Đồ án 1 – Syscall for File System

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
VNUHCM - UNIVERSITY OF SCIENCE

Hệ Điều Hành

Đồ án 2 – Đa chương

BÁO CÁO

Giảng viên

Phạm Tuấn Sơn Lê

Viết Long

Đồ án 2 – Đa chương

Contents

1 THÔNG TIN THÀNH VIÊN

MSSV	Họ tên	Đóng góp
20127577	Phan Nguyễn Phước Nguyên	100%
20127584	Trần Hữu Minh Nhật	100%
20127349	Trần Quốc Thuận	100%

2 CÀI ĐẶT ĐA CHƯƠNG

2.1 Cài đặt đa tiến trình

Nachos hiện tại chỉ là môi trường đơn chương. Để cho mỗi tiến trình được duy trì trong thread của nó ta phải quản lí việc cấp paht1 và thu hồi bộ nhớ, quản lí phần dữ liệu và đồng bộ hóa các tiến trình và tiểu trình

2.2 Giải thích các đối tượng

- **Cơ sở đồng bộ hóa (Semaphore)** (.threads/synch.h)

```
42      Semaphore(char *debugName, int initialValue); // set initial value
```

-Phương thức khởi tạo mặc định có tham số truyền vào là initialValue (giới hạn số tiến trình được phép thực thi cùng lúc)

```
46      void P(); // these are the only operations on a semaphore
47      void V(); // they are both *atomic*
```

-Giảm biến đếm /Tăng biến đếm Semaphore (đồng thời gọi 1 tiến trình thực thi trong hàng đợi)

- **Lớp Thread** (.threads/thread.h)
  -Tạo ra các tiểu trình bao gồm việc nạp và cấp phát vùng nhớ Stack quản lí trạng thái tiến trình

- **Lớp BitMap** (./userprog/bitmap.h)
  -Lưu vết các tiến tirnh2 hiện hành bao gồm 1 mảng cờ hiệu để đánh dấu các khung trang còn trống để nạp vào page tương ứng

- **Lớp PCB** (./userprog)
  -Quản lí process (Process Control Block)

❖ **Lớp Ptable** (./userprog)
-Quản lí các tiến trình được chạy trong hệ thống
-Contructor khởi tạo tiến trình đầu tiên ở vị trí 0 (đầu mảng) và từ vị trí bắt đầu ta tạo ra các tiến trình con thông qua system call Exec (giới hạn các tiến trình con bằng PCB* pcb[MAX_PROCESS])

2.3 Các bước cài đặt chi tiết

❖ Khai báo biến các biến toàn cục trong ./threads/system.h và tạo đối tượng trong system.cc

System.h

```
43  extern Semaphore *addrLock; // semaphore
44  extern BitMap *gPhysPageBitMap; // quan ly cac frame
45  extern PTable *pTab;    // quan ly bang tien trinh
46  extern STable *semTab;    // quan ly semaphore
47  #endif
```

System.cc

```
34  Semaphore *addrLock;  // semaphore
35  BitMap *gPhysPageBitMap;  // quan ly cac frame
36  PTable *pTab;   // quan ly bang tien trinh
37  STable *semTab;   // quan ly semaphore
38  #endif
```

❖ Cài đặt thêm 2 lớp PCB và PTable và khai báo trong file để quản lí tiến trình “Makefile.common” 2 lớp vừa thêm

❖ Điều chỉnh lại số khung trang và kích thước sector

./machine/machine.h

```
#define NumPhysPages 128
#define MemorySize (NumPhysPages * PageSize)
#define TLBSize 4 // if there is a TLB, make it small
```

/machine/disk.h

```
#define SectorSize    512 // number of bytes per disk sector
#define SectorsPerTrack   32  // number of sectors per disk track
#define NumTracks     32  // number of tracks per disk
#define NumSectors    (SectorsPerTrack * NumTracks)
          // total # of sectors per disk
```

❖ Chỉnh sửa lại ./threads/thread.h

```
    //FileTable* mTable; don dep space !=0
    int processID;          //Quan li ID phan biet cac tien trinh
    int exitStatus;         //exit code

    //Giai phong vùng nho tren bo nho ma tien trinh dang dung
    void FreeSpace(){
        if (space != 0)
            delete space;
    }
  // basic thread operations
```

Void **FreeSpace()** được cài đặt để giải phóng vùng nhớ trên bộ nhớ mà tiến trình đang dung

❖ Cài hàm **StartProcess_2(int ID)** trong ./userprog/progtest.cc

Mục đích là được dùng để trỏ đến bởi hàm Fork đến vùng nhớ của tiến trình con

❖ Cài lớp **AddressSpace** trong ./user/addrspace.cc và addrspace.h

- Mục đích làm biến đơn chương trình thành chương trình đa chương nhằm giải quyết vấn đề cấp phát frames bộ nhớ vật lí sao cho nhiều chương trình có thể nạp lên bộ nhớ cùng 1 lúc (sử dụng hàm Bitmap* gPhysPageBitMap)

- Sau đó sử lí giải phóng bộ nhớ khi userprogram kết thúc
- Khi hỗ trợ đa tiến trình bộ nhớ không còn được biểu diễn liên tiếp nữa vì thế tạo 1 pageTable = new TranslationEntry[numPages], và sử dụng Find() để tìm trang còn trống trong lớp Bitmap, cuối cùng mới nạp chương trình lên bộ nhớ chính bằng pageTable[i].physicalPage = gPhysPageBitMap -> Find()

❖ Cài đặt System call (./userprog/syscall.h)

- System call Exec

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

-Hàm được cài đặt ở lớp PCB: Exec(char* name, int pid)

-Hàm được cài đặt ở lớp Ptable: ExecUpdate(char* name)

- System call Join

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

-Hàm được cài đặt ở lớp PCB: JoinWait() ; ExitRelease()

-Hàm được cài đặt ở lớp Ptable: JoinUpdate(int ID)

- System call Exit

```
68   void Exit(int status);
```

-Hàm được cài đặt ở lớp PCB: JoinRelease() ; ExitWait()

-Hàm được cài đặt ở lớp Ptable: ExitUpdate(int exitcode)

3 CHƯƠNG TRÌNH PING NGƯỜI DÙNG

- Chương trình Ping

ping.c

```c
#include "syscall.h"

int main()
{

  int i;
  for(i =0; i< 1000; i++)
  {
    PrintChar('A');
  }

}

```

- Chương trình Pong

pong.c

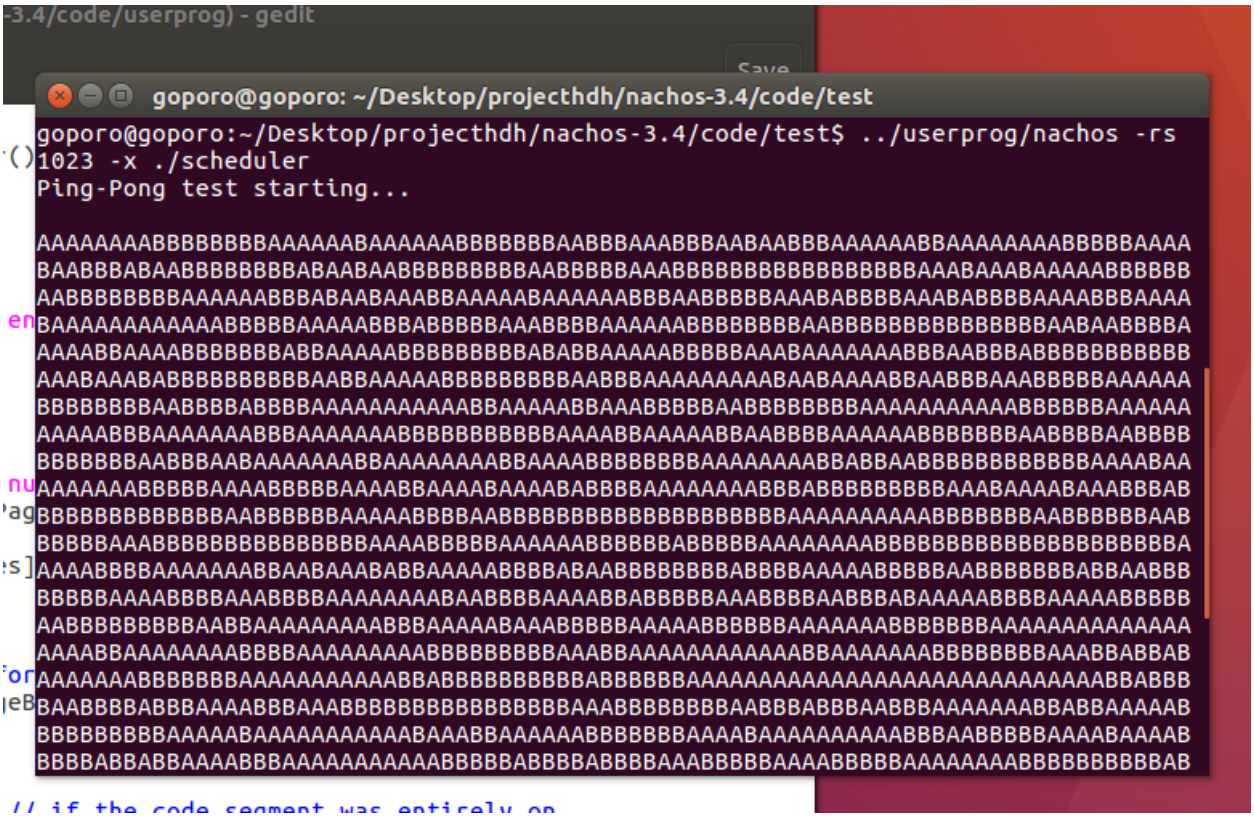
```c
#include "syscall.h"

int main()
{
  int i;
  for(i =0; i< 1000; i++)
  {
    PrintChar('B');
  }
}

```

- Chương trình Scheduler

scheduler.c

```
1
2  #include "syscall.h"
3
4
5  void main()
6  {
7    int pingPID, pongPID;
8    PrintString("Ping-Pong test starting...\n\n");
9
10   pingPID = Exec("./ping");
11   pongPID = Exec("./pong");
12
13   Join(pingPID);
14   Join(pongPID);
15
16
17 }
18
```

- Chạy thử lệnh *./userprog/nachos –rs 1023 –x ./test/scheduler*



4 TÀI LIỆU THAM KHẢO

- File trong tài liệu hướng dẫn thực hành
- [5]Da Chuong Dong Bo Hoa.pdf
- https://github.com/nguyenthanhchungfit/Nachos-Programing-HCMUS