چه شرایطی استفاده از پایپ نام‌دار نسبت به پایپ بی‌نام مناسب‌تر است و یا برعکس؟ مصالحه[6] میان انعطاف‌پذیری، هم‌گام‌سازی[7]، طول عمر پایپ و عملکرد آن را در استفاده از این دو نوع پایپ به اختصار توضیح دهید.

مقایسه عملی پایپ نام‌دار و بی‌نام:

در این مرحله به صورت عملی، دو برنامه که هر کدام مربوط به استفاده از یکی از دو نوع پایپ هستند را اجرا می‌کنید و گزارش مصرف منابع آن‌ها را تحلیل و مقایسه خواهید کرد. بدین منظور، دو فایل اسکریپت شِل به نام‌های named.sh و unnamed.sh در اختیار شما قرار داده شده است که یک منطق یکسان برنامه را اجرا می‌کنند و بدین منظور، فایل اول از پایپ نام‌دار و فایل دوم از پایپ بی‌نام استفاده می‌کند. برای فهم راحت‌تر، محتویات این دو فایل در زیر قابل مشاهده است:

```
# named.sh
for i in $(seq 1 1000); do
  mkfifo /tmp/f  # to create unnamed pipe
  (cat /tmp/f > /dev/null &)
```

[4] named pipe
[5] unnamed pipe
[6] trade-off
[7] synchronization

```
  dd if=/dev/zero bs=1M count=1 of=/tmp/f status=none
  rm /tmp/f
done

# unnamed.sh
for i in $(seq 1 1000); do
  dd if=/dev/zero bs=1M count=1 status=none | cat > /dev/null
done
```

منطق اجرای هر دو برنامه به این صورت است که در هر مرتبه تکرار، بلوکی به اندازه یک مگابایت از یک پایانه خوانده شده و سپس توسط دستور دیگری در پایانه‌ای دیگر نوشته می‌شود که این کار توسط دو دستور dd و cat انجام شده که ارتباط میان این دو یک بار توسط پایپ نام‌دار و بار دیگر توسط پایپ بی‌نام فراهم شده است (دقت شود که نماد | در فایل دوم بیانگر و معادل استفاده از پایپ بی‌نام در برنامه می‌باشد).

همچنین در کنار این دو فایل، فایلی به نام benchmark.sh نیز در اختیار شما قرار داده شده است. این برنامه، دو آرگومان دریافت می‌کند که آرگومان اول، نام برنامه هدف (named.sh و یا unnamed.sh) و آرگومان دوم، نام فایل خروجی است. عملکرد برنامه benchmark.sh بدین صورت است که برنامه هدف را به تعداد ۱۰ مرتبه انجام می‌دهد، و در نهایت میانگین چند معیار عملکردی را در این ۱۰ مرتبه اجرا با استفاده از دستور time به دست می‌آورد و در فایل خروجی چاپ می‌کند. برای مثال، برای به دست آوردن مقدار میانگین معیارهای عملکردی، در اجرای برنامه named.sh، دستور زیر را در خط فرمان وارد می‌کنیم (می‌توانید به جای bash، از شل مورد نظر خود استفاده کنید.):

```
bash benchmark.sh named.sh out.txt
```

حال از شما خواسته شده است تا برنامه benchmark را یک بار برای برنامه named.sh و یک بار برای unnamed.sh اجرا کنید و مقادیر معیارهای عملکردی موجود در خروجی آن دو را با یک دیگر مقایسه، تحلیل و توجیه کنید (برای مقایسه دو فایل خروجی می‌توانید از دستور diff استفاده کنید).

در مورد علت تفاوت در مقادیر معیارهای عملکردی توضیح دهید. آیا مقادیر حاصل شده، با انتظارات بیان شده شما در سوال تشریحی دوم همخوانی دارد؟ اگر بله، چگونه و اگر خیر، علت آن چیست؟

نکات و نحوهٔ تحویل

- کدهای شما می‌بایست تنها به زبان C/C++ نوشته شده و همچنین در سیستم عامل Linux کامپایل و اجرا شوند، در غیر این صورت نمره‌ای به شما تعلق نمی‌گیرد.
- استفاده از داده‌ساختارها و امکانات زبان C++ و همچنین توابع انجام عملیات رشته‌ای (مانند sprintf strtol atoi) بلامانع است.
- برای پاسخ به قسمت دوم پروژه (تحلیل و مقایسه عملکرد انواع Pipe)، گزارش کار، شامل پاسخ سوالات تشریحی، عکس‌های مربوط به خروجی‌های خواسته شده داده شده و نیز، توضیحات خواسته شده باید تحویل داده شود.
- تمامی فایل‌های خود از قبیل کد، Makefile و گزارش کار قسمت دوم را در پوشه‌ای تحت عنوان OS-CA2-StudentNumber قرار داده و به فرمت فایل zip در سامانه بارگذاری کنید. در هنگام تحویل، تنها فایل‌های بارگذاری‌شده در سامانه پذیرفته خواهند شد.
- این تمرین صرفا برای یادگیری شما طرح شده است. در صورت محرز شدن تقلب در تمرین، مطابق با قوانین درس برخورد خواهد شد.
- استفاده از هوش مصنوعی به عنوان *ابزار* بلامانع است. به این نکته توجه کنید که هدف یادگیری شماست در نتیجه انتظار می‌رود هنگام تحویل پروژه به کد نوشته شده مسلط باشید.
- سوالات خود را تا حد ممکن در صفحه‌ی ایلرن درس مطرح کنید تا سایر دانشجویان نیز از پاسخ آنها بهره‌مند شوند. در صورتی که قصد مطرح کردن سوال جزئی‌تری داشتید، از طریق ایمیل با طراحان این تمرین ارتباط برقرار کنید.
- مواردی که در صفحه ایلرن و یا گروه تیمز درس و ویدیوهای کمکی توضیح داده می‌شوند جزئی از پروژه خواهند بود.

موفق باشید!