**Mô tả cài đặt các hàm:**

**1. Cài đặt “exception.cc”**

**-** Bên trong thư mục .code/machine, file “machine.h” có chứa danh sách các exception. Sau đó, qua file “exception.cc” trong thư mục .code/userprog, ta viết các case trong hàm ExceptionType, cuối mỗi case sử dụng halt() để kết thúc chương trình.

**2. IncreasePC():**

**-** Hàm được viết trong file “exception.cc” trong thư mục userprog.

- Bắt đầu với việc đọc giá trị hiện tại của thanh ghi bằng cách sử dụng hàm ReadRegister của machine. Giá trị này là địa chỉ của lệnh hiện tại trong máy được lưu vào biến “counter” . Lưu giá trị này vào thanh ghi trước đó (PrevPCReg) bằng cách dùng hàm WriteRegister của machine, truyền vào 2 biến PrevPCReg và “counter”. Đọc giá trị của thanh ghi tiếp theo (PCNextReg), giá trị này là địa chỉ của lệnh tiếp theo sẽ được thực thi trong máy. Tiếp theo, ghi giá trị của PC hiện tại vào thanh ghi PC bằng hàm WriteRegister, điều này đảm bảo rằng PC trỏ đến lệnh tiếp theo sẽ được thực thi khi hàm này kết thúc. Cuối cùng, cập nhật thanh ghi tiếp theo của PC để trỏ đến lệnh kế tiếp (PC+4). Mỗi lệnh trong máy thường có độ dài 4 byte nên để thực hiện lệnh tiếp theo, giá trị của PC cần được tăng lên 4.

**3. Cài đặt syscall SC\_Create: int Create(char \*name);**

**-** CreateFile system call sẽ sử dụng Nachos FileSystem Object để tạo một file rỗng. System call CreateFile trả về 0 nếu thành công và -1 nếu có lỗi**.**

**-** Input: tên file

- Output: 0: thành công và -1: lỗi

**-** Các bước thực hiện:

1. Đọc địa chỉ ảo của tên file từ thanh ghi 4: đầu tiên, hàm đọc địa chỉ ảo của tên file từ thanh ghi 4 của máy. Địa chỉ này là địa chỉ ảo của vùng nhớ mà tên file được lưu trữ.

2. Chuyển đổi địa chỉ ảo sang địa chỉ hệ thống và lấy tên file: Sử dụng hàm “User2System” để chuyển đổi và lấy tên file.

3. Kiểm tra tính hợp lệ của tên file: kiểm tra tên file có rỗng không bằng cách sử dụng hàm “strlen”. Nếu tên file rỗng, hàm in ra thông báo lỗi, trả về giá trị thanh ghi 2 là -1 và tăng giá trị counter để chuyển đến lệnh tiếp theo.

4. Kiểm tra khả năng tạo file trong hệ thống: kiểm tra xem hệ thống có đủ bộ nhớ để tạo file không. Nếu không đủ bộ nhớ, hàm in ra thông báo lỗi, đật giá trị thanh ghi 2 là -1, giải phóng bộ nhớ đã cấp phát và tăng giá trị counter để chuyển đến lệnh tiếp theo.

5. Hàm gọi phương thức “Create” của fileSystem để tạo file với tên đã được nhận từ bước trước. Nếu việc tạo file thành công, hàm đặt giá trị trả về cho thanh ghi 2 là 0, giải phóng bộ nhớ đã cấp phát và tăng giá trị counter để chuyển đến lệnh tiếp theo.

6. Cuối cùng, hàm giải phóng bộ nhớ đã cấp phát cho tên file và tăng giá trị counter.

**3. Cài đặt SC\_Open:**  **OpenFileId Open(char \*name, int type);**

-Input: tên file trên user space và type (0: read và write, 1: read only).

- Output: Id của file nếu thành công và -1 nếu thất bại.

- Các bước thực hiện:

1. Đọc địa chỉ ảo của tên file từ thanh ghi số 4 của máy. Địa chỉ này là địa chỉ ảo của vùng nhớ mà tên file được lưu trữ.

2. Sau đó, sẽ đọc type từ thanh ghi số 5 của máy.

3. Kiểm tra xem số lượng file đã mở đạt tối đa hay chưa (tối đa là 10). Nếu đã tối đa, hàm đặt giá trị trả về thanh ghi 2 là -1, điều này có nghĩa là mở file thất bại.

4. Dùng hàm “User2System” để chuyển đổi địa chỉ ảo sang địa chỉ hệ thống và lấy tên file từ địa chỉ đó.

5. Kiểm tra xem tên file có phải là “stdin” hoặc “stdout” không. Nếu là 1 trong 2 thì in ra thông báo và đặt giá trị trả về thanh ghi 2 là 1 (“stdin”) hoặc 0 (“stdout”) để thông báo việc mở file thành công.

6. Gọi phương thức “Open” của fileSystem để mở file với tên file đã được nhận từ các bước trên. Nếu mở file thành công, hàm in ra thông báo và đặt giá trị trả về thanh ghi 2 là chỉ số của file đã mở trong mảng “openfile”. Nếu không, hàm in thông báo lỗi và đặt giá trị thanh ghi 2 là -1.

7. Cuối cùng, giải phóng bộ nhớ đã cấp phát cho tên file và tăng giá trị của counter.

**4. Cài đặt SC\_Close: int CloseFile(OpenFileId id);  
-** Input: ID của file

- Output: NULL

- Các bước thực hiện:

1. Đọc giá trị file descriptor từ thanh ghi 4 của máy. File descriptor này là chỉ số của file trong mảng “openfile” mà muốn đóng.

2. Kiểm tra xem file descriptor đọc được có hợp lệ hay không bằng cách so sánh với số lượng file đã mở trong hệ thống. Nếu file descriptor lớn hơn số lượng file đã mở, điều này chỉ ra rằng file không tồn tại hoặc không hợp lệ và in ra thông báo lỗi. Sau đó, đặt giá trị trả về cho thanh ghi 2 là -1.

3. Nếu file descriptor hợp lệ, tiến hành đóng file bằng cách gán “NULL” cho vị trí tương ứng trong mảng “openfile”. Sau đó, giải phóng bộ nhớ đã cấp phát cho file.

4. Cuối cùng, đặt giá trị trả về của thanh ghi 2 là 0, tăng program counter.

**5. Cài đặt SC\_Read: int Read(char \*buffer, int size, OpenFileId id);**

- Input: buffer, số byte cần đọc, ID của file.

- Output: -1 nếu thất bại, số byte đã đọc nếu thành công.

- Các bước thực hiện:

1. Đọc gái trị địa chỉ ảo của bộ đệm từ thanh ghi 4 của máy, đọc số lượng ký tự cần đọc từ thanh ghi 5 của máy, đây là số lượng tối đa các ký tự sẽ được đọc và sao chép vào bộ đệm.

2. Đọc giá trị file descriptor từ thanh ghi 6 của máy. File descriptor là chỉ số của file trong mảng “openfile” muốn đọc.

3. Kiểm tra xem file descriptor có hợp lệ hay không bằng cách so sánh với số lượng file đã mở trong hệ thống và kiểm tra xem có phải là “stdout” hay không. Nếu fiel descriptor không hợp lệ hoặc đang cố gắng đọc từ stdout, hàm sẽ in ra thông báo lỗi, trả giá trị thanh ghi số 2 là -1.

4. Tiếp theo, kiểm tra xem file có mở được hay không bằng cách kiểm tra vị trí tương ứng trong mảng “openfile” có trỏ đển một đối tượng file hay không. Nếu không, trả giá trị thanh ghi 2 là -1 và kết thúc hàm.

5. Cấp phát một bộ nhớ đệm trong kernel với kích thước là số lượng ký tự cần đọc.

6. Nếu file đang đọc là stdin, sử dùng hàm “Read” của “gSynchConsole” để đọc dữ liệu từ bàn phím và sao chép vào bộ đệm. Nếu là file khác, hàm sử dụng phương thức “Read” của file để đọc dữ liệu và sao chép vào bộ đệm.

7. Sau khi đọc dữ liệu, hàm sử dụng “System2User” để sao chép dữ liệu từ bộ đệm trong không gian kernel sang bộ đệm trong không gian người dùng.

8. Nếu việc đọc dữ liệu thành công, đặt giá trị trả về thanh ghi 2 là số lượng ký tự đã đọc và tăng giá trị này thêm 1 để đánh dấu vị trí kết thúc chuỗi. Nếu không, đặt giá trị của thanh ghi 2 là -1 chỉ ra đọc dữ liệu thất bại. Sau đó, tăng program counter.

**9. Cài đặt SC\_Write: int Write(char \*buffer, int size, OpenFileId id);**

- Input: buffer, số lượng kí tự cần ghi, ID của file.

- Output: -1 nếu thất bại, số byte đã đọc nếu thành công.

- Các bước thực hiện:

1. Đọc gái trị địa chỉ ảo của bộ đệm từ thanh ghi 4 của máy, đọc số lượng ký tự cần đọc từ thanh ghi 5 của máy, đây là số lượng tối đa các ký tự sẽ được đọc và sao chép vào bộ đệm.

2. Đọc giá trị file descriptor từ thanh ghi 6 của máy. File descriptor là chỉ số của file trong mảng “openfile” muốn đọc.

3. Kiểm tra xem file descriptor có hợp lệ hay không bằng cách so sánh với số lượng file đã mở trong hệ thống và kiểm tra xem có phải là “stdin” hay không. Nếu fiel descriptor không hợp lệ hoặc đang cố gắng ghi vào stdin, hàm sẽ in ra thông báo lỗi, trả giá trị thanh ghi số 2 là -1.

4. Tiếp theo, kiểm tra xem file có mở được hay không bằng cách kiểm tra vị trí tương ứng trong mảng “openfile” có trỏ đển một đối tượng file hay không. Nếu không, trả giá trị thanh ghi 2 là -1 và kết thúc hàm.

5. Kiểm tra xem file có phải là file chỉ đọc không bằng cách kiểm tra thuộc tính “type” trong đối tượng file. Nếu file là file chỉ đọc, in ra thông báo lỗi, đặt giá trị trả về thanh ghi 2 là -1, tăng program counter, kết thúc hàm.

6. Nếu file đang được ghi là stdout, sử dùng hàm “Write” của “gSynchConsole” để ghi dữ liệu vào stdout. Nếu là file khác thì dùng phương thức “Write” của file để ghi dữ liệu vào file.

7. Nếu ghi dữ liệu thành công, sao chép dữ liệu từ bộ đệm trong không gian kernel sang bộ đệm trong không gian người dùng.

8. Nếu việc đọc dữ liệu thành công, đặt giá trị trả về thanh ghi 2 là số lượng ký tự đã ghi, tăng program counter. Nếu không thì không thay đổi giá trị thanh ghi 2 và kết thúc hàm.

**10. Cài đặt SC\_Seek: int Seek(int pos, OpenFileId id);**

**-** Input: vị trí cần chỉ tới, ID của file.

- Output: -1 nếu thất bại, vị trí thực sự trong file nếu thành công.

- Các bước thực hiện:

1. Đọc giá trị vị trí cần di chuyển đến từ thanh ghi 4, đại diện cho vị trí mới mà con trỏ sẽ được di chuyển tới.

2. Đọc giá trị file descriptor từ thanh ghi 55 của máy. File descriptor này là chỉ số của file trong mảng “openfile”.

3. Kiểm tra xem file descriptor có hợp lệ không bằng cách so sánh với số lượng file đã mở trong hệ thống và kiểm tra xem vị trí tương ứng trong mảng “openfile” có trỏ đến một đối tượng file không. Nếu không hợp lệ, đặt giá trị trả về của thanh ghi 2 là -1 và kết thúc hàm.

4. Lấy độ dài của tệp mà con trỏ muốn di chuyển đến. Độ dài này được sử dụng để kiểm tra tính hợp lệ của vị trí mới được chỉ định.

5. Kiểm tra xem vị trí mới được chỉ định có hợp lệ không, tức là không lớn hơn độ dài của tệp. Nếu không hợp lệ, đặt giá trị trả về của thanh ghi 2 là -1 và kết thúc hàm.

6. Nếu vị trí mới là hợp lệ, hàm này sẽ di chuyển con trỏ tệp đến vị trí mới bằng cách sử dụng phương thức **“**Seek” của fileSystem.

7. Ghi vị trí mới của con trỏ tệp vào thanh ghi 2, tăng program counter.

**11. Cài đặt SC\_ReadChar**

**-** Các bước thực hiện:

1. Xác định số lượng kí tự tối đa mà có thể đọc được từ bàn phím, trong trường hợp này, số lượng kí tự tối đa được đặt là 255.

2. Tạo một buffer có kích thước 255 để lưu trữ dữ liệu đọc từ console, sau đó sử dụng phương thức “Read” của đối tượng “gSynchConsole” để đọc dữ liệu từ bàn phím và lưu trữ vào buffer.

3. Nếu số lượng ký tự đọc được lớn hơn 1, in ra thông báo lỗi cho người dùng và ghi thông điệp lỗi vào debug. Sau đó, đặt giá trị trả về của thanh ghi 2 là 0 để chỉ ra rằng có lỗi xày ra. Nếu số lượng ký tự đọc được là 0, in ra thông báo lỗi cho người dùng và ghi lỗi vào debug. Đặt giá trị trả về của thanh ghi 0 để chỉ ra có lỗi xảy ra. Nếu số lượng kí tự đọc được là 1, lấy kí tự đầu tiên từ buffer và ghi vào thanh ghi 2 của máy.

4. Sau khi đọc xong, giải phóng bộ nhớ của buffer để tránh leaking và tăng program counter.

**12. Cài đặt SC\_PrintChar**

- Các bước thực hiện:

1. Đọc kí tự từ thanh ghi 4 của máy vào biến “c”.

2. Sau khi đọc kí tự từ thanh ghi, dùng phương thức “Write” của đối tượng “gSynchConsole” để ghi kí tự đó ra màn hình. Kí tự được ghi bằng cách truyền địa chỉ của “c” và số lượng kí tự cần ghi, trong trường hợp này là 1.

3. Cuối cùng, tăng giá trị program counter.

**Define trong “syscall.h”**

* Trong file “syscall.h”, các syscall được define như sau: SC\_ReadChar và SC\_PrintChar là 10 và 11, SC\_ReadInt và SC\_PrintInt là 15 và 16, SC\_ReadString và SC\_PrintString là 17 và 18, SC\_ReadFloat và SC\_PrintFloat là 19 và 20.

**Khai báo biến toàn cục trong “.code/threads” 2 file “system.h” và “system.cc”**

* Trong file “system.h”, khai báo thư viện “synchcons.h” và một biến con trỏ gSynchConsole để dùng cho việc read và write ở console
* Trong file “system.cc”, cũng khai bào con trỏ gSynchConsole từ SynchConsole, cài đặt tạo mới và giải phóng bộ nhớ.

**Thay đổi trong .code/filesys/filesys.h**

* Trong lớp FileSystem,
  + khai báo “openfile” là một mảng con trỏ đến các đối tượng “OpenFile”, được sử dụng để theo dõi các file đang mở trong file system. Mỗi đối tượng “OpenFile” đại diện cho một file đang mở và cung cấp các phương thức để thực hiện thao tác đọc ghi, di chuyển trong file.
  + Khai báo biến “index” để theo dõi số lượng các file đang mở trong file system. Nó chỉ ra vị trí kế tiếp trong mảng “openfile” mà một file mới sẽ được thêm vào.
* Ở các hàm Open, tăng giá trị index mỗi khi kết thúc hàm để chỉ ra vị trí kế tiếp trong mảng “openfile”.
* Khai báo và cài đặt hàm open file với type chỉ định.

**Cài đặt chương trình sắp xếp mảng các số nguyên dùng thuật toán Quick Sort**

* Viết chương trình cài đặt trong file “quicksort.c” trong thư mục .code/test.
* Chương trình yêu cầu người dùng nhập số lượng phần tử của mảng.
* Nhập các phần tử của mảng tuần tự.
* Sau khi nhập mảng thì gọi hàm “quickSort” để sắp xếp các phần tử trong mảng.
* Một file tên “quicksort.txt” được mở để ghi kết quả vào.
* Nếu không mở được thì thông báo lỗi và kết thúc.
* Các phần tử trong mảng được chuyển thành chuỗi kí tự bằng hàm “intToString” và sau đó ghi vào file “quicksort.txt”, cách nhau bởi dấu cách.
* Sau khi ghi xong, file “quicksort.txt” được đóng.
* Chương trình in ra màn hình một thông báo cho người dùng biết rằng kết quả đã được ghi vào file “quicksort.txt”.
* Cuối cùng, kết thúc chương trình bằng Halt().