HỆ ĐIỀU HÀNH

ĐỒ ÁN 2

***GVLT**: Trần Trung Dũng*
***TG**: Lê Giang Thanh*
***TG**: Lê Quốc Hòa*

Bộ môn Mạng máy tính
Khoa Công nghệ thông tin
Đại học Khoa học tự nhiên TP HCM

MỤC LỤC

ĐỒ ÁN 2

1 Thông tin nhóm

MSSV	Họ Tên	Email	Điện thoại
1512479	Nguyễn Duy Tâm	ndtam@arduino.vn	01666942492
1512157	Cao Nguyễn Minh Hiếu	hunterhieu97@gmail.com	01652592239

2 Báo cáo đồ án

2.1 Thiết kế và cài đặt phần quản lý file

Để quản lý các thao tác trên tập tin thì hệ điều hành Nachos cung cấp một lớp FileSystem mà đại diện trong hệ thống là biến toàn cục fileSystem. Lớp này cung cấp các phương thức chức năng để thao tác trong kernel space như tạo, mở, xóa tập tin,... Chính vì vậy, để có thể thao tác với tập tin, ta cần tạo lập các syscall trong đó gọi sử dụng biến toàn cục này và nhiệm vụ là phải truyền đầy đủ các tham số và thông tin cần thiết cho phương thức của nó thực hiện.

a) Syscall CreateFile(char * name).

- Để tạo lập một file thông qua syscall CreateFile này ta sử dụng phương thức Create(char * name, int initialSize) của lớp FileSystem (fileSystem→Create(...)) do yêu cầu là tạo file rỗng nên tham số thứ 2 được cho bằng 0 và người dùng cần truyền tham số duy nhất cho syscall là tên của file cần tạo. Tham số tên này tồn tại trong user program tức là nằm trong user space vì vậy để dùng được cho biến tòa cục fileSystem ta cần sao chép nó vào kernel space bằng hàm User2System(...) thông qua địa chỉ logic mà con trỏ char * name nắm giữ. Quá trình tạo tập tin có thể thất bại do một số trường hợp như tên sai do chứa ký tự đặc biệt, thiếu bộ nhớ,... nên syscall cần có giá trị trả về để thông báo tình trạng kết quả như đề đã yêu cầu: 0 – thành công, -1 – thất bại.

b) Syscall OpenFileID Open(char *name, int type) và int Close(OpenFileID id).

- Syscall Open(...) thực hiện mở một file đã tồn tại trên đĩa. Để thực hiện thao tác trên chúng ta sử dụng phương thức Open (fileSyste→Open(char * name)). Tương tự CreateFile() thì syscall Open() cũng cần sao chép chuỗi tên của file từ user space vào kernel space như trình bày ở trên. Khi ta mở một file bằng phương thức này sẽ không phân biệt được file đó ở dạng nào (đọc ghi: 0 hay chỉ đọc: 1) vì nhóm đã thay đổi phương thức Open(...) thêm cho nó một tham số mặc định là type (mặc định bằng 0 để không bị xung đột với các chương trình ngầm của hệ thống Nachos). Mặc khác, phương thức Open trả về một con trỏ kiểu OpenFile *, tức là nó giữ địa chỉ của một biến OpenFile, nên ta cần thay đổi chút ít trong lớp OpenFile để sau này có thể biết nó đang mở ở định dạng nào. Nhóm đã thêm một thuộc tính là type trong lớp Open nhận hai giá trị là 0: đọc ghi và 1: chỉ đọc, tương ứng với tham số type truyền vào.

- Đồ án này thì mỗi tiến trình được cung cấp một bảng mô tả file có thể lưu được đặc tả 10 files. Để đảm bảo điều đó, lớp FileSystem đã được thêm thuộc tính OpenFile * fileTable[10] và int numFile. fileTable là mảng 10 phần tử với mỗi phần tử có kiểu là OpenFile * tức là nó trỏ tới vùng nhớ chứa đặc tả của 1 file đã được mở. Biến numFile cho biết số file đã mở và được chứa trong fileTable. Riêng hai file đặc biệt là ConsoleInput và ConsoleOutput được mặc định là hai phần tử thứ 0 và 1 trong fileTable (hai phần tử

này có giá trị bằng NULL) khi cần thao tác với hai file này thay vì dùng fileSystem như các file khác trên đĩa, ta dùng biến gSynchConsole để thao tác. Biến numFile có chức năng đếm số file đã mở và chứa trong bảng fileTable, nó sẽ được tăng khi mở thêm và giảm khi đóng file, cũng như kiểm tra đầy bảng fileTable để xử lý. Ngoài ra kiểu OpenFileID thực chất là kiểu int, nó mang ý nghĩa là thứ tự (chỉ số) của file đã mở trong bảng fileTable.

- Syscall Close(OpenFileID id) sẽ thực hiện thao tác xóa địa chỉ vùng nhớ của fileTable[id] nắm giữ (nếu tồn tại) và xóa phần tử fileTable[id] (set bằng NULL), giảm biến đếm numFile xuống.

- Ngoài ra, hai syscall này cũng có một số trường hợp lỗi như tên file không tồn tại, đóng file không được mở (con trỏ mang giá trị NULL),... sẽ được kiểm tra và xử lý để trả về giá trị thông báo cho người dùng: trả về OpenFileID id là thứ tự của nó trong bảng fileTable nếu thành công, -1 nếu mở không được hay 0 nếu đóng thành công và -1 cho trường hợp thất bại.

c) system call int Read(char *buffer, int charcount, OpenFileID id) và int Write(char *buffer, int charcount, OpenFileID id)

- Trong trường hợp này cần phân biệt rõ khi nào là thao tác trên file khi nào trên Console thông qua id của file cần đọc ghi (0 = ConsoleInput, 1 = ConsoleOutput). Nếu là ConsoleInput hay ConsoleOutput ta sẽ dùng biến toàn cục gSynchConsole để đọc ghi từ Console như đã từng làm trong đồ án trước. Khi file này là file thực sự lưu trên đĩa thì ta sẽ dùng phương thức đoc ghi của lớp OpenFile để thao tác: OpenFile::Read(...), OpenFile::ReadAt(...), OpenFile::Write(...), OpenFile::WriteAt(...).

- Một số chú ý cho hai syscall này là:

  - Phải kiểm tra xem file mở lên có đúng định dạng cần thao tác không: sử dụng thuộc tính type đã thêm vào lớp OpenFile đã nêu ở trên, ví dụ mở chỉ đọc thì không thể cho ghi. Với syscall Read thì ta có thể lướt qua bước này bởi đồ án yêu cầu thì 0: đọc – ghi, 1: chỉ đọc. Nhưng với syscall Write thì phải kiểm tra kỹ lưỡng việc này.
  - Đối với Console hay file cũng vậy, số ký tự yêu cầu đọc ghi và số ký tự đọc ghi được thật sự không phải bao giờ cũng giống nhau vì vậy phải trả về đúng số lượng thực tế và chính xác cho chương trình người dùng. Các phương thức đọc ghi của gSynchConsole và OpenFile đều trả về số byte thật sự nên ta tận dụng được điều này để xử lý.
  - Khi ghi thì chuỗi ký tự đang ở trong user program thuộc user space để các hàm, phương thức của lớp trong kernel space thao tác được cần sao chép vào vùng nhớ hệ điều hành (sử dụng User2System).
  - Tương ứng khi đọc được một chuỗi ký tự thì nó đang nằm trong kernel space muốn chương trình người dùng sử dụng được thì cần sao chép nó trả ra user space (sử dụng System2User).

- Giá trị trả về của hai syscall này ngoài số byte đọc ghi được, đôi khi còn là giá trị thể hiện thao tác bị lỗi do một số sai sót hay ngoại lệ như gặp ký tự kết thúc file khi đọc, type của file sai đinh dạng, file không tồn tại,…

- Ngoài ra phần đọc ghi này, sẽ thực hiện tại đọc ghi ngầm định tại vị trí hiện thời của con trỏ trong file được mở, chứ không phải đọc ghi từ đầu file.

d) system call int Seek (int pos, OpenFileID id)

- Do lớp OpenFile có hai phần khai báo và cài đặt như là hai lớp riêng, tùy vào chỉ thị tiền xử lý mà sẽ biên dịch phần khai báo và cài đặt tương ứng. Do đó để hỗ trợ cho syscall Seek, nhóm đã chỉnh sửa chút ít trong cải hai lớp kia. Đó là thêm hàm OpenFile::GetCurrentPos() để lấy vị trí hiện tại của con trỏ và OpenFile::Seek(…) cho lớp chưa có. Hàm OpenFile::Seek(…) cũng được sửa lại giá trị trả về là int cho biết vị trí sau khi dịch, điều này rất hữu ích cho chương trình người dùng.

- Syscall này có một số trường hợp cần xử lý thêm ngoài các trường hợp ngoại lệ file không tồn tại,…. như dịch về -1 là xuống cuối file, dịch về pos < -1 sẽ cho về 0 hay dịch quá kích thước file sẽ cho về cuối file.

e) Các chương trình demo

- Các chương trình mà đồ án yêu cầu hầu hết chỉ cần thao tác bằng cách gọi syscall tương ứng và thuật toán đơn giản. Phần này được thể hiễn rõ trong mã nguồn ngắn gọn của các chương trình đó và không có gì đặc sắc nên nhóm không trình bày chi tiết phần này.