**BÀI THỰC HÀNH 7.7**

Họ và tên: Nguyễn Hải Dương

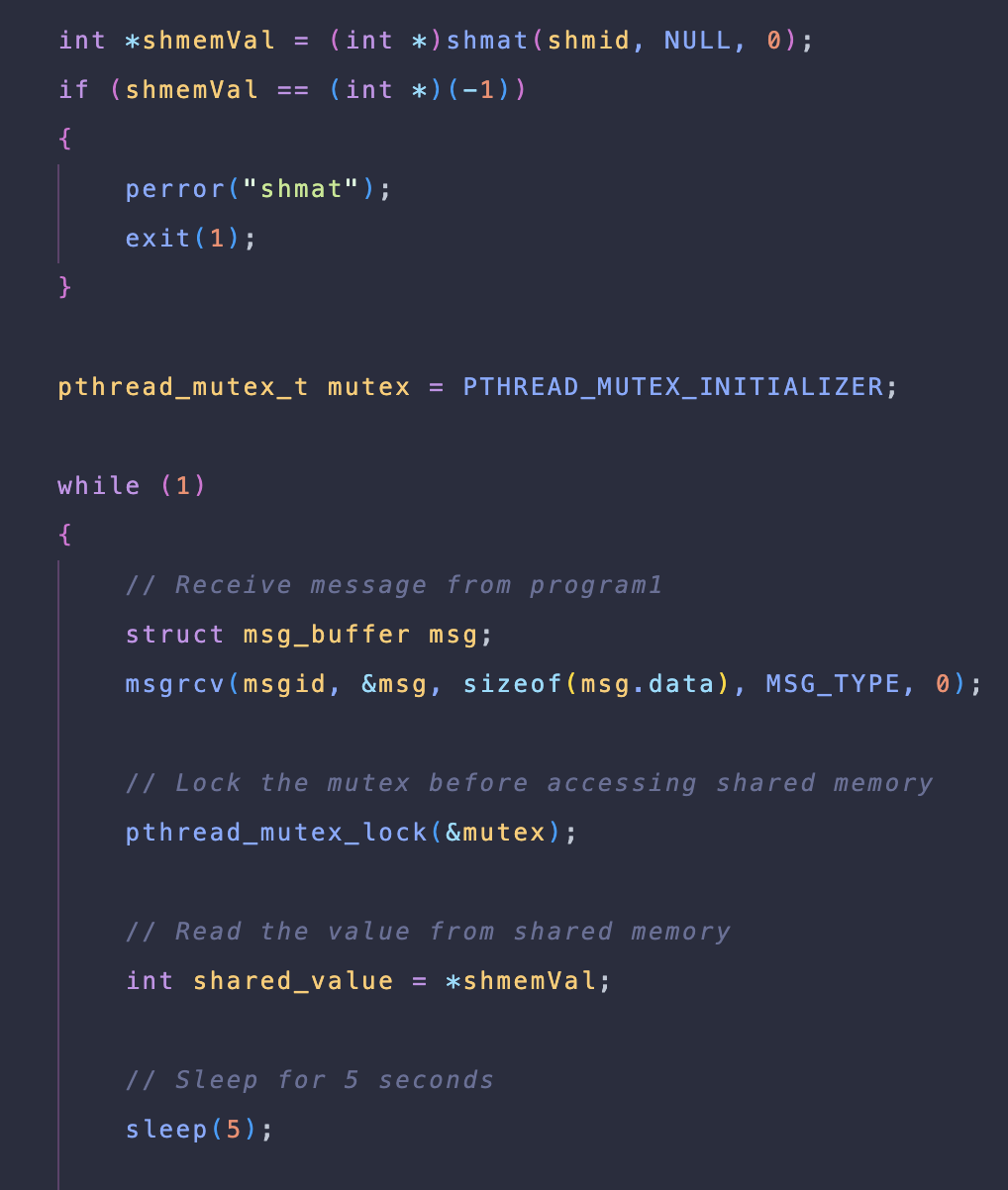
MSSV: 20194558

**1. Program 1**

File **opeProc.c**

A screen shot of a computer code

Description automatically generated with low confidence



A screenshot of a computer program

Description automatically generated with medium confidence

File **inputReceiptProc.c**

**A screen shot of a computer program

Description automatically generated with low confidence** **A picture containing text, screenshot, software, multimedia software

Description automatically generated**

**A screenshot of a computer program

Description automatically generated with low confidence**

**A picture containing text, screenshot

Description automatically generated**

**Giải thích**

File **opeProc.c**

* Hàm **main()** bắt đầu bằng cách tạo ra một khóa sử dụng **ftok()** để xác định hàng đợi tin nhắn. Sau đó, nó tạo hoặc mở hàng đợi tin nhắn bằng **msgget().**
* Sau đó, nó tạo ra một hàng đợi tin nhắn bằng cách sử dụng msgget với khóa đã tạo. Cờ IPC\_CREAT được sử dụng để tạo ra một hàng đợi tin nhắn mới nếu nó chưa tồn tại.
* Chương trình 1 bắt đầu một vòng lặp và chờ đợi tin nhắn người dùng nhập vào từ chương trình **inputReceiptProc**. Nó sẽ chặn cho đến khi nhận được một tin nhắn có loại MSG\_TYPE = 1.
* Khi một tin nhắn được nhận từ message queue, sử dụng mutex lock để tránh xung đột dữ liệu 🡪 giá trị **shared\_value** sẽ được cộng thêm với giá trị người dùng nhập vào thông qua message queue.
* Sau đó, nó gửi lại dữ liệu **shared\_value** mới thông qua message queue bằng hàm **msgsnd()**

File **inputReceiptProc**

* Chương trình dùng hàm **fork()** để tạo 2 tiến trình
  + Tiến trình con: Nhận data từ tiến trình cha 🡪 gửi data đó vào message queue và lắng nghe phản hồi
  + Tiến trình cha: Nhận dữ liệu từ người dùng 🡪 gửi cho tiến trình con
* Người dùng nhập 1 số nguyên dương vào bàn phím 🡪 tiến trình cha nhận giá trị đó và chuyển cho tiến trình con
* Tiến trình con nhận dữ liệu, gửi vào message queue để chương trình **opeProc.c** nhận được 🡪 Thực hiện theo như bước phía trên
* Sau khi chương trinh **opeProc.c** tính toán xong giá trị **shared\_value**, nhận giá trị đó và in ra màn hình console.

**2. Program 2**

File **accOrderCtrl.c**

A screen shot of a computer program

Description automatically generated

A screen shot of a computer code

Description automatically generated

A screen shot of a computer program

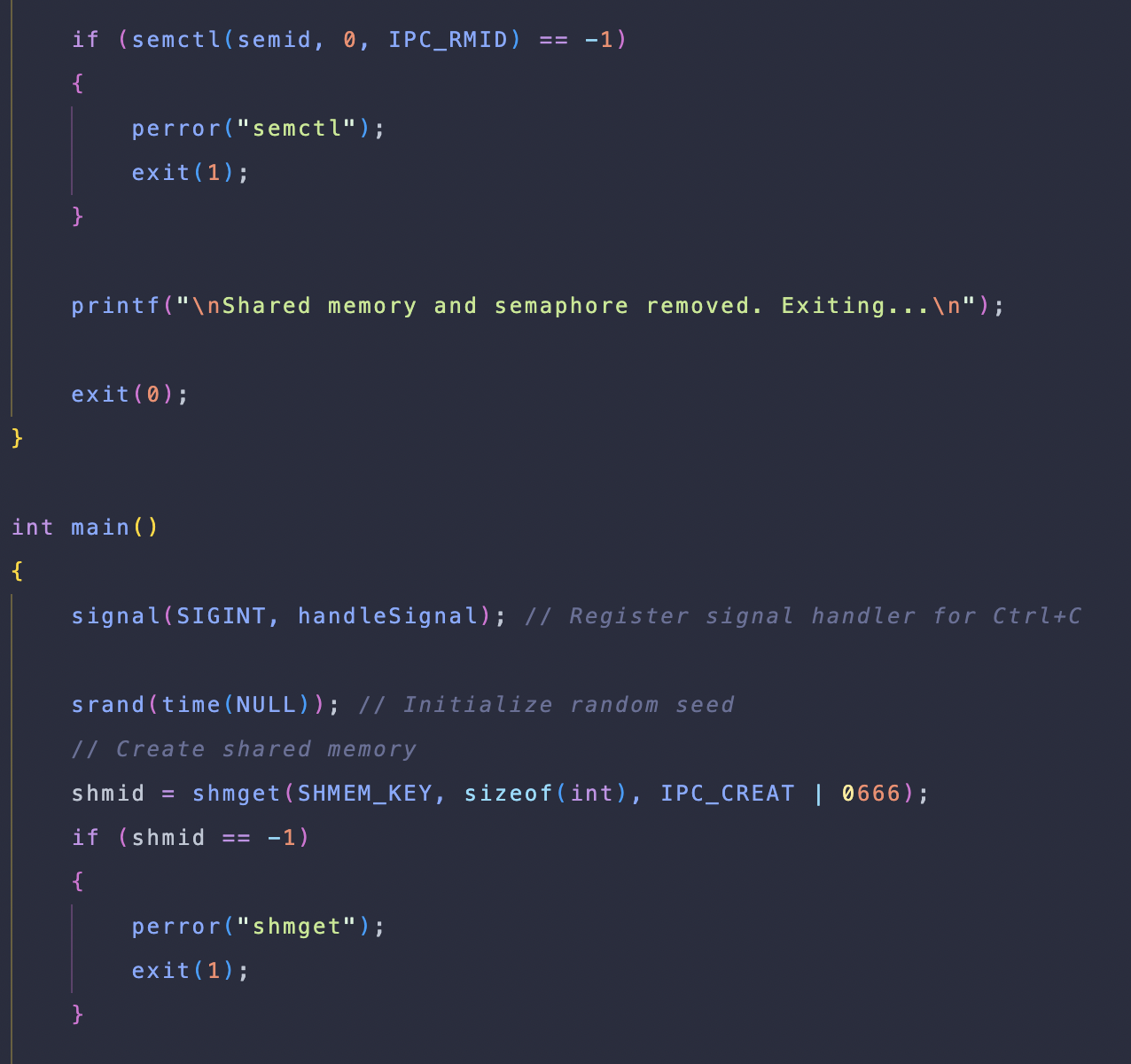
Description automatically generated

* Định nghĩa các hàm và kiểu dữ liệu cần thiết.
* Định nghĩa các hằng số và cấu trúc dữ liệu. **Request** là một hằng số đại diện cho loại tin nhắn cho yêu cầu truy cập bộ nhớ, **MAX\_RECORD** định nghĩa số lượng tối đa bản ghi truy vấn có thể lưu trữ, **NUMBER\_OF\_PROCESS** chỉ ra tổng số tiến trình và **FINISH** là loại tin nhắn cho một tiến trình hoàn thành truy cập.
* Vòng lặp chính bắt đầu. Trong vòng lặp này, tiến trình chờ yêu cầu truy cập bộ nhớ từ tiến trình khác. Nếu nhận được một tin nhắn yêu cầu từ một tiến trình khác mà không phải là tiến trình tiếp theo có quyền truy cập, nó lưu trữ yêu cầu trong mảng **inquiryRecord** để xử lý sau.
* Nếu nhận được yêu cầu truy cập từ tiến trình tiếp theo, nó gửi tín hiệu tới tiến trình đó để cho phép truy cập vào bộ nhớ. Sau đó, nó chờ tin nhắn kết thúc xử lý từ tiến trình đó.
* Sau khi tiến trình hiện tại hoàn thành truy cập, tiến trình chuyển sang tiến trình tiếp theo trong chuỗi vòng lặp, đồng thời kiểm tra xem có bất kỳ yêu cầu truy cập nào chờ đợi từ các tiến trình khác không phải là tiến trình tiếp theo. Nếu có, nó gửi tín hiệu cho tiến trình đó và xóa yêu cầu từ **inquiryRecord**.
* Quá trình trên tiếp tục lặp lại cho đến khi chương trình kết thúc hoặc xảy ra lỗi.

File **memAccProcGen.c**

**A screen shot of a computer code

Description automatically generated**

****

**A screen shot of a computer code

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

****

**A screenshot of a computer program

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

* Chương trình trên là một ví dụ về việc sử dụng các cơ chế **IPC (Inter-Process Communication)** trong ngôn ngữ lập trình C
* Tiếp theo, định nghĩa hai cấu trúc dữ liệu **data\_t** và **msg\_buffer**. Cấu trúc **data\_t** chứa thông tin về số tiến trình (**processNum**) và ID của tiến trình (**processId**). Cấu trúc **msg\_buffer** bao gồm một trường mtype kiểu long và một trường data kiểu data\_t.
* Sau đó, khai báo các biến toàn cục **shmid** (ID của shared memory), **semid** (ID của semaphore), **msgid** (ID của message queue)
* Tiếp theo là hàm **handleSignal** được định nghĩa để xử lý tín hiệu SIGINT (Ctrl+C). Trong hàm này, các phần tử IPC bao gồm shared memory, semaphore và message queue sẽ được xóa đi và chương trình sẽ thoát.
* Hàm **main** bắt đầu bằng việc gọi hàm **signal** để đăng ký bộ xử lý tín hiệu **handleSignal** cho tín hiệu SIGINT.
* Tiếp theo, hàm **srand** được gọi để khởi tạo seed cho hàm **rand**
* Tiếp theo, chương trình tạo shared memory thông qua hàm **shmget** với các tham số **SHMEM\_KEY**, **sizeof(int)**, **IPC\_CREAT | 0666**. Nếu việc tạo không thành công, thông báo lỗi được in ra và chương trình kết thúc.
* Tiếp theo, giá trị của semaphore được khởi tạo bằng 1 thông qua hàm **semctl** với các tham số **semid**, **0**, **SETVAL**, **arg**. Nếu việc khởi tạo không thành công, thông báo lỗi được in ra và chương trình kết thúc.
* Sau đó, một vòng lặp **for** được sử dụng để tạo ra 3 tiến trình con. Trong mỗi vòng lặp, hàm **fork** được gọi để tạo một tiến trình con.
* Trong tiến trình con, hàm **ftok** được sử dụng để tạo khóa (key) cho message queue
* Tiếp theo, vòng lặp vô hạn **while (1)** được sử dụng để thực hiện một số công việc nhất định trong mỗi tiến trình con.
  + Trước tiên, tiến trình con sẽ ngủ một khoảng thời gian ngẫu nhiên từ 1 đến 5 giây thông qua hàm **sleep** và in ra thông báo về thời gian ngủ.
  + Tiếp theo, tiến trình con sẽ gửi một thông điệp yêu cầu truy cập bộ nhớ đến tiến trình **accOrderCtrl** thông qua message queue. Thông điệp này chứa thông tin về số tiến trình và ID của tiến trình gửi đi.
  + Tiếp theo, tiến trình con sẽ gọi hàm **pause** để chờ tín hiệu **SIGUSR1**. Khi tiến trình nhận được tín hiệu này, nó sẽ tiếp tục thực hiện các công việc tiếp theo.
  + Sau đó, tiến trình con sẽ khóa semaphore thông qua hàm **semop** để đảm bảo chỉ có một tiến trình truy cập shared memory cùng một thời điểm.
  + Tiếp theo, tiến trình con sẽ gắn shared memory vào vùng nhớ của nó thông qua hàm **shmat**.
  + Sau đó, tiến trình con sẽ truy cập shared memory và thực hiện các tác vụ tùy thuộc vào giá trị của **i**.
    - Nếu **i** là 1, tiến trình con sẽ ghi "X=10, Y=0" vào **shmemVal**.
    - Nếu **i** là 2, tiến trình con sẽ tính toán (100 / X) và ghi "X=0, Y=10" vào **shmemVal.**
    - Nếu **i** là 3, tiến trình con sẽ tính toán (10000 / Y) và ghi "X=0, Y=0" vào **shmemVal.**
  + Sau khi tiến trình con hoàn thành công việc truy cập shared memory, tiến trình con sẽ mở khóa semaphore bằng cách gọi hàm **semop** với tham số **sem\_op** được đặt là **1**.
  + Trong khi đó, tiến trình cha cũng chạy trong một vòng lặp vô hạn **while (1)** để duy trì chương trình hoạt động.