

BÁO CÁO

**ĐỒ ÁN 3 HỆ ĐIỀU HÀNH
LẬP TRÌNH ĐỒNG BỘ VỚI
NACHOS**

Contents

1. THÔNG TIN NHÓM.....	1
2. THIẾT KẾ VÀ CÀI ĐẶT	2
2.1. Lớp Sem: (./userprog/stable.h)	2
2.2. Lớp STable (./userprog/stable.h)	2
2.3. Viết các system call	3
3. CHƯƠNG TRÌNH NGƯỜI DÙNG.....	6
TÀI LIỆU THAM KHẢO:	13

1. THÔNG TIN NHÓM

STT	MSSV	Họ tên	Mức độ đóng góp
1	1512034	Nguyễn Đăng Bình	100%
2	1512042	Nguyễn Thành Chung	100%
3	1512123	Hoàng Ngọc Đức	100%

Mức độ hoàn thành đồ án: 100%. Đã cài đặt cho chương trình chạy theo yêu cầu của đồ án.

2. THIẾT KẾ VÀ CÀI ĐẶT

2.1. Lớp Sem: (./userprog/stable.h)

Dùng để quản lý Semaphore. Các thuộc tính và phương thức

	Tên	Ý nghĩa
Thuộc tính	char name[50]	Lưu tên semaphore.
	Semaphore* sem	Dùng đối tượng semaphore đã cài đặt trong hệ thống để quản lý.
Phương thức	Sem()	Constructor của lớp
	~Sem()	Destructor của lớp
	wait()	Thực hiện thao tác chờ
	signal()	Thực hiện thao tác giải phóng Semaphore
	GetName	Trả về tên của semaphore

2.2. Lớp STable (./userprog/stable.h)

	Tên	Ý nghĩa
Thuộc tính	BitMap *bm	Quản lý slot trống của bảng.
	Sem* semTab [MAX_SEMAPHORE]	Quản lý các đối tượng lớp Sem. Số lượng tối đa là MAX_SEMAPHORE.
Phương thức	STable()	Constructor của lớp STable
	~STable()	Destructor của lớp STable
	int Create(char *name, int init)	Hàm tạo một semaphore mới có tên name và giá trị khởi tạo init và chèn vào bảng mô tả semaphore nếu thành công.
	int Wait(char *name)	Nếu tồn tại semaphore trong bảng thì thực hiện gọi semTab[i]->wait();
	int Signal(char *name)	Nếu tồn tại semaphore trong bảng thì thực hiện gọi semTab[i]->signal();
	int FindFreeSlot(int id)	Hàm tìm 1 slot trống trên bảng semTab chưa được dùng bởi semaphore nào cả.

Thực hiện khai báo lớp mới stable trong Makefile.common.

2.3. Viết các system call

Các system call về OpenFile đã báo cáo và cài đặt ở Project 1; các system call về đa chương đã báo cáo và cài đặt ở Project 2.

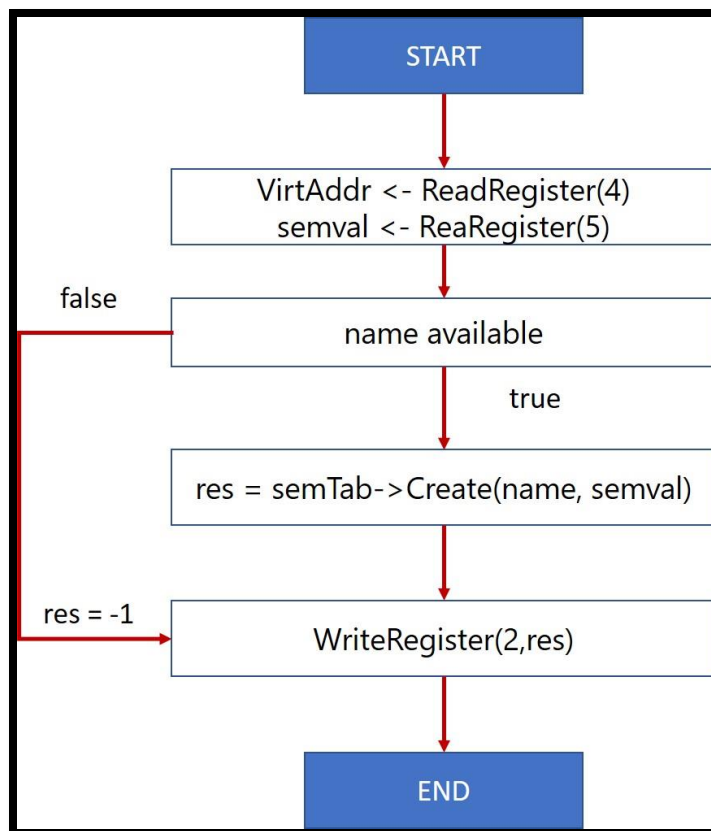
Khai báo biến toàn cục `STable* semTab` trong `./thread/system.h` và khởi tạo bên `./thread/system.cc`.

STable* semTab;

➤ Syscall CreateSemaphore

- Khai báo prototype `CreateSemaphore(char* name, int semval)` trong `./userprog/syscall.h`
- Cài đặt hàm `Create(char *name, int pid)` ở lớp `STable`.

Lưu đồ thuật toán:

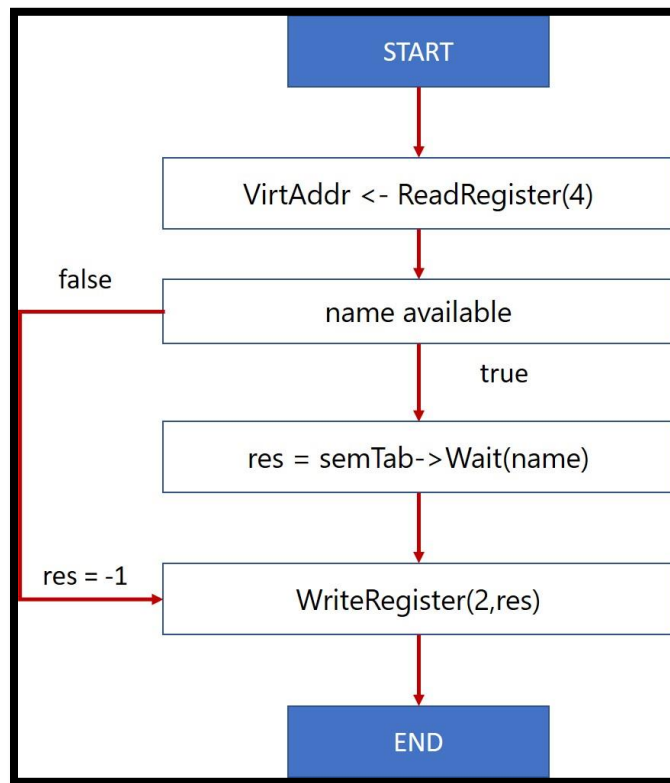


Hình 1. Lưu đồ thuật toán của system call
`CreateSemaphore`

➤ Syscall Wait

- Khai báo prototype `Wait(char* name)` trong `./userprog/syscall.h`
- Cài đặt hàm `Wait(char *name, int pid)` ở lớp `STable`.
- Cài đặt hàm `wait()` ở lớp `Sem`.

Lưu đồ thuật toán:

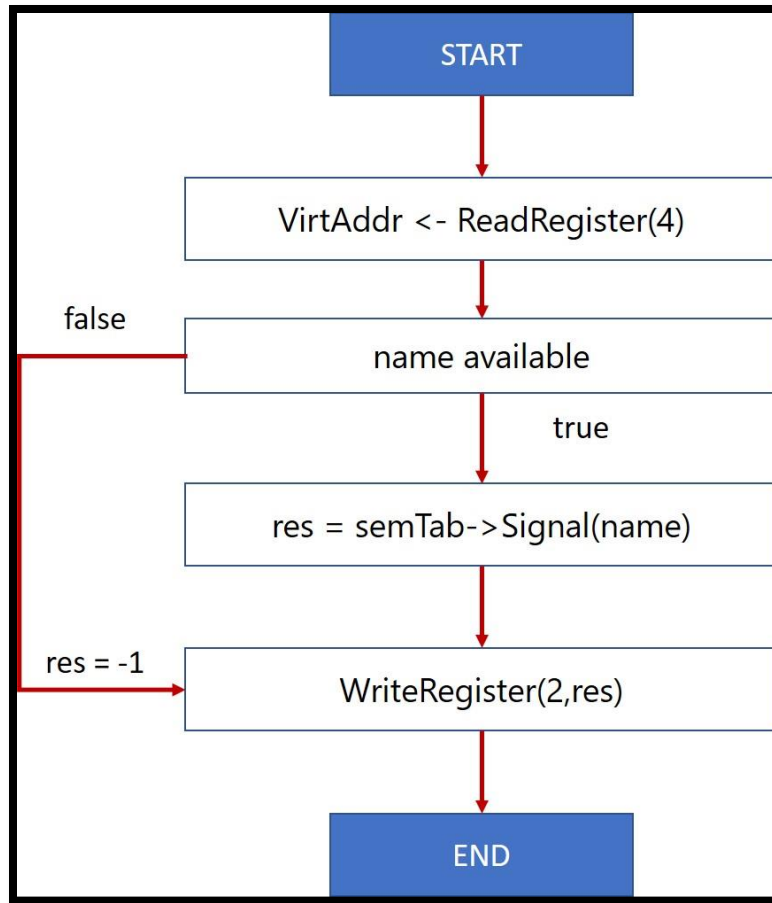


Hình 2 Lưu đồ thuật toán của system call Wait

➤ Syscall Signal

- Khai báo prototype `Signal(char* name)` trong `./userprog/syscall.h`
- Cài đặt hàm `Signal(char *name, int pid)` ở lớp `STable`.
- Cài đặt hàm `signal()` ở lớp `Sem`.

Lưu đồ thuật toán:



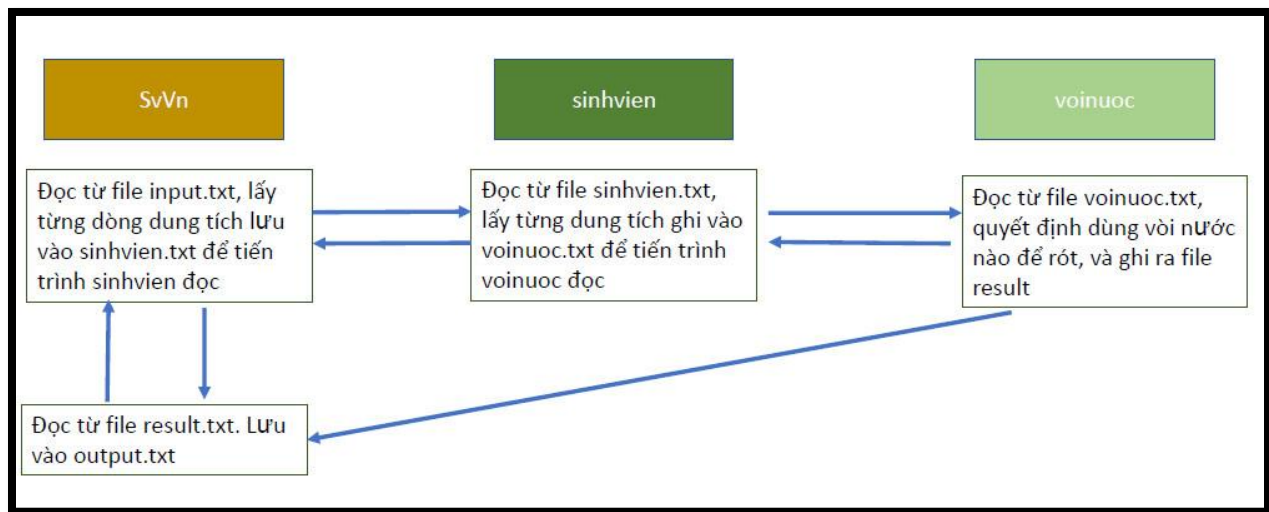
Hình 3 Lưu đồ thuật toán của system call Signal

3. CHƯƠNG TRÌNH NGƯỜI DÙNG.

Thực hiện việc đồng bộ, lập lịch đáp ứng yêu cầu bài toán để vòi nước rót nước cho các sinh viên. Mỗi vòi nước tại một thời điểm chỉ rót nước cho một sinh viên và có sử dụng 2 vòi nước.

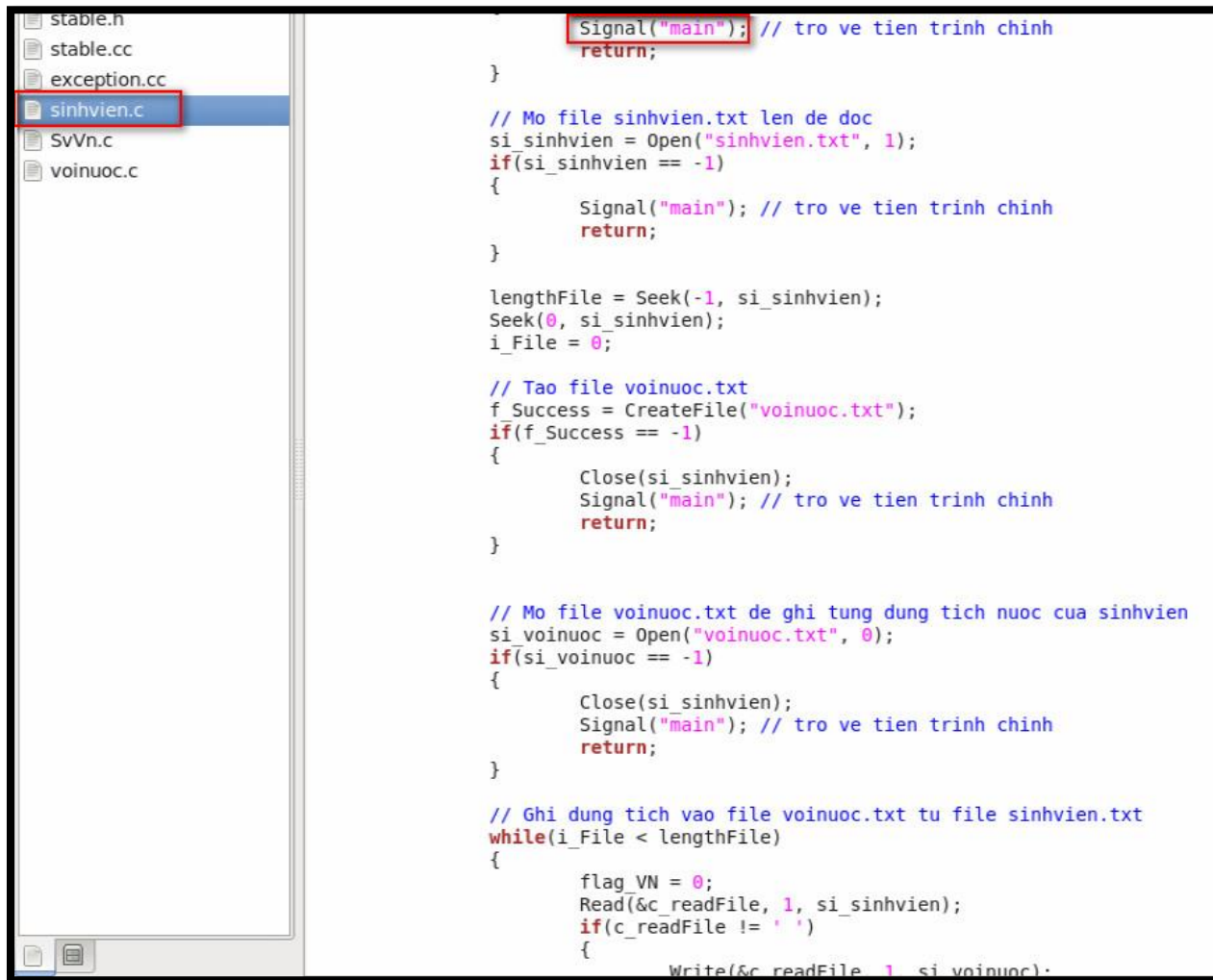
Ta tạo ba tiến trình SvVn (tiến trình chính), tiến trình sinhvien và voinuoc. Kết quả vòi nước nào thực hiện rót nước được ghi ở file output.txt

Mô hình kịch bản giao tiếp giữa các tiến trình.



Hình 4 Mô hình giao tiếp giữa các tiến trình

- Chương trình sinhvien



```
Signal("main"); // tro ve tien trinh chinh
return;

}

// Mo file sinhvien.txt len de doc
si_sinhvien = Open("sinhvien.txt", 1);
if(si_sinhvien == -1)
{
    Signal("main"); // tro ve tien trinh chinh
    return;
}

lengthFile = Seek(-1, si_sinhvien);
Seek(0, si_sinhvien);
i_File = 0;

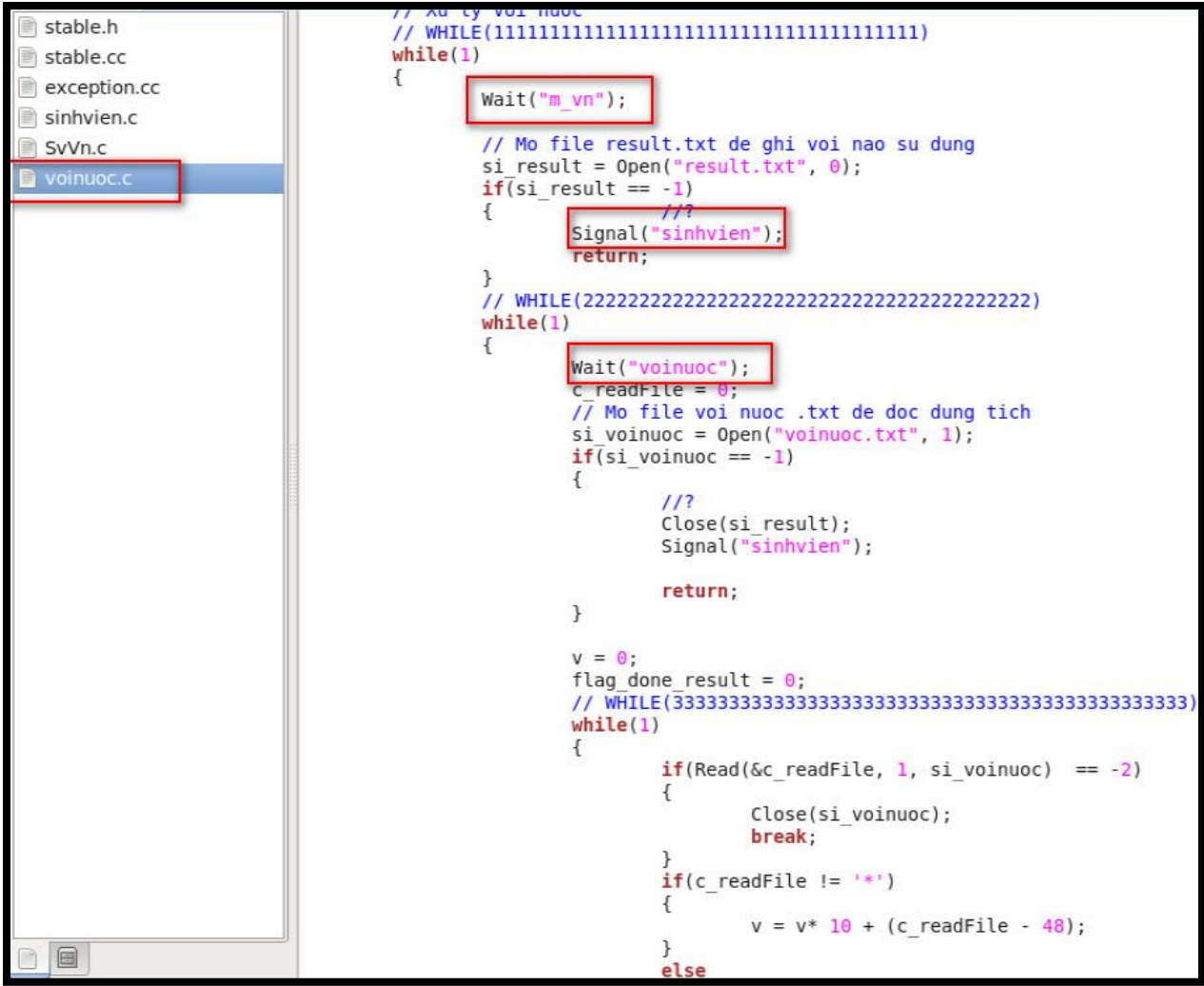
// Tao file voinuoc.txt
f_Success = CreateFile("voinuoc.txt");
if(f_Success == -1)
{
    Close(si_sinhvien);
    Signal("main"); // tro ve tien trinh chinh
    return;
}

// Mo file voinuoc.txt de ghi tung dung tich nuoc cua sinhvien
si_voinuoc = Open("voinuoc.txt", 0);
if(si_voinuoc == -1)
{
    Close(si_sinhvien);
    Signal("main"); // tro ve tien trinh chinh
    return;
}

// Ghi dung tich vao file voinuoc.txt tu file sinhvien.txt
while(i_File < lengthFile)
{
    flag_VN = 0;
    Read(&c_readFile, 1, si_sinhvien);
    if(c_readFile != ' ')
    {
        Write(&c_readFile, 1, si_voinuoc);
    }
}
```

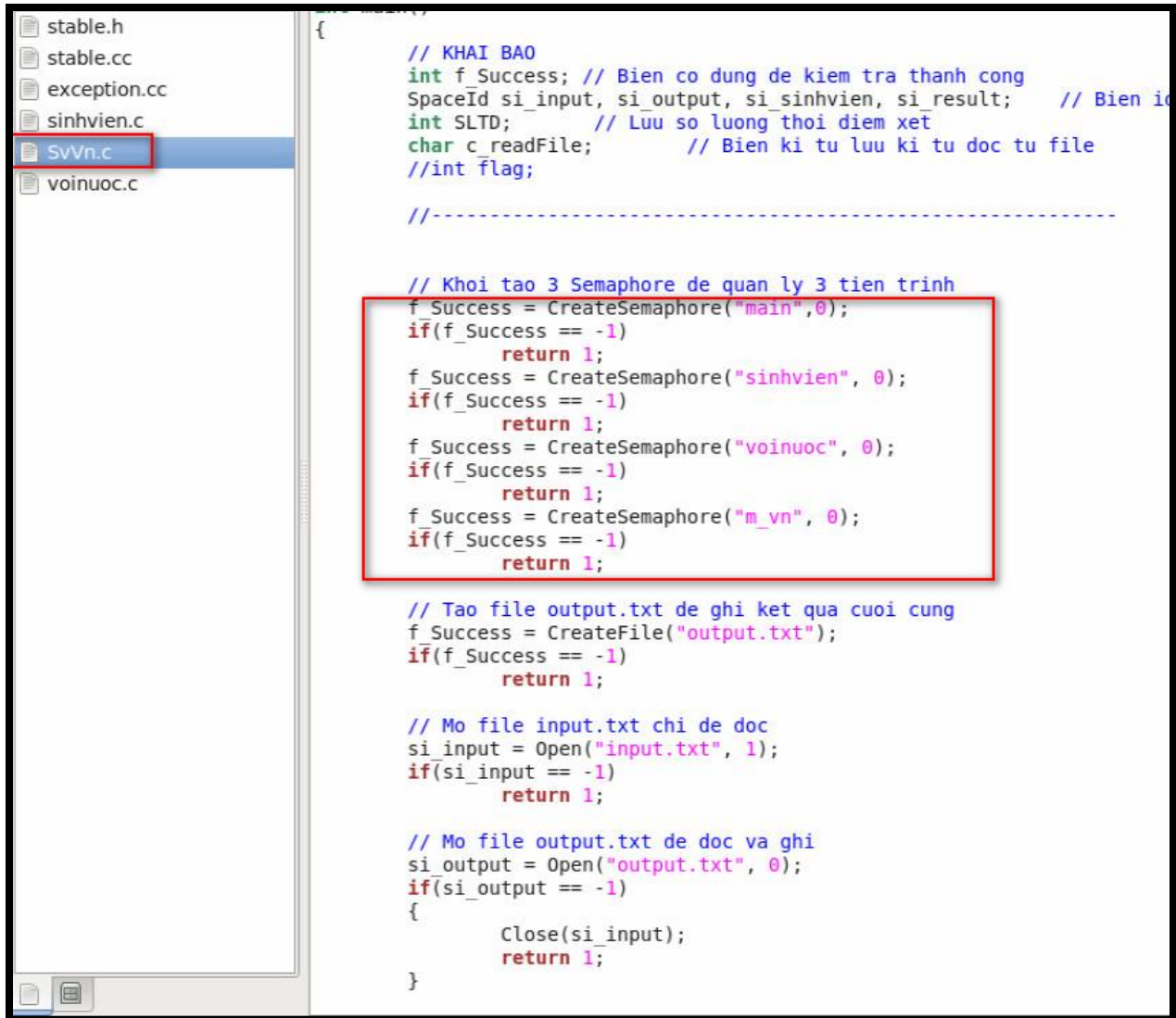
Hình 5 Chương trình sinhvien

- Chương trình voinuoc



Hình 6 Chương trình voinuoc

- Chương trình Scheduler



```
stable.h
stable.cc
exception.cc
sinhvien.c
SvVn.c
voinuoc.c

// KHAI BAO
int f_Success; // Bien co dung de kiem tra thanh cong
SpaceId si_input, si_output, si_sinhvien, si_result; // Bien id
int SLTD; // Luu so luong thoi diem xet
char c_readFile; // Bien ki tu luu ki tu doc tu file
//int flag;

//-----

// Khoi tao 3 Semaphore de quan ly 3 tien trinh
f_Success = CreateSemaphore("main", 0);
if(f_Success == -1)
    return 1;
f_Success = CreateSemaphore("sinhvien", 0);
if(f_Success == -1)
    return 1;
f_Success = CreateSemaphore("voinuoc", 0);
if(f_Success == -1)
    return 1;
f_Success = CreateSemaphore("m_vn", 0);
if(f_Success == -1)
    return 1;

// Tao file output.txt de ghi ket qua cuoi cung
f_Success = CreateFile("output.txt");
if(f_Success == -1)
    return 1;

// Mo file input.txt chi de doc
si_input = Open("input.txt", 1);
if(si_input == -1)
    return 1;

// Mo file output.txt de doc va ghi
si_output = Open("output.txt", 0);
if(si_output == -1)
{
    Close(si_input);
    return 1;
}
```

Hình 7 Chương trình SvVn khởi tạo các semaphore

```

exception.cc
sinhvien.c
SvVn.c
voinuoc.c

while(1)
{
    Read(&c_readFile, 1, si_input);
    if(c_readFile != '\n')
    {
        if(c_readFile >= '0' && c_readFile <= '9')
            SLTD = SLTD * 10 + (c_readFile - 48);
        else
            break;
    }
}

// Goi thuc thi tien trinh sinhvien.c
f_Success = Exec("./test/sinhvien");
if(f_Success == -1)
{
    Close(si_input);
    Close(si_output);
    return 1;
}

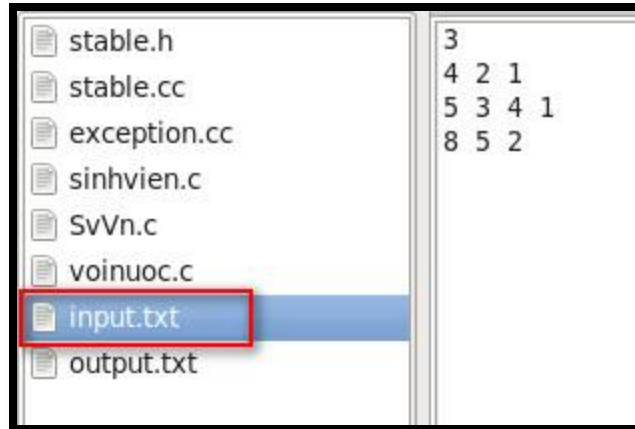
// Goi thuc thi tien trinh voinuoc.c
f_Success = Exec("./test/voinuoc");
if(f_Success == -1)
{
    Close(si_input);
    Close(si_output);
    return 1;
}

// Thuc hien xu ly khi nao het thoi diem xet thi thoi
while(SLTD--)
{
    // Tao file sinhvien.txt
    f_Success = CreateFile("sinhvien.txt");
    if(f_Success == -1)
    {
        Close(si_input);
        Close(si_output);
        return 1;
    }
}

```

Hình 8 Chương trình SvVn thực thi các tiến trình con

Kiểm tra thực thi của chương trình:



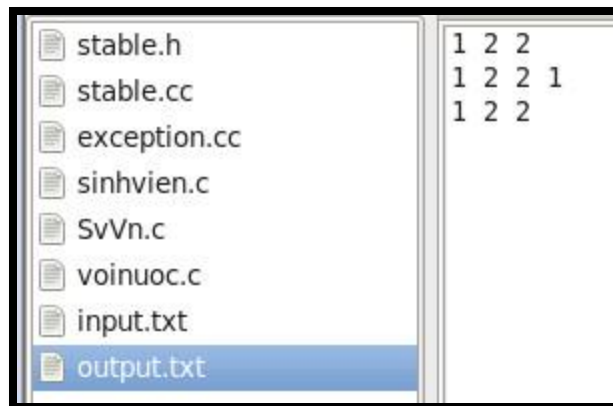
Hình 9 file input.txt

```
sv@localhost:~/hdh/project3/nachos-3.4/code
File Edit View Search Terminal Help
[sv@localhost code]$ ./userprog/nachos -rs 1023 -x ./test/SvVn

Size: 3072 | numPages: 6 | PageSize: 512 | Numclear: 256
Physic Pages 0
Physic Pages 1
Physic Pages 2
Physic Pages 3
Physic Pages 4
Physic Pages 5

Size: 2560 | numPages: 5 | PageSize: 512 | Numclear: 250
Physic Pages 6
Physic Pages 7
Physic Pages 8
Physic Pages 9
Physic Pages 10
```

Hình 10 Thực thi chương trình SvVn



Hình 11 file output.txt kết quả cuối cùng

TÀI LIỆU THAM KHẢO:

nachos_canban.pdf

nachos_study_book.pdf

DoAn3_NachOS.pdf

HuongDan_Project3.pdf

Huong Dan Cac Syscall Ve Da Chuong.pdf