درس سیستم عامل

دکتر رضا انتظاری ملکی – دکتر وحید ازهری

تمرین سری پنجم

طراحان ..... محمدمهدی شامخی – پدرام خجسته راد

تاریخ انتشار ..... ۴ / ۱۰ / ۱۴۰۳

تاریخ تحویل ..... ۱۸ / ۱۰ / ۱۴۰۳

آداب نامه تمارین

- نمره هر تمرین از 100 می‌باشد و بارم هر سوال روبه‌روی آن نوشته شده است.
- پاسخ های خود را تنها در صفحه کلاس در **کوئرا** آپلود کنید.
- فرمت فایل ارسالی zip و شامل کد c، makefile و گزارش‌کار می‌باشد. نامگذاری فایل ارسالی باید مطابق زیر باشد:

HW#_StdID_StdName.zip

- گزارش‌کار ارسالی شامل تکه کدها همراه با توضیحات مربوط به آن‌ها باشد. همچنین نتایج مربوطه به صورت تصویر همراه با توضیحات آن‌ها درج شوند.
- به هیچ وجه تمرینی را از دیگران کپی نکنید. در صورت مشاهده تقلب و کپی در تمرینات، نمره هر دو طرف صفر در نظر گرفته می‌شود.

سوال اول) کار با pipe - مجموع اعداد اول (۱۵ نمره)

می‌خواهیم با کمک تعدادی فرایند فرزند (child process)، اعداد اول کوچکتر از صد میلیون را تشخیص داده و مجموع آن‌ها را محاسبه کنیم.

برای این کار، در ابتدا با استفاده از دستور زیر تعداد logical processor های موجود در سیستم را به دست بیاورید.

```
#include <unistd.h>
long nproc = sysconf(_SC_NPROCESSORS_ONLN);
```

سپس با ایجاد فرایندهای فرزند به این تعداد، محاسبات مورد نیاز را بین این فرایندها تقسیم کنید.
سعی کنید تا حد امکان، تقسیم محاسبات را به صورت متوازن و برابر بین فرایندهای فرزند انجام دهید.

فرایندهای فرزند، باید در هنگام پیدا کردن هر عدد اول، آن را با استفاده از pipe به فرایند والد منتقل کنند.
فرایند والد نیز در این حین، اعداد اول را از pipe دریافت می‌کند و حاصل جمع آن‌ها را محاسبه می‌کند.

در انتها، پس از اتمام محاسباتِ تمامی فرایندهای فرزند، فرایند والد مجموع محاسبه‌شده را در خروجی چاپ می‌کند.

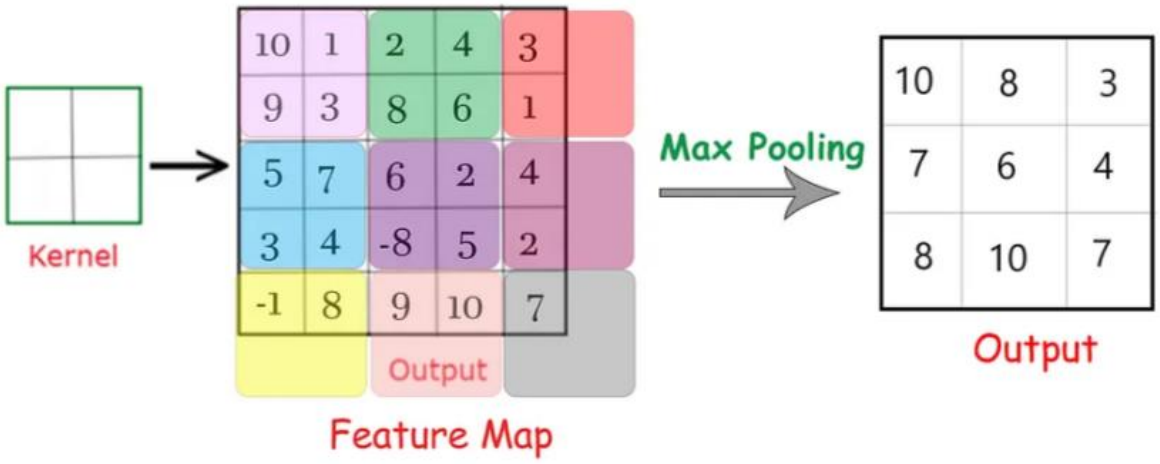
سوال دوم) Max Pooling (۲۵ نمره)

در این سوال به کمک حافظه اشتراکی POSIX، می‌خواهیم عملیات max pooling را روی ماتریس M1 با ابعاد M*N با استفاده از یک صفحه K*L انجام داده و خروجی را در ماتریس M2 ذخیره کنیم. برای درک بهتر این عملیات به مثال انتهای صفحه توجه کنید. $(M, N, K, L \leq 100)$

این عملیات کار زمانبری است، بنابراین می‌خواهیم این عملیات را به صورت همزمان با استفاده از چندین فرایند مختلف انجام دهیم. هر فرایند یک سطر از ماتریس جواب را محاسبه می‌کند. نتیجه محاسبات فرایند ها، ابتدا در یک حافظه مشترک ذخیره می‌شوند و پس از اتمام محاسبات از حافظه مشترک به ماتریس M2 در فرایند parent منتقل می‌شود.

مقادیر M, N, K, L باید به صورت hard-code با استفاده از define تعریف شوند.
مقداردهی ماتریس M1 باید به صورت رندوم با اعداد 1 تا 9 انجام شود.
نیازی نیست که ماتریس M1 هم در حافظه اشتراکی ذخیره شود.
حین اجرای کدی که نوشته‌اید، در ترمینال لاگ پرینت کنید تا همزمانی اجرای فرایندها مشخص باشد.

مثال:



در این قسمت صفحه 2×2 سمت چپ را روی ماتریس وسطی با ابعاد 5×5 حرکت می‌دهیم و در هر قسمت مقدار ماکسیمم را انتخاب می‌کنیم. نتیجه ماتریس سمت راست خواهد بود.

سوال سوم) Merge Sort (۲۵ نمره)

در این سوال قصد داریم به کمک حافظه اشتراکی POSIX، الگوریتم merge sort را به صورت توزیع‌شده پیاده سازی کنیم.
عملیاتی که باید انجام دهید، مرتب‌سازی یک آرایه به طول N است. حداکثر فرایندهای فرزندی که در هر لحظه از سیستم می‌توانند حضور داشته باشند نیز M است.

دقت کنید که M, N توانی از 2 هستند و باید به صورت hard-code با استفاده از define تعریف شوند.

نحوه مرتب‌سازی به این صورت است که فرایند والد، آرایه‌ای از اعداد رندوم 1 تا 100 به عنوان حافظه مشترک ایجاد کرده و سپس به کمک M فرایند ایجاد شده توسط والد، merge sort را انجام داده و قسمت‌های سورت شده به فرایند والد برگردانده می‌شود، حال دوباره فرایند والد تعدادی فرایند ایجاد می‌کند تا قسمت‌های سورت شده را ادغام کنند و این کار را تا زمانی ادامه می‌دهد که آرایه سورت شود.
آرایه سورت شده را با استفاده از فرایند والد در خروجی چاپ کنید.

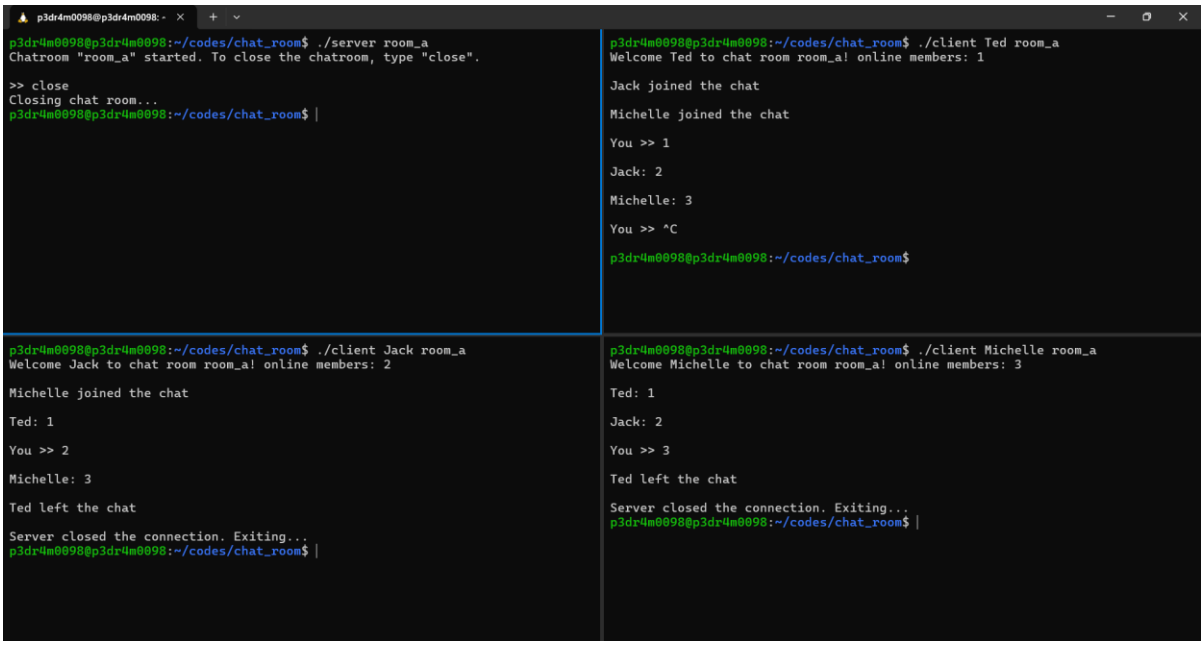
برای N = 2^20 بهترین M را پیدا کنید و زمانی که طول می‌کشد تا آرایه سورت شود را با زمانی که یک فرایند کل عملیات سورت را انجام می‌دهد یعنی M = 1 مقایسه کنید و نتیجه را در یک فایل گزارش دهید.

سوال چهارم) کار با named pipe – پیاده‌سازی اتاق گفتگو (۳۵ نمره)

با استفاده از named pipe (FIFO) یک برنامه اتاق گفتگو (chat room) طراحی کنید.

برنامه شما شامل دو فایل server.c و client.c خواهد بود که باید نمونه‌های مختلف از برنامه‌های server و client بتوانند به صورت مستقل اجرا شده و با یکدیگر کار کنند، به طوری که بتوان همزمان، چندین اتاق گفتگو و چند عضو در هر اتاق گفتگو داشت.

در تصویر زیر، نمونه‌ای از عملکرد برنامه را مشاهده می‌کنید:



ساختار ارتباط فرایندها باید به این صورت باشد:

- هر server، برای دریافت اطلاعات از سوی client ها، تنها یک FIFO ورودی داشته باشد.
- server برای ارسال اطلاعات به هر client، یک FIFO خروجی جداگانه ایجاد کند.

فهرست جزئیات عملکردهای مورد انتظار برنامه، در صفحه بعدی آورده می‌شود.

پیاده‌سازی شما باید عملکردهای زیر را پشتیبانی کند:

- اعلام ورود و خروج کاربران به اتاق
- رساندن پیام هر کاربر به بقیه کاربران
- دریافت پیام‌ها به صورت زنده و بدون تاخیر در client
- پشتیبانی از وجود کاربرانی با نام یکسان
- امکان دریافت ورودی از کنسول، همزمان با دریافت و پردازش اطلاعات فرایندهای دیگر
- آزادسازی منابع تخصیص‌یافته (memory, FIFO)، بعد از خروج کاربران و یا بسته شدن اتاق
- در صورت خروج همهٔ افراد اتاق، اتاق باید به فعالیت ادامه دهد، و کاربران بتوانند دوباره وارد اتاق شده و از آن استفاده کنند.
  - در صورتی که کاربری مشغول نوشتن پیام خود باشد، و پیامی از دیگران دریافت کند، نوشتهٔ او در صفحهٔ کنسول، overwrite خواهد شد؛ نیازی به حل این مشکل نیست.

- در صورتی که برای ارتباط client و server، پروتکل (روش تبادل اطلاعات) قابل توجهی در نظر گرفته‌اید، آن را در گزارش خود توضیح دهید.

راهنمایی:

- برای ذخیره اطلاعات کاربران فعلی اتاق، می‌توانید از یک لیست پیوندی (linked list) از struct اطلاعات کاربران استفاده کنید.
- می‌توانید با استفاده از signal، خروج کاربران و یا بسته شدن اتاق را با keyboard interrupt انجام دهید.
- می‌توانید برای راحتی استفاده از برنامه‌ها، ورودی (نام کاربر و یا اتاق) را به صورت آرگومان در نظر بگیرید.
- در صفحه بعدی، یک Makefile آورده می‌شود که می‌توانید از آن استفاده کنید.

Makefile:

```
# Compiler

CC = gcc

# Targets
TARGETS = server client

# Compile the programs
compile: $(TARGETS)

server: server.c
    $(CC) -lpthread -lrt -o server server.c

client: client.c
    $(CC) -lpthread -lrt -o client client.c

# Create a chat room
room:
    mkdir -p fifo_files
    @./server $(name)

# Create a chat member
member:
    @./client $(name) $(room)

clean:
    rm -f $(TARGETS)

help:
    @echo "Usage:"
    @echo "  make compile          Compile the programs"
    @echo "  make room name=<room_name>    Create a chat room"
    @echo "  make member name=<member_name> room=<room_name>    Create a chat member"
    @echo "  make clean            Clean up the compiled programs"

.PHONY: compile room member clean help
```