ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN

MÔN HỌC: HỆ ĐIỀU HÀNH

ĐỀ TÀI: ĐA CHƯƠNG, LẬP LỊCH VÀ ĐỒNG BỘ HOÁ TRONG NACHOS

Giáo viên hướng dẫn:

ThS. Lê Viết Long

Thành phố Hồ Chí Minh, tháng 12 năm 2022

LỜI CẢM ƠN

Đầu tiên, nhóm em xin chân thành cảm ơn thầy ThS. Lê Viết Long, người đã hướng dẫn tận tình, giúp đỡ cho nhóm em hoàn thành đồ án này.

Bài tập đồ án lần này nhóm em đã cố gắng hết sức để hoàn thành nhưng cũng không tránh khỏi những sai sót. Vì thế em rất mong sự góp ý, chỉ bảo từ thầy cho em sau này có thể phát triển tốt hơn nữa.

Em rất mong nhận được sự quan tâm, giúp đỡ của thầy. Xin chân thành cảm ơn!

MỤC LỤC

1. Tổng quan

1.1. Thông tin thành viên trong nhóm:

- Phan Phong Lưu – 20120326
- Phạm Bảo Huy – 20120298
- Ngô Võ Quang Huy – 20120295

1.2. Bảng phân công công việc:

Thành viên / Công việc	Ngô Võ Quang Huy 20120295	Phạm Bảo Huy 20120298	Phan Phong Lưu 20120326
Cài đặt System call: CreateFile, OpenFileID, Close, Read, Write	x		
Cài đặt System call: Exec, Join, Exit			x
Cài đặt System call: CreateSemaphore, Wait, Signal		x	
Xây dựng ứng dụng “Thống kê sử dụng máy nóng lạnh”		x	
Demo			x
Báo cáo	x		

1.3. Đánh giá đồ án

Yêu cầu	Mức độ hoàn thành (%)	Ghi chú
Cài đặt các system call: CreateFile, OpenFileID, Close, Read, Write, Exec, Join, Exit, CreateSemaphore, Wait, Signal	100	
Viết chương trình "Thống kê sử dụng máy nóng lạnh"	100	
Báo cáo	100	

- Đánh giá tổng thể mức độ hoàn thành: 100%

1.4. Thông tin cơ bản:

- Ngôn ngữ lập trình: C++, C
- Công cụ hỗ trợ: VMWare Workstation 16 Pro
- Hệ điều hành: Ubuntu 14.06
- Môi trường lập trình: Visual Studio Code, version 1.35

2. Mô tả các bước thực hiện:

2.1. Thiết kế

Bước 1: Khai báo các biến toàn cục trong ./threads/system.h:

```
extern Semaphore *addrLock;     // semaphore
extern BitMap *gPhysPageBitMap; // quan ly cac frame
extern PTable *pTab;            // quan ly bang tien trinh
extern STable *semTab;          // quan ly semaphore
```

Bước 2: Tạo các đối tượng trong ./threads/system.cc:

```
Semaphore *addrLock;     // semaphore
BitMap *gPhysPageBitMap; // quan ly cac frame
PTable *pTab;            // quan ly bang tien trinh
STable *semTab;          // quan ly semaphore
```

Bước 3:

- Tạo 3 file ./userprog/pcb.cc, ./userprog/ptable.cc và ./userprog/stable.cc để cài đặt lần lượt 4 lớp PCB, PTable, Semaphore và STable dựa trên các file ./userprog/pcb.h, ./userprog/ptable.h và ./userprog/stable.h có sẵn.
- Khai báo trong ./Makefile.common để quản lí tiến trình 3 lớp PCB, PTable và STable vừa thêm.

```
USERPROG_H = ../userprog/addrspace.h\
    ../userprog/bitmap.h\
    ../filesys/filesys.h\
    ../filesys/openfile.h\
    ../machine/console.h\
    ../machine/machine.h\
    ../machine/mipssim.h\
    ../machine/translate.h\
    ../threads/synchcons.h\
    ../userprog/pcb.h\
    ../userprog/ptable.h\
    ../userprog/stable.h

USERPROG_C = ../userprog/addrspace.cc\
    ../userprog/bitmap.cc\
    ../userprog/exception.cc\
    ../userprog/progtest.cc\
    ../machine/console.cc\
    ../machine/machine.cc\
    ../machine/mipssim.cc\
    ../machine/translate.cc\
    ../threads/synchcons.cc\
    ../userprog/pcb.cc\
    ../userprog/ptable.cc\
    ../userprog/stable.cc

USERPROG_O = addrspace.o bitmap.o exception.o progtest.o console.o machine.o \
    mipssim.o translate.o synchcons.o pcb.o ptable.o stable.o
```

Bước 4:

- Điều chỉnh lại số sector lên 512 và số khung trang lên 128.
    - Điều chỉnh số sector trong file ./machine/disk.h:

```
#define SectorSize    512 // number of bytes per disk sector
```

    - Điều chỉnh lại số khung trang trong file ./machine/machine.h:

```
#define NumPhysPages    128
```

Bước 5:

- Cài đặt thêm trong class Thread (./threads/thread.h):
    - Thêm biến processID kiểu int để kiểm tra, phân biệt các tiến trình khác nhau.
    - Thêm biến exitStatus kiểu int để kiểm tra exit code của tiến trình.

- Cài đặt hàm *`void FreeSpace()`* để giải phóng vùng nhớ trên bộ nhớ mà tiến trình đang sử dụng.

```
int processID;
int exitStatus;
void FreeSpace()
{
  if (space != 0)
    delete space;
}
```

Bước 6:

- Trong ./userprog/progtest.cc, cài đặt thêm hàm *`void StartProcess_2(int id)`*, hàm Fork sẽ gọi hàm này để trỏ tới vùng nhớ của tiến trình con.

```
void StartProcess_2(int id)
{
    char *fileName = pTab->GetFileName(id);

    AddrSpace *space;
    space = new AddrSpace(fileName);

    if (space == NULL)
    {
        printf("\nPCB::Exec : Can't create AddSpace.");
        return;
    }

    currentThread->space = space;

    space->InitRegisters();
    space->RestoreState();

    machine->Run();
    ASSERT(FALSE);
}
```

Bước 7:

- Trong ./userprog/addrspace.h và ./userprog/addrspace.cc, cài đặt bổ sung thêm để chuyển từ đơn chương sang đa chương:

- Cài đặt biến toàn cục *Bitmap* gPhysPageBitMap* để quản lí các frames. Điều này cho phép nhiều chương trình có thể nạp lên bộ nhớ cùng thời điểm.
- Giải phóng bộ nhớ khi user program kết thúc.
- Cài đặt biến *pageTable = new TranslationEntry[numPages]*, tránh để bộ nhớ biểu diễn liên tiếp nhau. Tìm trang còn trống bằng phương thức *Find()* của lớp Bitmap, tiếp theo nạp chương trình lên bộ nhớ chính.

```
pageTable = new TranslationEntry[numPages];
for (i = 0; i < numPages; i++) {
    // for now, virtual page # = phys page #
    pageTable[i].virtualPage = i;
    pageTable[i].physicalPage = gPhysPageBitMap->Find();
    pageTable[i].valid = TRUE;
    pageTable[i].use = FALSE;
    pageTable[i].dirty = FALSE;
    pageTable[i].readOnly = FALSE; // if the code segment was entirely on
                                   // a separate page, we could set its
                                   // pages to be read-only
}
```

Bước 8: Cài đặt các system call

2.2. Cài đặt các system call

2.2.1. CreateFile

- Khai báo hàm ở ./userprog/syscall.h:

```
int CreateFile(char *name);
```

- Xử lí SC_CreateFile trong ./userprog/exception.cc:

```
case SC_CreateFile: {
    // Input: Dia chi tu vung nho user space la ten file
    // Output: Tra ve 0 neu thanh cong, -1 neu bi loi
    // Chuc nang: Tao ra file voi tham so la ten file
    int virtAddr;
    char* filename;
    DEBUG('a', "\n SC_CreateFile call ...");
    DEBUG('a', "\n Reading virtual address of filename");

    //Doc dia chi cua file tu thanh ghi R4
    virtAddr = machine->ReadRegister(4);
    DEBUG('a', "\n Reading filename.");

    //Sao chep khong gian bo nho User sang System, voi do dang toi da la (32 + 1) bytes
    filename = User2System(virtAddr, MaxFileLength + 1);

    // Neu ten file rong
    if (strlen(filename) == 0) {
        printf("\n File name is not valid");
        DEBUG('a', "\n File name is not valid");
        machine->WriteRegister(2, -1);
        break;
    }

    //Neu khong du khong gian bo nho
    if (filename == NULL) {
        printf("\n Not enough memory in system");
        DEBUG('a', "\n Not enough memory in system");
        machine->WriteRegister(2, -1);
        delete filename;
        break;
    }

    DEBUG('a', "\n Finish reading filename.");

    //Tao file bang ham Create cua fileSystem, tra ve ket qua
    if (!fileSystem->Create(filename, 0)) {
        //Tao file that bai
        printf("\n Error create file '%s'", filename);
        machine->WriteRegister(2, -1);
        delete filename;
        break;
    }
    //Tao file thanh cong
    machine->WriteRegister(2, 0);
    delete filename;
    break;
}
```

2.2.2. OpenFileID

- Khai báo hàm ở ./userprog/syscall.h:

```
typedef int OpenFileId Open(char *name, int type);
```

- Xử lí SC_Open trong ./userprog/exception.cc:

```
case SC_Open: {
    // Input: Dia chi cua ten file va type 0: mo de doc va ghi, 1: mo chi doc
    // Output: Tra ve ID cua file kieu OpenFileID neu thanh cong, -1 neu bi loi
    // Chuc nang: Tra ve ID cua file.

    int virtAddr = machine->ReadRegister(4);
    int type = machine->ReadRegister(5);
    char* filename;

    // Copy chuoi tu vung nho User Space sang System Space voi chieu dai toi da cua ten file la MaxFileLength
    filename = User2System(virtAddr, MaxFileLength);

    int freeSlot = fileSystem->FindFreeSlot();
    //Chi xu li khi con slot trong
    if (freeSlot != -1) {
        if (type == 0 || type == 1) //chi xu li khi type = 0 hoac 1
        {
            fileSystem->oFile[freeSlot] = fileSystem->Open(filename, type);
            //Mo file thanh cong
            if ((fileSystem->oFile[freeSlot]) != NULL) {
                machine->WriteRegister(2, freeSlot); //tra ve OpenFileID
            }
        } else if (type == 2) // xu li stdin voi type quy uoc la 2
        {
            machine->WriteRegister(2, 0); //tra ve OpenFileID
        } else // xu li stdout voi type quy uoc la 3
        {
            machine->WriteRegister(2, 1); //tra ve OpenFileID
        }
        delete[] filename;
        break;
    }
    machine->WriteRegister(2, -1); //Khong mo duoc file tra ve -1
    delete[] filename;
    break;
}
```

2.2.3. Close

- Khai báo hàm ở ./userprog/syscall.h:

```
int Close(OpenFileId id);
```

- Xử lí SC_Close trong ./userprog/exception.cc:

```
case SC_Close: {
    //Input id cua file(OpenFileID)
    // Output: tra ve 0 neu thanh cong, -1 neu that bai
    // Chuc nang: Giai phong vung nho va dong file
    int fid = machine->ReadRegister(4);
    if (fid >= 0 && fid <= 14) //Chi xu li khi fid nam trong [0, 14]
    {
        if (fileSystem->oFile[fid]) //neu mo file thanh cong
        {
            delete fileSystem->oFile[fid]; //Xoa vung nho luu tru file
            fileSystem->oFile[fid] = NULL; //Gan vung nho NULL
            machine->WriteRegister(2, 0);
            break;
        }
    }
    machine->WriteRegister(2, -1);
    break;
}
```

2.2.4. Read

- Khai báo hàm ở ./userprog/syscall.h:

```
int Read(char *buffer, int charcount, OpenFileId id);
```

- Xử lí SC_Read trong ./userprog/exception.cc:

```
case SC_Read: {
    // Input: buffer(char*), so ky tu(charcount), id cua file(OpenFileID)
    // Output: tra ve -1 neu loi, hoac so byte doc thuc su neu thanh cong hoac tra ve -2 neu doc toi cuoi file
    // Cong dung: Doc file voi tham so la buffer, so ky tu cho phep va id cua file
    int virtAddr = machine->ReadRegister(4);
    int charcount = machine->ReadRegister(5);
    int id = machine->ReadRegister(6);
    int old_pos;
    int new_pos;
    char* buffer;
    // Kiem tra id cua file truyen vao co nam ngoai bang mo ta file khong
    if (id < 0 || id > 14) {
        printf("\nKhong the read vi id nam ngoai bang mo ta file.");
        machine->WriteRegister(2, -1);
        IncreasePC();
        return;
    }
    // Kiem tra file co ton tai khong
    if (fileSystem->oFile[id] == NULL) {
        printf("\nKhong the read vi file nay khong ton tai.");
        machine->WriteRegister(2, -1);
        IncreasePC();
        return;
    }
    // Xet truong hop doc file stdout (type quy uoc la 3) thi tra ve -1
    if (fileSystem->oFile[id]->type == 3) {
        printf("\nKhong the read file stdout.");
        machine->WriteRegister(2, -1);
        IncreasePC();
        return;
    }
    old_pos = fileSystem->oFile[id]->GetCurrentPos(); // Kiem tra thanh cong thi lay vi tri old_pos
    buffer = User2System(virtAddr, charcount); // Copy chuoi tu vung nho User Space sang System Space voi bo dem buffer dai charcount
    // Xet truong hop doc file stdin (type quy uoc la 2)
    if (fileSystem->oFile[id]->type == 2) {
        // Su dung ham Read cua lop SynchConsole de tra ve so byte thuc su doc duoc
        int size = gSynchConsole->Read(buffer, charcount);
        System2User(virtAddr, size, buffer); // Copy chuoi tu vung nho System Space sang User Space voi bo dem buffer co do dai la so byte thuc su
        machine->WriteRegister(2, size); // Tra ve so byte thuc su doc duoc
        delete buffer;
        IncreasePC();
        return;
    }
    // Xet truong hop doc file binh thuong thi tra ve so byte thuc su
    if ((fileSystem->oFile[id]->Read(buffer, charcount)) > 0) {
        // So byte thuc su = new_pos - old_pos
        new_pos = fileSystem->oFile[id]->GetCurrentPos();
        // Copy chuoi tu vung nho System Space sang User Space voi bo dem buffer co do dai la so byte thuc su
        System2User(virtAddr, new_pos - old_pos, buffer);
        machine->WriteRegister(2, new_pos - old_pos);
    } else {
        // Truong hop con lai la doc file co noi dung la NULL tra ve -2
        //printf("\nDoc file rong.");
        machine->WriteRegister(2, -2);
    }
    delete buffer;
    IncreasePC();
    return;
}
```

2.2.5. Write

- Khai báo hàm ở ./userprog/syscall.h:

```
int Write(char *buffer, int charcount, OpenFileId id);
```

- Xử lí SC_Write trong ./userprog/exception.cc:

```
case SC_Write: {
    // Input: buffer(char*), so ky tu(int), id cua file(OpenFileID)
    // Output: -1: Loi, So byte write thuc su: Thanh cong, -2: Thanh cong
    // Cong dung: Ghi file voi tham so la buffer, so ky tu cho phep ghi va id cua file
    int virtAddr = machine->ReadRegister(4);
    int charcount = machine->ReadRegister(5);
    int id = machine->ReadRegister(6);
    int old_pos;
    int new_pos;
    char* buffer;
    // Kiem tra id cua file truyen vao co nam ngoai bang mo ta file khong
    if (id < 0 || id > 14) {
        printf("\nKhong the ghi vi id nam ngoai bang mo ta file.");
        machine->WriteRegister(2, -1);
        IncreasePC();
        return;
    }
    // Kiem tra file co ton tai khong
    if (fileSystem->oFile[id] == NULL) {
        printf("\nKhong the ghi vi file nay khong ton tai.");
        machine->WriteRegister(2, -1);
        IncreasePC();
        return;
    }
    // Xet truong hop ghi file chi doc (type la 1) hoac file stdin (type la 2) thi tra ve -1
    if (fileSystem->oFile[id]->type == 1 || fileSystem->oFile[id]->type == 2) {
        printf("\nKhong the ghi file stdin hoac file chi doc.");
        machine->WriteRegister(2, -1);
        IncreasePC();
        return;
    }
    old_pos = fileSystem->oFile[id]->GetCurrentPos(); // Kiem tra thanh cong thi lay vi tri old_pos
    buffer = User2System(virtAddr, charcount); // Copy chuoi tu vung nho User Space sang System Space voi bo dem buffer chieu dai la charcount
    // Xet truong hop ghi file read & write (type la 0) thi tra ve so byte thuc su
    if (fileSystem->oFile[id]->type == 0) {
        if ((fileSystem->oFile[id]->Write(buffer, charcount)) > 0) {
            // So byte thuc su = new_pos - old_pos
            new_pos = fileSystem->oFile[id]->GetCurrentPos();
            machine->WriteRegister(2, new_pos - old_pos);
            delete buffer;
            IncreasePC();
            return;
        }
    }
    // Xet truong hop con lai ghi file stdout (type la 3)
    if (fileSystem->oFile[id]->type == 3)
    {
        int i = 0;
        // Vong lap de write den khi gap ky tu '\n'
        while (buffer[i] != 0 && buffer[i] != '\n') {
            gSynchConsole->Write(buffer + i, 1);
            i++;
        }
        buffer[i] = '\n';
        gSynchConsole->Write(buffer + i, 1); // Write ky tu '\n'
        machine->WriteRegister(2, i - 1); // Tra ve so byte thuc su write duoc
        delete buffer;
        IncreasePC();
        return;
    }
}
```

2.2.6. Exec

- Khai báo hàm ở ./userprog/syscall.h:

```
SpaceId Exec(char *name);
```

- Cài đặt hàm vừa khai báo ở ./userprog/pcb.cc:

```
int PCB::Exec(char *filename, int id)
{
    // Gọi mutex->P(); để giúp tránh tình trạng nạp 2 tiến trình cùng 1 lúc.
    multex->P();

    // Kiểm tra thread đã khởi tạo thành công chưa, nếu chưa thì báo lỗi là không đủ bộ nhớ, gọi mutex->V() và return.
    this->thread = new Thread(filename);

    if (this->thread == NULL)
    {
        printf("\nPCB::Exec:: Not enough memory..!\n");
        multex->V();
        return -1;
    }

    //  Đặt processID của thread này là id.
    this->thread->processID = id;
    // Đặt parrentID của thread này là processID của thread gọi thực thi Exec
    this->parentID = currentThread->processID;
    // Gọi thực thi Fork(StartProcess_2,id) => Ta cast thread thành kiểu int, sau đó khi xử ký hàm StartProcess ta cast Thread về đúng kiểu của nó.
    this->thread->Fork(StartProcess_2, id);

    multex->V();
    // Trả về id.
    return id;
}
```

- Cài đặt hàm ExecUpdate(char* name) ở ./userprog/ptable.cc:

```
int PTable::ExecUpdate(char *name)
{
    //Gọi mutex->P(); để giúp tránh tình trạng nạp 2 tiến trình cùng 1 lúc.
    bmsem->P();

    // Kiểm tra tính hợp lệ của chương trình “name”.
    // Kiểm tra sự tồn tại của chương trình “name” bằng cách gọi phương thức Open của lớp fileSystem.
    if (name == NULL)
    {
        printf("\nPTable::Exec : Can't not execute name is NULL.\n");
        bmsem->V();
        return -1;
    }
    // So sánh tên chương trình và tên của currentThread để chắc chắn rằng chương trình này không gọi thực thi chính nó.
    if (strcmp(name, "./test/scheduler") == 0 || strcmp(name, currentThread->getName()) == 0)
    {
        printf("\nPTable::Exec : Can't not execute itself.\n");
        bmsem->V();
        return -1;
    }

    // Tìm slot trống trong bảng Ptable.
    int index = this->GetFreeSlot();

    // Check if have free slot
    if (index < 0)
    {
        printf("\nPTable::Exec :There is no free slot.\n");
        bmsem->V();
        return -1;
    }

    //Nếu có slot trống thì khởi tạo một PCB mới với processID chính là index của slot này
    pcb[index] = new PCB(index);
    pcb[index]->SetFileName(name);

    // parrentID là processID của currentThread
    pcb[index]->parentID = currentThread->processID;

    // Gọi thực thi phương thức Exec của lớp PCB.
    int pid = pcb[index]->Exec(name, index);

    // Gọi bmsem->V()
    bmsem->V();
    // Trả về kết quả thực thi của PCB->Exec.
    return pid;
}
```

- Xử lí SC_Exec trong ./userprog/exception.cc:

```
case SC_Exec: {
    // Input: vi tri int
    // Output: tra ve -1 neu that bai, tra ve id cua thread dang chay neu thanh cong
    int virtAddr;
    virtAddr = machine->ReadRegister(4);
    char* name;
    name = User2System(virtAddr, MaxFileLength + 1); // Lay ten chuong trinh, nap vao kernel

    if (name == NULL) {
        DEBUG('a', "\n Not enough memory in System");
        printf("\n Not enough memory in System");
        machine->WriteRegister(2, -1);
        return;
    }
    OpenFile* oFile = fileSystem->Open(name);
    if (oFile == NULL) {
        printf("\nExec:: Can't open this file.");
        machine->WriteRegister(2, -1);
        IncreasePC();
        return;
    }

    delete oFile;

    int id = pTab->ExecUpdate(name);
    machine->WriteRegister(2, id);

    delete[] name;
    IncreasePC();
    return;
}
```

2.2.7. Join

- Khai báo hàm ở ./userprog/syscall.h:

```
int Join(SpaceId id);
```

- Cài đặt hàm JoinWait(), ExitRelease() khai báo ở ./userprog/pcb.cc:

```
void PCB::JoinWait()
{
    //Gọi joinsem->P() để tiến trình chuyển sang trạng thái block và ngừng lại, chờ JoinRelease để thực hiện tiếp.
    joinsem->P();
}
```

```
// Release wating process
void PCB::ExitRelease()
{
    // Gọi exitsem-->V() để giải phóng tiến trình đang chờ.
    exitsem->V();
}
```

- Cài đặt hàm JoinUpdate(char* name) ở ./userprog/ptable.cc:

```
int PTable::JoinUpdate(int id)
{
    // Ta kiểm tra tính hợp lệ của processID id và kiểm tra tiến trình gọi Join có phải là cha của tiến trình
    // có processID là id hay không. Nếu không thỏa, ta báo lỗi hợp lý và trả về -1.
    if (id < 0)
    {
        printf("\nPTable::JoinUpdate : id = %d", id);
        return -1;
    }
    // Check if process running is parent process of process which joins
    if (currentThread->processID != pcb[id]->parentID)
    {
        printf("\nPTable::JoinUpdate Can't join in process which is not it's parent process.\n");
        return -1;
    }

    // Tăng numwait và gọi JoinWait() để chờ tiến trình con thực hiện.
    // Sau khi tiến trình con thực hiện xong, tiến trình đã được giải phóng
    pcb[pcb[id]->parentID]->IncNumWait();

    //pcb[id]->boolBG = 1;

    pcb[id]->JoinWait();

    // Xử lý exitcode.
    int ec = pcb[id]->GetExitCode();
    // ExitRelease() để cho phép tiến trình con thoát.
    pcb[id]->ExitRelease();

    // Successfully
    return ec;
}
```

- Xử lí SC_Join trong ./userprog/exception.cc:

```
case SC_Join: {
    int id = machine->ReadRegister(4);

    int res = pTab->JoinUpdate(id);

    machine->WriteRegister(2, res);
    IncreasePC();
    return;
}
```

2.2.8. Exit

- Khai báo hàm ở ./userprog/syscall.h:

```
void Exit(int exitCode);
```

- Cài đặt hàm JoinRelease(), ExitWait() khai báo ở ./userprog/pcb.cc:

```
// JoinRelease process calling JoinWait
void PCB::JoinRelease()
{
    // Gọi joinsem->V() để giải phóng tiến trình gọi JoinWait().
    joinsem->V();
}

// Let process tranlation to block state
// Waiting for ExitRelease to continue exec
void PCB::ExitWait()
{
    // Gọi exitsem-->V() để tiến trình chuyển sang trạng thái block và ngừng lại, chờ ExitReleaseđể thực hiện tiếp.
    exitsem->P();
}
```

- Cài đặt hàm JoinUpdate(char* name) ở ./userprog/ptable.cc:

```
int PTable::ExitUpdate(int exitcode)
{
    // Nếu tiến trình gọi là main process thì gọi Halt().
    int id = currentThread->processID;
    if (id == 0)
    {

        currentThread->FreeSpace();
        interrupt->Halt();
        return 0;
    }

    if (IsExist(id) == false)
    {
        printf("\nPTable::ExitUpdate: This %d is not exist. Try again?", id);
        return -1;
    }

    // Ngược lại gọi SetExitCode để đặt exitcode cho tiến trình gọi.
    pcb[id]->SetExitCode(exitcode);
    pcb[pcb[id]->parentID]->DecNumWait();

    // Gọi JoinRelease để giải phóng tiến trình cha đang đợi nó(nếu có) và ExitWait() để xin tiến trình cha
    // cho phép thoát.
    pcb[id]->JoinRelease();
    //
    pcb[id]->ExitWait();

    Remove(id);
    return exitcode;
}
```

- Xử lí SC_Exit trong ./userprog/exception.cc:

```
case SC_Exit: {
    int exitStatus = machine->ReadRegister(4);

    if (exitStatus != 0) {
        IncreasePC();
        return;
    }

    int res = pTab->ExitUpdate(exitStatus);

    currentThread->FreeSpace();
    currentThread->Finish();
    IncreasePC();
    return;
}
```

2.2.9. CreateSemaphore

- Khai báo hàm ở ./userprog/syscall.h:

```
int CreateSemaphore(char *name, int semval);
```

- Cài đặt hàm Create(char* name,int init) khai báo ở ./userprog/stable.cc:

```
int STable::Create(char *name, int init)
{

    // Check da ton tai semaphore nay chua?
    for (int i = 0; i < MAX_SEMAPHORE; i++)
    {
        if (bm->Test(i))
        {
            if (strcmp(name, semTab[i]->GetName()) == 0)
            {
                return -1;
            }
        }
    }
    // Tim slot tren bang semTab trong
    int id = this->FindFreeSlot();

    // Neu k tim thay thi tra ve -1
    if (id < 0)
    {
        return -1;
    }

    // Neu tim thay slot trong thi nap Semaphore vao semTab[id]
    this->semTab[id] = new Sem(name, init);
    return 0;
}
```

- Xử lí SC_CreateSemaphore trong ./userprog/exception.cc:

```
case SC_CreateSemaphore: {
    int virtAddr = machine->ReadRegister(4);
    int semval = machine->ReadRegister(5);

    char* name = User2System(virtAddr, MaxFileLength + 1);
    if (name == NULL) {
        DEBUG('a', "\n Not enough memory in System");
        printf("\n Not enough memory in System");
        machine->WriteRegister(2, -1);
        delete[] name;
        IncreasePC();
        return;
    }
    int res = semTab->Create(name, semval);
    if (res == -1) {
        DEBUG('a', "\n Khong the khoi tao semaphore");
        printf("\n Khong the khoi tao semaphore");
        machine->WriteRegister(2, -1);
        delete[] name;
        IncreasePC();
        return;
    }
    delete[] name;
    machine->WriteRegister(2, res);
    IncreasePC();
    return;
}
```

2.2.10.Wait:

- Khai báo hàm ở ./userprog/syscall.h:

```
int Wait(char *name);
```

- `int Signal(char *name);`
- Cài đặt hàm Wait(char* name) khai báo ở ./userprog/stable.cc:

```
int STable::Wait(char *name)
{
    for (int i = 0; i < MAX_SEMAPHORE; i++)
    {
        // Kiem tra o thu i da duoc nap semaphore chua
        if (bm->Test(i))
        {
            // Neu co thi tien hanh so sanh name voi name cua semaphore trong semTab
            if (strcmp(name, semTab[i]->GetName()) == 0)
            {
                // Neu ton tai thi cho semaphore down();
                semTab[i]->wait();
                return 0;
            }
        }
    }
    printf("Khong ton tai semaphore");
    return -1;
}
```

- Xử lí SC_Wait trong ./userprog/exception.cc:

```
case SC_Wait: {
    int virtAddr = machine->ReadRegister(4);

    char* name = User2System(virtAddr, MaxFileLength + 1);
    if (name == NULL) {
        DEBUG('a', "\n Not enough memory in System");
        printf("\n Not enough memory in System");
        machine->WriteRegister(2, -1);
        delete[] name;
        IncreasePC();
        return;
    }

    int res = semTab->Wait(name);

    if (res == -1) {
        DEBUG('a', "\n Khong ton tai ten semaphore nay!");
        printf("\n Khong ton tai ten semaphore nay!");
        machine->WriteRegister(2, -1);
        delete[] name;
        IncreasePC();
        return;
    }

    delete[] name;
    machine->WriteRegister(2, res);
    IncreasePC();
    return;
}
```

2.2.11. Signal:

- Khai báo hàm ở ./userprog/syscall.h:

```
int Signal(char *name);
```

- Cài đặt hàm Signal(char* name) khai báo ở ./userprog/stable.cc:

```
int STable::Signal(char *name)
{
    for (int i = 0; i < MAX_SEMAPHORE; i++)
    {
        // Kiem tra o thu i da duoc nap semaphore chua
        if (bm->Test(i))
        {
            // Neu co thi tien hanh so sanh name voi name cua semaphore trong semTab
            if (strcmp(name, semTab[i]->GetName()) == 0)
            {
                // Neu ton tai thi cho semaphore up();
                semTab[i]->signal();
                return 0;
            }
        }
    }
    printf("Khong ton tai semaphore");
    return -1;
}
```

- Xử lí SC_Signal trong ./userprog/exception.cc:

```
case SC_Signal: {
    int virtAddr = machine->ReadRegister(4);

    char* name = User2System(virtAddr, MaxFileLength + 1);
    if (name == NULL) {
        DEBUG('a', "\n Not enough memory in System");
        printf("\n Not enough memory in System");
        machine->WriteRegister(2, -1);
        delete[] name;
        IncreasePC();
        return;
    }

    int res = semTab->Signal(name);

    if (res == -1) {
        DEBUG('a', "\n Khong ton tai ten semaphore nay!");
        printf("\n Khong ton tai ten semaphore nay!");
        machine->WriteRegister(2, -1);
        delete[] name;
        IncreasePC();
        return;
    }

    delete[] name;
    machine->WriteRegister(2, res);
    IncreasePC();
    return;
}
```

3. Mô tả chương trình người dùng

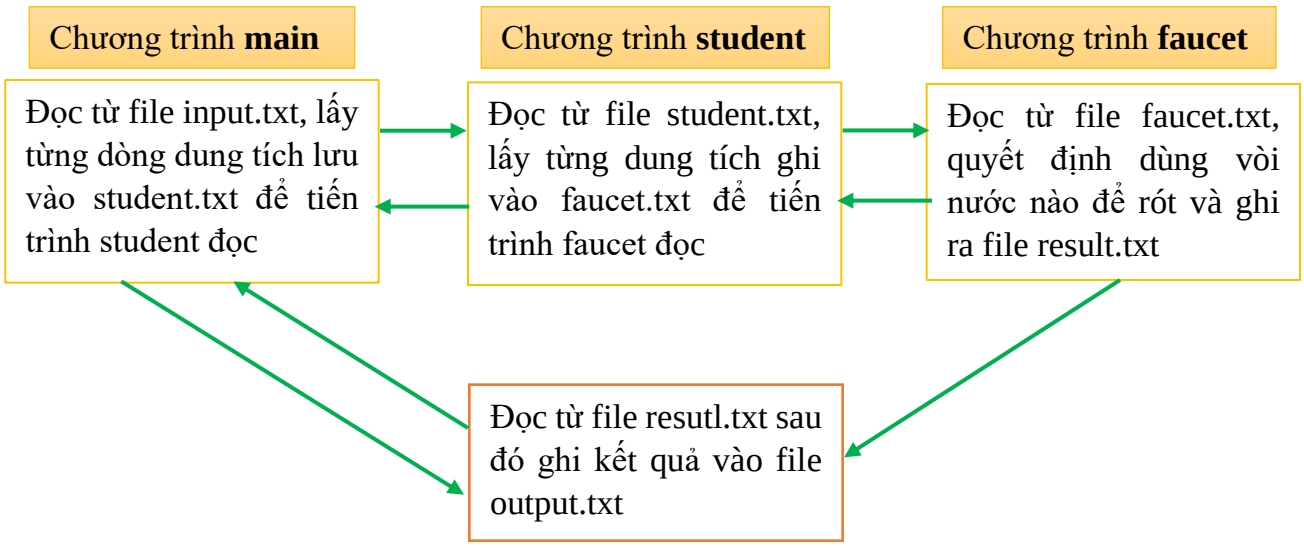
3.1. Các file text dùng để đọc và ghi

- *input.txt:* Dòng đầu là số nguyên dương N cho biết số thời điểm được xét. N dòng tiếp theo, mỗi dòng chứa n số nguyên dương cho biết có n Sinh viên đến uống nước và giá trị của mỗi số nguyên dương cho biết dung tích của chai nước mà sinh viên mang theo.
- *output.txt:* Gồm N dòng, mỗi dòng chứa n cặp số nguyên (a, b). Trong đó, a là dung tích chai nước mà sinh viên mang theo, b đại diện giá trị 1 hoặc 2 cho vòi 1 hoặc vòi 2.
- *student.txt:* chứa hàng đợi các sinh viên có nhu cầu uống nước tại thời điểm i.
- *faucet.txt:* chứa dung tích bình nước của sinh viên đứng ở vị trí j, tại thời điểm i.
- *result.txt:* chứa kết quả xử lí vòi nào rót nước cho các sinh viên có nhu cầu uống nước tại thời điểm i, lần lượt từ 1 đến j.

3.2. Tiến trình

- main.c: chương trình main
- student.c: chương trình xử lý hàng đợi sinh viên
- faucet.c: chương trình xử lý rót nước cho sinh viên đứng ở vị trí j, tại thời điểm i

3.3. Kịch bản giao tiếp giữa các tiến trình



3.4. Chi tiết phần demo:

Demo lập trình NachOS | Phần 2 - YouTube

Tài liệu tham khảo:

1. Tài liệu về project của thầy Lê Viết Long. https://drive.google.com/drive/folders/1aWQMUf8pBRBPd6ije7lP0E8rhJV9bEFO?usp=sharing

2. *Nachos* (2022). Available at: https://homes.cs.washington.edu/~tom/nachos/ (Accessed: 12 December 2022).
3. (2022) *Student.cs.uwaterloo.ca*. Available at: https://student.cs.uwaterloo.ca/~cs350/common/NachosTutorialF06.pdf (Accessed: 12 December 2022).
4. (2022) *Adrien.krahenbuhl.fr*. Available at: http://adrien.krahenbuhl.fr/courses/UnivBordeaux/M1-System/System-PracticalWorks0-FirstSteps.pdf (Accessed: 12 December 2022).
5. (2022) *Tanviramin.com*. Available at: http://tanviramin.com/documents/nachos2.pdf (Accessed: 12 December 2022).
6. *CMPSCI 377 Lab 3* (2022). Available at: https://lass.cs.umass.edu/~shenoy/courses/fall99/labs/lab3.html (Accessed: 12 December 2022).