Trường đại học Khoa Học Tự Nhiên – ĐHQG TP.HCM

Khoa Công nghệ thông tin

fit@hcmus

HỆ ĐIỀU HÀNH

PROJECT 02: ĐA CHƯƠNG

I. DANH SÁCH NHÓM:

Bạch Minh Khôi – 19127181

Lâm Quốc Cường - 19127345

Lê Thành Đạt - 19127354

II. Cài đặt:

1. Các lớp chính:

Lớp Semaphore (./threads/synch.cc):

Là cơ sở trong việc đồng bộ hoá bằng các tăng/giảm giá trị của biến đếm Semaphore.

Semaphore(char * filename, int maxProcess): Phương thức khởi tạo mặc định có tham số truyền vào là initialValue với ý nghĩa là Semaphore này sẽ có tối đa initialValue tiến trình được phép thực thi cùng lúc.

void P(): Giảm biến đếm semaphore xuống, block tiến trình nếu như biến đếm này bằng 0.

void V(): Tăng biến đếm semaphore lên và gọi một tiến trình thực thi nếu tiến trình này đang chờ thực thi từ hàng đợi queue.

Lớp Thread: (./threads/thread.h)

Lớp thread tạo ra các tiểu trình bao gồm việc nạp và cấp phát vùng nhớ Stack, quản lý trạng thái của tiến trình.

Một vài thuộc tính quan trọng:

char* name: Lưu tên của tiến trình, đường dẫn tương đối tới chương trình thực thi.

AddrSpace * space: Vùng nhớ của tiến trình trên ram ảo.

Một vài hàm quan trọng:

void Thread(char * threadName): Khởi tạo một tiến trình với threadName là tên file cần được thực thi (mặc định nằm trong thư mục test). VD: /test/ping.c sẽ có filename là ping.

void Fork(VoidFunctionPtr func, int arg): Cấp phát vùng nhớ Stack cho tiến trình, đưa tiến trình vào danh sách ReadyList và chuyển trạng thái thành READY.

Lớp BitMap: (./lib/bitmap.h)

Lớp này để lưu vết các tiến trình hiện hành. Gồm 1 mảng các flag để đánh dấu các khung trang còn trống trong các page tương ứng ở class AddrSpace.

Các hàm quan trọng cần dùng:

void Mark(int which): Đánh dấu khung trang này được sử dụng.

int Find(): Tìm một khung trang trống và đánh dấu nó đã được sử dụng.

int NumClear(): Trả về tổng số khung trang còn trống trên bộ nhớ.

Lớp PCB: (./userprog/userprog.h)

PCB (Process Control Block): Lưu thông tin để quản lý process. Một số thuộc tính quan trọng:

int pid: id riêng của tiến trình để phân biệt các tiến trình.

Thread* thread: tiến trình được nạp.

int parentID: id của tiến trình cha.

FileName: tên của tiến trình.

3 Semaphore join, exec, mul: Để quản lý quá trình Join, Exit và nạp chương trình.

Lớp PTable: (./userprog/userprog.h)

Dùng để quản lý các tiến trình được chạy trong hệ thống.

PCB* pcb[MAX_PROCESSES] là một mảng mô tả các tiến trình, có số phần tử tối đa MAX_PROCESSES = 10. Mỗi phần tử là một con trỏ tới lớp PCB. Constructor của lớp sẽ khởi tạo tiến trình cha ở vị trí 0 tương đương với phần tử đầu tiên của mảng. Từ tiến trình này, chúng ta sẽ tạo ra các tiến trình con thông qua system call Exec().

2. Chi tiết:

Bước 1: Khai báo các biến toàn cục trong **./threads/main.h** và tạo đối tượng trong **./threads/main.cc.**

main.h

```
extern Kernel *kernel;
extern Debug *debug;
```

main.cc

```
// global variables
Kernel *kernel;
Debug *debug;
```

Bước 2: Cài đặt 2 lớp PCB và PTable và tiến hành khai báo trong build.linux/Makefile.

```
USERPROG_H = ../userprog/addrspace.h\
    ../userprog/syscall.h\
    ../userprog/synchconsole.h\
    ../userprog/noff.h\
    ../userprog/pcb.h\
    ../userprog/ptable.h

USERPROG_C = ../userprog/addrspace.cc\
    ../userprog/exception.cc\
    ../userprog/synchconsole.cc\
    ../userprog/pcb.cc\
    ../userprog/ptable.cc

USERPROG_O = addrspace.o exception.o synchconsole.o pcb.o ptable.o
```

Bước 3: Chỉnh sửa lại class Thread trong ./threads/thread.h

Thêm int processID để quản lý ID phân biệt giữa các tiến trình.

Thêm int exitStatus để kiểm tra exit code của tiến trình.

Bước 4: Cài đặt hàm **StartProcess(int id) (./userprog/exception.cc)** Là hàm dùng để hàm Fork trỏ hàm này đến vùng nhớ của tiến trình con.

Bước 5: Cài đặt lớp **AddrSpace (./userprog/addrspace.cc và addrespace.h)**.
Giải quyết vấn đề cấp phát các frames bộ nhớ vật lý sao cho nhiều chương trình có thể nạp lên bộ nhớ cùng một lúc -> sử dụng biến toàn cục **Bitmap* gPhysPageBitMap (./threads/main.cc)** để quản lý các frames.

Xử lý giải phóng bộ nhớ khi user program kết thúc.

Thay đổi đoạn lệnh nạp user program lên bộ nhớ. Tạo một **pageTable = new TranslationEntry[numPages]**, tìm trang còn trống bằng phương thức **Find()** của lớp Bitmap, sau đó nạp chương trình lên bộ nhớ chính.

Bước 6: Cài đặt system call

System call Exec:

Khai báo trong ./userprog/syscall.h

SpaceID Exec(char *name);

- Cài đặt hàm Exec(char *name, int pid) ở lớp PCB.
- Cài đặt hàm ExecUpdate(char* name) ở lớp PTable.

System call Join:

Khai báo ở ./userprog/syscall.h:

int Join(SpaceID id).

- Cài đặt: JointWait(), ExitRelease() ở lớp PCB.
- Cài đặt: JoinUpdate(int id) ở lớp PTable.

System call Exit:

Khai báo ở ./userprog/syscall.h:

void Exit(int exitCode).

- Cài đặt hàm: JoinRelease(), ExitWait() ở lớp PCB.
- Cài đặt: ExitUpdate(int exitcode) ở lớp PTable.

Bước 7: cài đặt chương trình người dùng: (./test)

ping.c

```c
#include "syscall.h"
int main()
{
    int i;
    for (i = 0; i < 1000; i++)
    {
        PrintString("A");
    }
}
```

pong.c

```c
#include "syscall.h"
int main()
{
    int i;
    for (i = 0; i < 1000; i++)
    {
        PrintString("B");
    }
}
```

scheduler.c

```c
#include "syscall.h"

void main(){
    PrintString("\n____PING PONG___\n");
    Exec("ping");
    Exec("pong");
    PrintString("\n___FINISH___\n");
}
```

Tiến hành khai báo trong (**./test/Makefile**)

```
PROGRAMS = add halt shell matmult sort segments help ascii ping pong scheduler
```

```
scheduler.o: scheduler.c
	$(CC) $(CFLAGS) -c scheduler.c
scheduler: scheduler.o start.o
	$(LD) $(LDFLAGS) start.o scheduler.o -o scheduler.coff
	$(COFF2NOFF) scheduler.coff scheduler

ping.o: ping.c
	$(CC) $(CFLAGS) -c ping.c
ping: ping.o start.o
	$(LD) $(LDFLAGS) start.o ping.o -o ping.coff
	$(COFF2NOFF) ping.coff ping

pong.o: pong.c
	$(CC) $(CFLAGS) -c pong.c
pong: pong.o start.o
	$(LD) $(LDFLAGS) start.o pong.o -o pong.coff
	$(COFF2NOFF) pong.coff pong
```

KẾT QUẢ THU ĐƯỢC:

III. Tài liệu tham khảo:

Các tài liệu được các thầy cung cấp trong quá trình học.
Buổi seminar project 2 của thầy Nguyễn Thanh Quân.
https://github.com/nguyenthanhchungfit/Nachos-Programing-HCMUS
http://blog.terrynini.tw/tw/OS-NachOS-HW1/