GEBZE TECHNICAL UNIVERSITY
COMPUTER ENGINEERING

2017 SPRING
CSE 244 SYSTEM PROGRAMMING

MIDTERM REPORT

141044041
BURAK KAĞAN KORKMAZ

https://github.com/burakkorkmaz/CSE244_HWs

REQUIREMENTS

1) Server

./timeServer <ticks in miliseconds> <n> <mainpipename>

<ticks in miliseconds> : Serverın kaç milisaniyede bir clientdan gelen requestleri kontrol edeceği değer.
<n> : (Integer) Server her client için 2n x 2n invertible matris oluşturacak.
<mainpipename> : Server ile clientin haberleşeceği ana fifonun adı.

Her client için bir process oluşturacak ve o processler her bir client için random 2n x 2n invertable bir matris oluşturacak ve clienta gönderecek.

Client request sinyalini ana fifo üzerinden gönderecek ve server client için oluşturduğu matrisi başka bir fifo üzerinden gönderecek.

2) Client

./seeWhat <mainpipename>

Bu program serverdan veri (2n x 2n invertible matris) alabilmek için servera request sinyali gönderecek. Aynı zamanda main fifo üzerinden pid sini de göndermesi gerekiyor. Serverdan matrisi alacak ve bu matrisle ilgili bazı işlemler yapacak. Bu işlemleri yapmak için en az 2 tane process oluşturmalıdır. Oluşturacağı processlerden biri n x n lik **shifted inverse** matrisi bulacak ve result1 i hesaplayacak.

Result1 = det(orjinal matris) – det(shifted inverse matris)

Diğer proses **2d convolution matris** oluşturak ve result2'yi hesaplayacak.

Result2 = det(orjinal matris) – det(2d conconvolution)

Program result1 ve result2'nin ne kadar zamanda(time elapsed) hesaplandığını bulması gerekiyor.

3) Results

./showResult <⋯>

Her clientdan result1, result2 ve clientın pidsini alacak ve onları dosyasına ve ekrana aşağıdaki gibi yazdıracak:

pid	Result1	Result2
...	...	...

1.Problem Solution Approach

Projeye öncelikle random matris oluşturmakla başladım. MatrixGeneretor parametre olarak size alıyor ve srand kullanarak random matrisi oluşturuyor.
Determinant işleminin daha hızlı çalışabilmesi için iterative olarak Sayısal Analiz dersinde öğrendiğim Gaussian Metodunu ve Pivot değişimini (swapRows fonksiyonunu) uyguladım.
Cofactor ve Transpose fonksiyonları için https://www.cs.rochester.edu/~brown/Crypto/assts/projects/adj.html isimli linkten yararlandım. Gerekli değişiklikleri yaparak InversoOfMatrix fonsiyonunu yazdım.

TimeServer.c;

TimeServer'a genel olarak baktığımızda bir mainServer ve buna bağlı (her bir client için) bir childServer mevcut durumdadır. MainServer sadece mainFifo ile Client'ın PID numarasını alır ve forklayıp childServer'a yollar. Bundan sonraki tüm iletişim client ile childServer arasında süregelmektedir.

Daha yakından bakacak olursak öncelikle,
timeServer'ı oluşturuken ctrl+c'yi yakalayacak signalHandler fonksiyonunu yazdım. Sinyali yakalarken "signal.h" kütüphanesinde bulunan sigaction struct'ını ve fonksiyonunu kullandım. Mainfifoyu oluşturup kontrollerini yaptım. SignalHandler'a bağlı bir global flag eşliğinde server'ın çalışacağı dögüyü oluşturdum.

Döngü içinde client'in pid numarasını alarak forklayıp her bir client için ayrı childServerların çalışmasını sağladım. ChildServer her client'a özel (client pid kullanarak) bir fifo dosyası oluşturuyor. İlk olarak bu fifo üzerinden kendi pid numarasını client e yollayarak Client-ChildServer iletişimini kuruyor. Ardından matrisin size'ını ve döngü ile matrisin elemanlarını client'a gönderiyor.

SIGINT(Ctrl^C) sinyali ile karşılaştığında gerekli işlemleri yaparak bitiriyor.

SeeWhat.c;

Öncelikle SIGTERM ve SIGINT sinyallerini (varsa) yakalıyor. Server kapatılırsa clientlere SIGTERM sinyali yolluyor. Böylece sistem eş zamanlı olarak kapanması sağlanıyor. Eğer client'a SIGINT sinyali

gönderilirse onu yakalıyor ve servera haber vermek üzere sinyal yolluyor.

Client’ın çalışma sürecini incelersek önce mainFifo’ya pid numarasını yazarak İletişimin kurulmasını sağlıyor. İletişim kurulduktan sonra kendisine ait olan fifoyu okuma modunda açarak server’ın PID numarasını ve matrisin boyutunu (2n) alıyor. Matris boyutunu kullanarak yeni bir matris oluşturuyor.

Sonraki adımda matrisin elemanalrını dögü ile teker teker alıp oluştuduğu matrisi dolduruyor. Artık Shifted Inverse ve 2D Convolution işlemleri gerçekleştirilebilir.

SIGINT ya da SIGTERM aldığında gerekli işlemleri yapıp bitirir.

Running and Results

Argüman Kontrolleri

```
eksor@Eksor:~/GitHub/CSE244_HWs/Midterm$ make
rm -f *.o timeServer seeWhat showResults fifo*
gcc -c timeServer.c
gcc timeServer.o -o timeServer
gcc -c seeWhat.c
gcc seeWhat.o -o seeWhat
eksor@Eksor:~/GitHub/CSE244_HWs/Midterm$ ./timeServer 5 0 fifo
Error: Invalid Matrix Size!
Usage: ./timeServer <ticks in milliseconds> <matrix size n> <main fifoname>
eksor@Eksor:~/GitHub/CSE244_HWs/Midterm$
```

```
eksor@Eksor:~/GitHub/CSE244_HWs/Midterm$ make
rm -f *.o timeServer seeWhat showResults fifo*
gcc -c timeServer.c
gcc timeServer.o -o timeServer
gcc -c seeWhat.c
gcc seeWhat.o -o seeWhat
eksor@Eksor:~/GitHub/CSE244_HWs/Midterm$ ./timeServer 5 12 fifo
Error: Invalid Matrix Size!
Usage: ./timeServer <ticks in milliseconds> <matrix size n> <main fifoname>
eksor@Eksor:~/GitHub/CSE244_HWs/Midterm$
```

```
eksor@Eksor:~/GitHub/CSE244_HWs/Midterm$ make
rm -f *.o timeServer seeWhat showResults fifo*
gcc -c timeServer.c
gcc timeServer.o -o timeServer
gcc -c seeWhat.c
gcc seeWhat.o -o seeWhat
eksor@Eksor:~/GitHub/CSE244_HWs/Midterm$ ./timeServer 500
Usage: ./timeServer <ticks in milliseconds> <matrix size n> <main fifoname>
eksor@Eksor:~/GitHub/CSE244_HWs/Midterm$
```

```
eksor@Eksor:~/GitHub/CSE244_HWs/Midterm$ ./seeWhat
Usage: ./seeWhat <main fifoname>
eksor@Eksor:~/GitHub/CSE244_HWs/Midterm$
```