**Hệ điều hành – NachOS**

**Thành viên:**

**Các system call:**

1. **Halt ():**

* Sử dụng hàm *Halt()* của lớp Interrupt để tắt NachOS và in ra các thông số.

1. **ReadInt ():**

* Một số interger chỉ có thể có tối đa 10 số nên tạo mảng ký tự *buffer* với 12 ký tự để nhận thông tin đọc từ màn hình Console ( ký tự đầu cho ‘-‘ (nếu là số âm), 10 ký tự giữa cho chữ số, ký tự cuối cho ‘\0’ ).
* Đọc số integer cần đọc từ màn hình Console và lưu từng ký tự vào mảng *buffer*.
* Đọc mảng tuần tự để tạo ra một số interger để lưu vào thanh ghi số 2.
* Nếu có ký tự nào không hợp lệ trong chuỗi -> không phải interger -> lưu 0 vào thanh ghi số 2. Ngược lại -> đây là integer -> lưu số integer vào thanh ghi số 2.
* Trả về số integer.
* Tăng PC lên.

1. **PrintInt (int number):**

* Đọc thanh ghi số 4 để lấy giá trị cần in.
* Một số interger chỉ có thể có tối đa 10 số nên tạo mảng ký tự *buffer, rev* với 12 ký tự để in thông tin ra màn hình Console ( ký tự đầu cho ‘-‘ (nếu là số âm), 10 ký tự giữa cho chữ số, ký tự cuối cho ‘\0’ ).
* Lưu từng chữ số của số integer cần in từ phải sang trái vào mảng *rev.*
* Sau đó, lưu ngược lại từng ký tự trong *rev* vào mảng *buffer* theo thứ tự từ trái sang phải để có thứ tự đúng.
* In các ký tự của mảng buffer ra màn hình Console.
* Tăng PC lên

1. **ReadFloat ():**

* Đọc từ màn hình Console để lấy được số floating point cần đọc và lưu từng ký tự vào một mảng ký tự.
* Kiểm tra xem số này là dương hay âm.
* Đọc mảng tuần tự để tạo ra một số floating point hợp lệ.
* Nếu chuỗi không hợp lệ -> không phải số floating point -> lưu 0.0 vào thanh ghi số 2. Ngược lại -> đây là số floating point -> lưu số vào thanh ghi số 2.
* Do thanh ghi số 2 chỉ nhận số integer nên ta lấy dạng thể hiện nhị phân của số floating point chuyển thành một số ingeter có dạng thể hiện nhị phân tương ứng để lưu vào thanh ghi số 2.
* Trả về số floating point.
* Tăng PC lên.

1. **PrintFloat (float number):**

* Đọc từ thanh ghi số 4 để lấy số floating point cần in và thanh ghi chỉ chứa số integer nên ta chuyển số integer lấy được thành floating point với cùng dạng biểu diễn nhị phân.
* Tạo một mảng ký tự *buffer* để lưu từng ký tự trong số floating point vào để in ra màn hình Console.
* Lấy phần nguyên của số floating point và kiểm tra xem phần nguyên là số âm, số dương hay số 0.
* Nếu là số âm -> chuyển dấu cho thành số dương, vị trí đầu của *buffer* là ‘-‘. Nếu là số 0 -> vị trí đầu tiên của *buffer* là số 0.
* Lưu từng chữ số của phần nguyên vào mảng ký *rev* theo thứ tự từ phải sang trái.
* Lưu từng ký tự trong mảng *rev* ngược lại vào mảng *buffer* theo thứ tự từ trái sang phải để đúng vị trị sau đó là ‘.’ và lưu từng ký tự của phần thập phân vào *buffer* theo thứ tự từ trái sang phải.
* In mảng *buffer* ra màn hình Console.
* Tăng PC lên.

1. **ReadChar ():**

* Đọc 1 ký tự người dùng nhập vào từ màn hình Console.
* Chuyển ký tự từ dạng integer để lưu vào thanh ghi số 2.
* Trả về ký tự.
* Tăng PC lên.

1. **PrintChar (char character):**

* Đọc thanh ghi số 4 để lấy một số integer và chuyển số integer này về dạng character.
* In ký tự tìm được ra màn hình Console.
* Tăng PC lên.

1. **ReadString (char\* buffer, int length):**

* Đọc từ thanh ghi số 4 để lấy địa chỉ lưu chuỗi sẽ đọc trong userspace.
* Đọc từ thanh ghi số 5 để lấy độ dài chuỗi mong muốn.
* Chuyển địa chỉ cùng độ dài tìm được qua kernalspace thông qua hàm User2System.
* Đọc từ màn hình Console và lưu từng ký tự đọc được vào chuỗi trong kernelspace.
* Chuyển chuỗi từ trong kernelspace về lại userspace cho người dùng.
* Trả về chuỗi.
* Tăng PC lên.

1. **PrintString (char\* buffer):**

* Đọc từ thanh ghi số 4 để lấy địa chỉ lưu chuỗi cần in ra màn hình trong userspace.
* Chuyển chuỗi từ userspace qua kernelspace.
* In chuỗi từ trong kernelspace ra màn hình Console.
* Tăng PC lên.

1. **CreateFile (char\* name):**

* Đọc từ thanh ghi số 4 để lấy địa chỉ của mảng chứa tên file trong vùng nhớ userspace.
* Chuyển mảng từ vùng nhớ userspace qua kernelspace bằng hàm User2System.
* Nếu chuyển thất bại thì lưu -1 vào thanh ghi số 2 và tăng PC lên.
* Dùng đối tượng fileSystem của lớp OpenFile để tạo ra file mới với tên là các ký tự trong mảng vừa chuyển qua kernelspace.
* Nếu tạo file thất bại thì lưu -1 vào thanh ghi số 2 vào tăng PC lên
* Tạo file thành công thì lưu 0 vào thanh ghi số 2 và tăng PC lên.

1. **Open (char\* name, int type):**

* Đọc thanh ghi số 4 để lấy vị trí của chuỗi chứa tên file cần mở trong vùng nhớ userspace.
* Đọc thanh ghi số 5 để lấy thể loại file cần mở (cho phép đọc và ghi, chỉ đọc) và lưu vào biến đếm type.
* Chuyển chuỗi chứa tên file từ trong vùng nhớ userspace qua vùng nhớ kernelspace.
* Dùng hàm FindFreeSpace() của lớp FileSystem để tìm vùng nhớ trống trong bảng mô tả để mở file mới.
* Nếu không còn chỗ trống để mở file mới thì lưu -1 vào thanh ghi số 2 và tăng PC lên rồi trả về.
* Kiểm tra biến đếm type:

+ Nếu type là 0, 1 mở file tại vị trí vùng nhớ trống tìm được và lưu vị trí vùng nhớ vào thanh ghi số 2, lưu -1 nếu không mở được.

+ Nếu type là 2, 3 mở Console Input hoặc Console Output và lưu 0 vào thanh ghi số 2 nếu là Console Input, lưu 1 nếu là Console Output.

+ Nếu là giá trị khác thì giá trị không hợp lệ và lưu -1 vào thanh ghi số 2.

* Tăng PC lên.

1. **Close (OpenFileID id):**

* Đọc thanh ghi số 4 để lấy để lấy ID của của file muốn đóng dưới dạng số nguyên.
* Bảng mô tả file chỉ lưu được 10 file nên nếu ID < 0 hoặc ID > 9 thì lưu -1 vào thanh ghi số 2, tăng PC và trả về.
* Dùng mảng openfile[int id] của lớp FileSystem để kiểm tra xem file với ID này đã được đóng chưa hoặc có tồn tại hay không.
* Nếu chưa đóng thì tiến hành đóng file lại và lưu 0 vào thanh ghi số 2, ngược lại lưu -1 vào thanh ghi số 2. Sau cùng, tăng PC lên và trả về.

1. **Read (char\* buffer, int charcount, OpenFileID id):**

* Đọc thanh ghi số 4 để lấy vị trí của mảng ký tự *buffer* cần đọc ký tự từ file vào trong vùng nhớ userspace.
* Đọc thanh ghi số 5 để lấy số lượng ký tự muốn đọc từ file.
* Đọc thanh ghi số 6 để lấy ID của file cần đọc.
* Kiểm tra ID, nếu ID không hợp lệ thì lưu -1 vào thanh ghi số, tăng PC lên và trả về.
* Kiểm tra file với ID hợp lệ, nếu file đã đóng hoặc không tồn tại thì lưu -1 vào thanh ghi số 2, tăng PC lên và trả về.
* Nếu người dùng muốn đọc thông tin từ file Console Output (type = 3) thì lưu -1 vào thanh ghi số 2, tăng PC lên và trả về bởi vì không thì đọc thông tin từ file này.
* Chuyển mảng ký tự *buffer* từ vùng nhớ userspace qua vùng nhớ kernelspace.
* Đọc số lượng ký tự được yêu cầu từ file vào mảng ký tự đang ở trong kernelspace và chuyển mảng ký tự về lại vùng nhớ userspace.
* Lưu số lượng ký tự đọc được vào thanh ghi số 2, nếu ở cuối file thì -2 vào thanh ghi số 2.
* Tăng PC lên.

1. **Write (char\* buffer, int charcount, OpenFileID id):**

* Đọc thanh số 4 để lấy vị trí của mảng ký tự muốn viết ký tự vào file với ID hợp lệ.

1. **Len (OpenFileID id):**

* Đây là một system call nhóm viết thêm để lấy được độ dài của file với ID hợp lệ.
* Các thao tác mở file hoàn toàn giống system call Open(char\* name, int type).
* Nếu mở file thành công, dùng hàm Length() của lớp FileSystem để lấy được chiều dài của file với ID hợp lệ.
* Lưu độ dài vừa lấy được vào thanh ghi số 2, tăng PC lên và trả về.

1. **ToString (char\* buffer, int num):**

**Các chương trình minh họa:**

1. **help:** dùng để in ra màn hình các dòng giới thiệu cơ bản về nhóm. Nội dung mô tả được đọc từ tập tin mota.txt

* Dùng system call Open(char\* name, int type) để mở file “mota.txt”.
* Dùng system call Len(OpenFileID id) để lấy độ dài của file.
* Dùng system call Read(char\* buffer, int charcount, OpenFileID id) để đọc tất cả thông tin của file “mota.txt” và lưu từng ký tự vào mảng ký tự buffer.
* Dùng system call PrintString(char\* buffer) để in từng ký tự của mảng ký tự buffer ra màn hình.
* Dùng system call Halt() để dừng NachOS lại và in các thông số ra màn hình.

1. **ascii:** dùng để in ra màn hình bảng mã ASCII và ghi kết quả vào tập tin ascii.txt

* Dùng system call CreateFile(char\* name) để tạo “ascii.txt”.
* Dùng system call Open(char\* name, int type) để mở file “ascii.txt” .
* Dùng vòng lặp để lặp 126 lần và lần lượt dùng system call PrintChar(char character) để in ký tự ra màn hình Console và dùng system call Write(char\* buffer, int charcount, OpenFileID id) để viết 126 ký tự ASCII vào file “ascii.txt”.
* Dùng system call CloseFile(OpenFileID id) để đóng file “ascii.txt” lại.
* Dùng system call Halt() để dừng NachOS lại và in các thông số ra màn hình.

1. **quicksort:** cho phép nhập vào n số nguyên từ bàn phím (n < 100), sử dụng thuật toán QuickSort để sắp xếp và lưu kết quả vào tập tin quicksort.txt

* Dùng system call PrintString(char\* buffer) để in dòng “Input n: “ ra màn hình Console yêu cầu người dùng nhập vào số lượng số nguyên muốn đưa vào mảng.
* Dùng system call ReadInt() để đọc số người dùng nhập vào từ màn hình Console.
* Dùng vòng lặp for vào kết hợp system call PrintString(char\* buffer) và ReadInt() để nhập tất cả số nguyên vào mảng số nguyên.
* Dùng thuật toán Quick Sort để sắp xếp lại mảng số nguyên hiện có.
* Dùng system call CreateFile(char\* buffer) để tạo ra file “quicksort.txt”
* Dùng system call Open(char\* name, int type) để mở file “quicksort.txt”.
* Sử dụng vòng lặp for để quét qua tất cả số nguyên trong mảng đã sắp xếp.
* Dùng system call ToString(char\* buffer, int num) để chuyển số nguyên về thành chuỗi ký tự để có thể lưu vào file.
* Dùng system call Write(char\* buffer, int charcount, OpenFileID id) để viết các chuỗi ký tự có được từ các số nguyên vào file “quicksort.txt”.
* Dùng system call CloseFile(OpenFileId id) để đóng file lại.
* Dùng system call Halt() để dừng NachOS lại và in các thông số ra màn hình.