

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN MẠNG MÁY TÍNH - VIỄN THÔNG



BÀI BÁO CÁO

Project 2 – Hệ điều hành Nachos

Môn học: Hệ điều hành

Lớp: 22CLC06

Giáo viên hướng dẫn: Ths. Lê Viết Long

Sinh viên thực hiện:

- + 22127022 - Võ Hoàng Anh**
- + 22127154 - Nguyễn Gia Huy**
- + 22127210 - Phạm Anh Khôi**
- + 22127413 - Phạm Hoàng Tiên**

Mục lục

I. Thông tin nhóm.....	3
II. Bảng phân công công việc.....	3
III. Đánh giá mức độ hoàn thành.....	3
IV. Các bước thực hiện.....	4
A. Cài đặt Nachos trên Ubuntu.....	4
B. Đọc hiểu các tập tin trong đề án.....	5
C. Hiểu thiết kế của hệ điều hành.....	6
D. Sử dụng Class SynchConsole.....	7
1. Tại sao cần dùng Class SynchConsole ?.....	7
2. Cách thêm Class SynchConsole vào Nachos.....	8
E. Cài đặt các Syscall.....	10
1. Syscall Read - Print cho Int - Float - Char - String.....	10
a. SC_ReadChar.....	13
b. SC_PrintChar.....	14
c. Các Syscall còn lại.....	14
2. Syscall Create, Open, Close, Read, Write cho File.....	15
a. openfile.*.....	15
b. filesys.*.....	15
c. Syscall SC_Open.....	16
d. Các Syscall còn lại.....	18
F. Viết chương trình người dùng.....	18
V. Hình ảnh demo chương trình.....	22
A. help.....	22
B. ascii.....	22
C. quicksort.....	23
D. mergesort.....	23
VI. Tài liệu tham khảo.....	24

I. Thông tin nhóm

- Môn: Hệ điều hành
- Lớp: 22CLC06
- Project 2 - Hệ điều hành Nachos
- Thành viên nhóm
- + 22127022 - Võ Hoàng Anh
- + 22127154 - Nguyễn Gia Huy
- + 22127210 - Phạm Anh Khôi
- + 22127413 - Phạm Hoàng Tiên

II. Bảng phân công công việc

MSSV	Họ và tên	Công việc
22127022	Võ Hoàng Anh	<code>ReadInt, PrintInt, ReadChar, PrintChar</code>
22127154	Nguyễn Gia Huy	<code>CompareFloat, FreeFloat, FloatToString</code> , chương trình minh họa, viết báo cáo
22127210	Phạm Anh Khôi	<code>ReadFloat, PrintFloat, ReadString, PrintString</code>
22127413	Phạm Hoàng Tiên	<code>Create, Open, Close, Read, Write</code>

III. Đánh giá mức độ hoàn thành

STT	Tên công việc	Mức độ hoàn thành
1	<code>ReadInt, PrintInt</code>	Hoàn thành
2	<code>ReadFloat, PrintFloat</code>	Hoàn thành
3	<code>ReadChar, PrintChar</code>	Hoàn thành
4	<code>ReadString, PrintString</code>	Hoàn thành
5	<code>Create, Open, Close</code>	Hoàn thành
6	<code>Read, Write</code>	Hoàn thành
7	<code>help, ascii</code>	Hoàn thành
8	<code>quicksort, mergesort</code>	Hoàn thành

IV. Các bước thực hiện

Bảng dưới đây là danh sách các file chính mà nhóm đã làm việc trong project lần này

STT	Tên file	Ý nghĩa	Mật độ
1	<code>syscall.h</code>	Khai báo các <code>System call(SC)</code>	10%
2	<code>exception.cc</code>	Viết các xử lý ngoại lệ cho <code>SC</code>	70%
3	<code>start.*</code>	Khai báo <code>SC</code> dưới dạng <code>Assembly</code>	5%
4	<code>openfile.*, filesys.*</code>	Hỗ trợ các thao tác với file	5%
5	<code>./test/*</code>	Viết các chương trình người dùng	10%

A. Cài đặt Nachos trên Ubuntu

- Yêu cầu: Ubuntu phiên bản 14.04 32 bit
- Link download:

<https://drive.google.com/uc?id=0B9icmOO7kaOncHpYTS1sN2Z0MkU&export=download>

Các bước tiến hành: Download nguyên gói Nachos đã cấu hình kèm cross-compiler-mips (download tập tin nachos.zip trong thư mục tài liệu hướng dẫn)

Tạo thư mục HDH ở Desktop để tiện làm việc

- + Bước 1: Cài đặt gcc và g++
 - + `sudo apt-get install gcc`
 - + `sudo apt-get install g++`
- + Bước 2: Giải nén nachos ở Desktop/HDH
- + Bước 3: Sửa Makefile (ở thư mục nachos 3.4/code): Sửa dòng `MAKE = gmake -> MAKE = make`
- + Bước 4: Truy cập vào thư mục code và biên dịch nachos bằng lệnh `make`
- + Bước 5: Chạy thử chương trình: vào thư mục `threads`, chạy lệnh nachos:
 - + `./nachos`

Kết quả như sau là thành công:

No threads ready or runnable, and no pending interrupts.

Assuming the program completed.

Machine halting!

Ticks: total 10, idle 0, system 10, user 0

Disk I/O: reads 0, writes 0

Console I/O: reads 0, writes 0

Paging: faults 0

Network I/O: packets received 0, sent 0

Cleaning up...

Một số lưu ý:

- + Có thể sử dụng lệnh `./userprog/nachos -rs 1023 -x ./test/halt` để chạy lệnh `halt` bằng chương trình `nachos`. Ý nghĩa cụ thể của các tham số trong câu lệnh sau:
 - `./userprog/nachos`: chạy chương trình nachos trong thư mục `./code/userprog`
 - `-rs`: Cờ Random seed
 - `1023`: Giá trị của cờ random seed (Giá trị này có thể là bất kì giá trị nào)
 - `-x`: Cờ thông báo chạy 1 chương trình người dùng
 - `./test/halt`: Giá trị của cờ là chương trình halt trong thư mục `./code/test`
- + Khi biên dịch bằng make sẽ tạo ra nhiều chương trình nachos ở các thư mục, một số `nachos` ở các thư mục sẽ giống nhau nhưng ta thống nhất sử dụng chương trình `nachos` ở thư mục `./userprog`. Chạy một chương trình người dùng thông qua lệnh (tại thư mục `./code`)


```
./userprog/nachos -x ./test/[Tên chương trình]
```

B. Đọc hiểu các tập tin trong đồ án

- + **progtest.cc** kiểm tra các thủ tục để chạy chương trình người dùng
- + **syscall.h** system call interface: các thủ tục ở kernel mà chương trình người dùng có thể gọi
- + **exception.cc** xử lý system call và các exception khác ở mức user, ví dụ như lỗi trang, trong phần mã chúng tôi cung cấp, chỉ có 'halt' system call được viết
- + **bitmap.*** các hàm xử lý cho lớp bitmap (hữu ích cho việc lưu vết các ô nhớ vật lý)
- + **fileSYS.h**
- + **openfile.h** định nghĩa các hàm trong hệ thống file nachos. Trong đồ án này chúng ta sử dụng lời gọi thao tác với file trực tiếp từ Linux, trong đồ án khác chúng ta sẽ triển khai hệ thống file trên ổ đĩa giả lập. (nếu kịp thời gian)
- + **translate.*** Phiên bản nachos chúng tôi gửi các bạn, chúng tôi giả sử mỗi địa chỉ ảo là cũng giống hệt như địa chỉ vật lý, điều này giới hạn chúng ta chỉ chạy 1 chương trình tại một thời điểm. Các bạn có thể viết lại phần này để cho phép nhiều chương trình chạy cùng lúc trong đồ án sau.
- + **machine.*** mô phỏng các thành phần của máy tính khi thực thi chương trình người dùng: bộ nhớ chính, thanh ghi, v.v.
- + **mipssim.cc** mô phỏng tập lệnh của MIPS R2/3000 processor (**Chứa lệnh tăng PROGRAM COUNTER, tìm từ khóa Advanced**, dùng trước khi trả giá trị về của các syscall)

- + **console.*** mô phỏng thiết bị đầu cuối sử dụng UNIX files. Một thiết bị có đặc tính (i) đơn vị dữ liệu theo byte, (ii) đọc và ghi các bytes cùng một thời điểm, (iii) các bytes đến bất đồng bộ
- + **synchconsole.*** nhóm hàm cho việc quản lý nhập xuất I/O theo dòng trong Nachos.
- + **../test/*** Các chương trình C sẽ được biên dịch theo MIPS và chạy trong Nachos
- + **system.*** chứa khai báo và xử lý các biến toàn cục sử dụng trong Nachos

C. Hiểu thiết kế của hệ điều hành

Giống như các hệ điều hành hiện đại, hệ điều hành Nachos cũng chia thành 2 không gian bộ nhớ:

- + **Kernel - system space**: không gian hệ thống, chứa các thông tin về chương trình người dùng
- + **User space**: không gian người dùng, chứa các thông tin về biến, hàm, bộ nhớ, chương trình của hệ thống

Việc chuyển quyền điều khiển từ **user mode** thành **system mode** được thực hiện thông qua **system calls**, software interrupt/trap.

Vì vùng nhớ *user space* không có ý nghĩa với *kernel*, chúng ta cần chuyển nội dung từ *user space* vào *kernel* sao cho kernel có thể xử lý được dữ liệu này. Khi trả thông tin từ system về *user space*, thì các giá trị được đặt trong các thanh ghi của CPU

Dưới đây là một số ví dụ thực tế minh họa cho các lý thuyết ở trên bằng mã nguồn

```
/// @brief Read a string from the console input
/// @param buffer Buffer to store the string (Stored in register 4)
/// @param size Size of the buffer (Length of the string) (Stored in register 5)
void ReadString(char *buffer, int size);
```

```
/// @brief Handle system call ReadString from user program
void Handle_SC_ReadString()
{
    int virtAddr;        // Virtual address of input string from user space
    int length;          // Length of input string
    char *buffer = NULL; // Buffer to store input string

    virtAddr = machine->ReadRegister(4); // Read virtual address of input string PARAMETER from register 4
    length = machine->ReadRegister(5);   // Read maximum length of input string PARAMETER from register 5

    buffer = User2System(virtAddr, length); // Copy buffer from User memory space to System memory space

    gSynchConsole->Read(buffer, length); // Use SynchConsole to read buffer from console

    System2User(virtAddr, length, buffer); // Copy buffer from System memory space to User memory space

    delete buffer;
    return IncreasePC();
}
```

Ví dụ trên minh họa về `System call ReadString` với 2 tham số đầu vào là `buffer` và `size` dùng để đọc một chuỗi kí tự từ `Console`. Ở chương trình người dùng, khi gọi hàm (từ bây giờ sẽ dùng từ `Syscall`, là các hàm được khai báo trong `syscall.h`). Tham số từ chương trình người dùng truyền vào `Syscall` sẽ được lưu trong các thanh ghi, cụ thể trong `Syscall` trên, tham số `buffer` được lưu trong thanh ghi thứ 4 và tham số `size` được lưu trong thanh ghi thứ 5.

Ở `Kernel space`, hàm `Handle_SC_ReadString` xử lí `Syscall ReadString` sẽ sử dụng các biến cục bộ của `Kernel` với dữ liệu được đọc từ `User space` thông qua `machine->ReadRegister`. Như ở ví dụ trên thì giá trị của `buffer` sẽ được truyền vào tham số `virtAddr` thông qua phương thức đọc thanh ghi thứ 4 của `machine`, tương tự như vậy với biến `length`.

Tuy nhiên, giá trị địa chỉ truyền vào `buffer` của `User space` (lúc này được lưu trong biến `virtAddr`) chỉ là địa chỉ ảo của `User space`. Ta cần chuyển dữ liệu từ `User space` sang `Kernel` thông qua hàm `User2System`. Tại đây, ta có được dữ liệu `buffer` của `Kernel` có giá trị bằng với `buffer` của `User Space` (nhưng khác về địa chỉ). Sau đó ta đọc dữ liệu chuỗi nhập vào thông qua dòng tiếp theo `gSynchConsole->Read(buffer, length)`, phần sau sẽ nói rõ hơn về `class SynchConsole` và các phương thức xử lí của nó

Sau khi đọc dữ liệu vào `buffer` của `Kernel` xong, ta thực hiện sao chép giá trị của `buffer` ở `Kernel` vào `buffer` của `User space` (lúc này là biến `virtAddr`). Sau khi sao chép xong, giá trị của `buffer` ở `Kernel` và `User space` là như nhau (tuy nhiên vẫn khác địa chỉ)

Cuối cùng ta xóa vùng nhớ đã cấp phát ở `Kernel` trong biến `buffer` và tăng giá trị `Program Counter` (điều này là bắt buộc với mỗi hàm xử lí `Syscall`)

D. Sử dụng Class SynchConsole

1. Tại sao cần dùng Class SynchConsole ?

`Class SynchConsole` hỗ trợ đọc và ghi dữ liệu từ `Kernel` vào `User space`. Cụ thể, lớp này hỗ trợ 2 phương thức `Read` để đọc dữ liệu từ `Terminal (User space)` vào `Kernel buffer` và phương thức `Write` để ghi dữ liệu từ `Kernel buffer` vào `Terminal`. Phần khai báo của 2 phương thức được mô tả chi tiết như đoạn mã bên dưới.

```

/// @brief Read a buffer from the console input device and store it in the kernel buffer
/// @param into Buffer to store the input
/// @param numBytes Number of bytes to read
/// @return Number of bytes actually read
int Read(char *into, int numBytes);

/// @brief Write a buffer from the kernel to the console output device
/// @param from Kernel buffer to write
/// @param numBytes Number of bytes to write
/// @return Number of bytes actually written
int Write(char *from, int numBytes);

```

Tóm lại, nếu các **Register** trong **machine** được dùng để đọc và ghi giá trị tham số, giá trị trả về của **Syscall** thì **SynchConsole** dùng để đọc dữ liệu từ **Userspace** và ghi dữ liệu vào **User space**

2. Cách thêm **Class SynchConsole** vào **Nachos**

- + **Bước 1:** Chuẩn bị 2 file **synchcons.h** và **synchcons.cc** đã được cung cấp trong đồ án
- + **Bước 2:** Sao chép 2 file trên vào thư mục **/code/threads**
- + **Bước 3:** Mở file **Makefile.common**, thêm vào cuối các mục **USERPROG_H**, **USERPROG_C**, **USERPROG_O** các đường dẫn như hình bên dưới

```

USERPROG_H = ../userprog/addrspace.h\
../userprog/bitmap.h\
../fileysys/fileysys.h\
../fileysys/openfile.h\
../machine/console.h\
../machine/machine.h\
../machine/mipssim.h\
../machine/translate.h\
../threads/synchcons.h

USERPROG_C = ../userprog/addrspace.cc\
../userprog/bitmap.cc\
../userprog/exception.cc\
../userprog/progtest.cc\
../machine/console.cc\
../machine/machine.cc\
../machine/mipssim.cc\
../machine/translate.cc\
../threads/synchcons.cc

USERPROG_O = addrspace.o bitmap.o exception.o progtest.o console.o machine.o \
✦ mipssim.o translate.o synchcons.o      You, yesterday • update

```


- + **Bước 4:** Trong `/code/threads`, mở `system.h`, khai báo biến `extern` để có thể sử dụng trong toàn bộ mã nguồn trong phần `USER_PROGRAM`

```
#ifdef USER_PROGRAM
#include "machine.h"
#include "synchcons.h"
#include "synch.h"
extern Machine *machine;
extern SynchConsole *gSynchConsole;

#endif
```

- + **Bước 5:** Tiếp tục mở `system.cc`, tại các mục `USER_PROGRAM`, khai báo biến, cấp phát bộ nhớ khi bắt đầu vào thu hồi bộ nhớ đã cấp phát cho biến khi kết thúc chương trình. Tham khảo hình minh họa bên dưới

```
#ifdef USER_PROGRAM // requires either FILESYS or FILESYS_STUB
Machine *machine; // user program memory and registers
SynchConsole *gSynchConsole; // the synchronized console
#endif
```

```
#ifdef USER_PROGRAM
    machine = new Machine(debugUserProg); // this must come first
    gSynchConsole = new SynchConsole();
#endif
```

```
#ifdef USER_PROGRAM
    delete machine;
    delete gSynchConsole;
#endif
```

- + **Bước 6:** Sử dụng biến `gSynchConsole` để xử lý các `Syscall` trong `exception.cc`

E. Cài đặt các Syscall

1. Syscall Read - Print cho Int - Float - Char - String

Vì các chương trình người dùng chỉ có thể dùng các Syscall để tương tác với hệ điều hành nên cần phải viết các Syscall cho các thao tác nhập xuất các kiểu dữ liệu khác nhau (Chứ không thể scanf, printf đối với C hay cin, cout với C++). Sau đây là hướng dẫn chi tiết các bước để viết một Syscall, để đơn giản trong quá trình xử lý logic, ta sẽ minh họa xử lý Syscall ReadChar và PrintChar

- + Bước 1: Mở syscall.h, định nghĩa giá trị cho Syscall (kí hiệu là SC_ + tên Syscall). Hình bên dưới định nghĩa giá trị của SC_ReadChar là 45 và SC_PrintChar là 46

```

13  #ifndef SYSCALLS_H
14  #define SYSCALLS_H
15
16  #include "copyright.h"
17
18  /* system call codes -- used by the stubs to tell the kernel which system call
19   * is being asked for
20   */
21  #define SC_Halt 0
22  #define SC_Exit 1
23  #define SC_Exec 2
24  #define SC_Join 3
25  #define SC_CreateFile 4
26  #define SC_Open 5
27  #define SC_Read 6
28  #define SC_Write 7
29  #define SC_Close 8
30  #define SC_Fork 9
31  #define SC_Yield 10
32
33  #define SC_ReadInt 41
34  #define SC_PrintInt 42
35  #define SC_ReadFloat 43
36  #define SC_PrintFloat 44
37  #define SC_ReadChar 45
38  #define SC_PrintChar 46
39  #define SC_ReadString 47
40  #define SC_PrintString 48
41

```

- + Bước 2: Trong syscall.h, sau khi định nghĩa, ta khai báo các Syscall dưới dạng các hàm như hình minh họa dưới đây

```

72  ✨
73  /// @brief Read a character from the console input and write it to register 2
74  /// @return Character read from the console input
75  char ReadChar();
76  ...
77  /// @brief Write a character to the console output
78  /// @param character Character is stored in register 4
79  void PrintChar(char character);
80
81  /// @brief Read a string from the console input
82  /// @param buffer Buffer to store the string (Stored in register 4)
83  /// @param size Size of the buffer (Length of the string) (Stored in register 5)
84  void ReadString(char *buffer, int size);
85
86  /// @brief Write a string to the console output
87  /// @param buffer Buffer contains the string (Stored in register 4)
88  void PrintString(char *buffer);
89

```

- + Bước 3: Trong thư mục `/code/test`, mở `start.s` và `start.c`, thêm vào 2 file này đoạn mã như hình bên dưới

```

151
152     .globl ReadChar
153     .ent    ReadChar
154 ReadChar:
155     addiu $2,$0,SC_ReadChar
156     syscall
157     j     $31
158     .end ReadChar
159
160     .globl PrintChar
161     .ent    PrintChar
162 PrintChar:
163     addiu $2,$0,SC_PrintChar
164     syscall
165     j     $31
166     .end PrintChar
167

```

- + Bước 4: Trong thư mục `code/userprog`, mở `exception.cc`, viết mã nguồn xử lý `Syscall`. Hình bên dưới minh họa cấu trúc tổng thể (sử dụng cấu trúc `switch-case` của hàm `ExceptionHandler` để xử lý các `Exception`. Trong đó `SyscallException` xảy ra khi gọi `Syscall`

```

560 /// @brief Exception handler for user program system calls
561 /// @param which Type of exception
562 void ExceptionHandler(ExceptionType which)
563 {
564     int type = machine->ReadRegister(2); // Read system call code from register 2
565
566     switch (which)
567     {
568     case SyscallException: // System call exception
569         switch (type)
570         {
571             case SC_Halt:
572                 return Handle_SC_Halt();
573             case SC_ReadInt:
574                 return Handle_SC_ReadInt();
575             case SC_PrintInt:
576                 return Handle_SC_PrintInt();
577             case SC_ReadFloat:
578                 // SynchPrint("ReadFloat\n");
579                 return Handle_SC_ReadFloat();
580             case SC_PrintFloat:
581                 // SynchPrint("PrintFloat\n");
582                 return Handle_SC_PrintFloat();
583             case SC_ReadChar:
584                 return Handle_SC_ReadChar();
585             case SC_PrintChar:
586                 return Handle_SC_PrintChar();
587             case SC_ReadString:
588                 return Handle_SC_ReadString();
589             case SC_PrintString:
590                 return Handle_SC_PrintString();
591             case SC_CreateFile:
592                 return Handle_SC_CreateFile();
593             case SC_Open:
594                 return Handle_SC_Open();
595             case SC_Close:
596                 return Handle_SC_Close();
597             case SC_Read:
598                 return Handle_SC_Read();
599             case SC_Write:
600                 return Handle_SC_Write();
601             default:
602                 interrupt->Halt();
603                 break;
604         }
605     case NoException:
606         return;
607     case PageFaultException:
608         printf("PageFaultException: No valid translation found\n");
609         interrupt->Halt();
610         break;
611     case ReadOnlyException:
612         printf("ReadOnlyException: Write attempted to page marked \"read-only\"\n");
613         interrupt->Halt();
614         break;

```

Về cấu trúc tổng thể là như vậy. Tuy nhiên, để chương trình chạy đúng, ta cần xử lý chính xác các `Syscall`. Dưới đây là hình minh họa mã nguồn và giải thích chi tiết mã nguồn của hàm `Handle_SC_ReadChar` để xử lý khi gọi `SC_ReadChar` và `Handle_SC_PrintChar` để xử lý khi gọi `SC_PrintChar`

a. SC_ReadChar

```

143
144 /// @brief Handle system call ReadChar from user program
145 void Handle_SC_ReadChar()
146 {
147     int maxBytes = 255;           // Maximum length of buffer
148     char *buffer = new char[255]; // Buffer to store data
149     int numBytes = 0;             // The number of bytes read from console
150     char c = 0;                  // Character to store the result
151
152 > if (buffer == NULL) // Check if buffer is NULL (not enough memory in system)...
156 else // Buffer is not NULL
157 {
158     numBytes = gSynchConsole->Read(buffer, maxBytes); // Read buffer from console and return the number of bytes read
159
160 > if (numBytes > 1) // Check if the number of bytes read is greater than 1...
165 > else if (numBytes == 0) // Check if the number of bytes read is 0: empty character...
170 else // If the number of bytes read is 1, return the character to register 2
171 {
172     c = *buffer;
173 }
174 }
175 machine->WriteRegister(2, c); // Write the character to register 2
176 delete buffer;
177 return IncreasePC();
178 }
179

```

Chú thích chi tiết của hàm xử lý đã được comment ở đoạn **code** trên. Có một số điểm cần lưu ý như sau:

- + Sử dụng phương thức **Read** của **gSynchConsole** để đọc dữ liệu từ **User space**
- + Sau khi xử lý xong về mặt logic, ghi kết quả (kí tự được nhập) vào **Register[2]**, vì thanh ghi thứ 2 lưu giá trị trả về của **Syscall**. Phải ghi vào thì giá trị **Syscall** (giá trị của hàm **ReadChar** mới trả về giá trị để lưu vào biến khi sử dụng **Syscall** (hàm) trong chương trình người dùng
- + Cần tăng **Program Counter** sau mỗi **Syscall** để chương trình hoạt động đúng. Dưới đây là minh họa về hàm tăng **Program Counter**

```

56 /// @brief Increase Program Counter (needed for each system call) (4 bytes for each instruction)
57 void IncreasePC()
58 {
59     int counter = machine->ReadRegister(PCReg); // Read current Program Counter
60     machine->WriteRegister(PrevPCReg, counter); // Write current Program Counter to Previous Program Counter
61     counter = machine->ReadRegister(NextPCReg); // Read Next Program Counter
62     machine->WriteRegister(PCReg, counter); // Write Next Program Counter to Program Counter
63     machine->WriteRegister(NextPCReg, counter + 4); // Write Next Program Counter + 4 to Next Program Counter
64 }

```

b. SC_PrintChar

```

/// @brief Handle system call PrintChar from user program
void Handle_SC_PrintChar()
{
    char c = (char)machine->ReadRegister(4); // Read character parameter from register 4

    if (c != 0) // Check if the character is not null
    {
        // SynchPrint("Character that you entered: ");
        gSynchConsole->Write(&c, 1); // Write the character to console
    }

    return IncreasePC();
}

```

Tương tự như SC_ReadChar, chú thích chi tiết đã được comment code trên mã nguồn.

Một số lưu ý:

- + Đọc giá trị tham số đầu vào của Syscall void PrintChar(char c) ở thanh ghi thứ 4 và lưu vào biến cục bộ của hàm
- + Sử dụng phương thức Write của gSynchConsole để in kí tự đọc được ra màn hình
- + Tăng Program Counter thông qua hàm IncreasePC

c. Các Syscall còn lại

Các Syscall nhập xuất cho các kiểu dữ liệu còn lại trong đề án này cũng như các kiểu dữ liệu khác được cài đặt tương tự về mặt cấu trúc, chỉ khác phần xử lí logic tương ứng với từng kiểu dữ liệu khác nhau

- + SC_ReadInt
- + SC_PrintInt
- + SC_ReadFloat
- + SC_PrintFloat
- + SC_ReadString
- + SC_PrintString

Lưu ý:

- + Vì thanh ghi trong machine chỉ lưu được giá trị số nguyên nên SC_ReadFloat và SC_PrintFloat cần thao tác với float* để đảm bảo tính đúng đắn.
- + Vì chương trình người dùng không hỗ trợ toán tử giải tham chiếu, vì vậy khi muốn thực hiện các thao tác khác với số thực, ta cần viết các System call cho chúng. Ví dụ như CompareFloat, FreeFloat, FloatToString

2. Syscall Create, Open, Close, Read, Write cho File

Đối với các Syscall liên quan để xử lý File, chúng ta cần điều chỉnh lại các file `openfile.*` và `filesys.*` trong thư mục `/code/filesys`.

a. `openfile.*`

`Openfile.h` định nghĩa `class OpenFile` và các phương thức xử lý file trong hệ thống file `nachos` như: `Read`, `Write`, `Seek`, `Length`, `GetCurrentPos` để hỗ trợ cho phần cài đặt

Với đoạn mã nguồn được cho trước trong đồ án này, ta cần thêm vào `class` thuộc tính `type` để biểu diễn trạng thái của file (`openmode`) và cài đặt thêm `Constructor` có tham số `type` như hình minh họa bên dưới

```

108
109 | #define READ_WRITE 0
110 | #define READ_ONLY 1
111 | #define STDIN 2
112 | #define STDOUT 3
    | You, 57 seconds ago | 1 author (You)
113 | class OpenFile
114 | {
115 | public:
116 |     int type; // Open mode
117 |     // Open a file whose header is located at "sector" on the disk
118 |     OpenFile(int sector);
119 |     // Open a file with a specific type
120 |     OpenFile(int sector, int type);
121 |

```

`Type` có thể mang 1 trong 4 giá trị được định nghĩa như trên hình:

- + `READ_WRITE`: file cho phép đọc và ghi
- + `READ_ONLY`: file chỉ cho phép đọc
- + `STDIN`: thiết bị nhập chuẩn (bàn phím)
- + `STDOUT`: thiết bị xuất chuẩn (màn hình)

b. `filesys.*`

`Fileys.h` định nghĩa `class FileSystem` để quản lý các tập tin trong hệ thống file `nachos`, hỗ trợ một số phương thức như `Open` để mở file, `Remove` để đóng file, `Print` để in danh sách file hiện hành, ...

Với đoạn mã được cung cấp trong `filesystem.*` của đề án lần này, ta cần thêm vào `class FileSystem` một số thuộc tính và phương thức để quản lí và xử lí bảng file. Hình bên dưới là ví dụ minh họa về cách cài đặt bảng file đơn giản

```

126  class FileSystem
127  {
128  public:
129      OpenFile **file_table; // A table to store all the file
130      int index;             // Index of the file table
131
132      /// @brief Open the file with the given name and open mode
133      /// @param name Name of the file in ./code directory
134      /// @param type Type of the file (0->3)
135      /// @return OpenFile pointer of the opened file
136      OpenFile *Open(char *name, int type);
137

```

Thuộc tính `file_table` là một con trỏ `OpenFile` cấp 2 để quản lí danh sách các file trong hệ thống. Mỗi khi một file được mở, chương trình sẽ cấp phát bộ nhớ trên 1 vị trí của `file_table` để quản lí file đó. Khi đóng file, chương trình sẽ thu hồi bộ nhớ từ vùng đã được cấp phát cho file đó.

c. Syscall SC_Open

`SC_Open` dùng để mở một file trên hệ thống file `nachos`. Ví dụ về một cách cài đặt hàm xử lí `SC_Open` được minh họa chi tiết trong hình dưới đây

+ Trong `syscall.h`

```

115
116  /// @brief Open the file with the given name and open mode
117  /// @param name Name of the file in ./code directory (Stored in register 4)
118  /// @param type Type of the file (0: Read/Write, 1: Read only, 2: Console Input)
119  /// @return OpenFileId of the opened file (a slot in file table)
120  /// @note + Return -1 if the file is not found or the file is already opened
121  /// @note + Return -2 if unknown error occurs
122  OpenFileId Open(char *name, int type);
123

```


+ Trong `exception.cc`

```

372
373 /// @brief Handle system call Open from user program (Open file)
374 void Handle_SC_Open()
375 {
376     int virtAddr = machine->ReadRegister(4); // Virtual address of filename PARAMETER from register 4
377     int type = machine->ReadRegister(5);      // Type (Open mode) of the file PARAMETER from register 5
378     char *filename;                          // Buffer to store filename
379     int result = -2;                          // Result of the function
380
381     filename = User2System(virtAddr, MaxFileLength); // Copy buffer from User memory space to System memory space
382
383     int allocatedSlot = fileSystem->GetAllocatedSlot(); // Find free slot in file system
384
385     if (allocatedSlot != -1) // If there is free slot
386     {
387         if (type == READ_WRITE || type == READ_ONLY) // Handle ReadOnly and ReadWrite file cases
388         {
389             fileSystem->file_table[allocatedSlot] = fileSystem->Open(filename, type); // Open file with filename and type
390
391             if (fileSystem->file_table[allocatedSlot] != NULL)
392             {
393                 result = allocatedSlot; // Success, return free slot for normal file
394             }
395             else // Open file failed
396             {
397                 result = -1; // Failed to open file, return -1
398             }
399         }
400         else if (type == STDIN) // ConsoleInput: stdin
401         {
402             result = 0; // Success, return 0 for STDIN
403         }
404         else // ConsoleOutput: stdout
405         {
406             result = 1; // Success, return 1 for STDOUT
407         }
408     }
409     else // No free slot...
410     {
411         if (filename != NULL)
412             delete[] filename;
413
414         machine->WriteRegister(2, result); // Write result to register 2
415         return IncreasePC();
416     }
417 }
418
419
420
421

```

Một số lưu ý trong đoạn mã trên:

- + `fileSystem` là biến `extern FileSystem` được khai báo trong `system.h` ở phần `FILESYS_NEEDED`
- + Phương thức `GetAllocatedSlot()` trả về một vị trí trống trên bảng file
- + Phương thức `Open()` dùng để mở file và gán nó vào bảng file
- + Với mỗi giá trị của `type` sẽ ứng với mỗi cách xử lý khác nhau
- + Kết quả trả về của `Syscall` là vị trí của file trên bảng file

d. Các `Syscall` còn lại

Các `Syscall` cho `File` còn lại được cài đặt về mặt cấu trúc tương tự như `SC_Open`, chỉ khác về xử lý logic

- + `SC_CreateFile`: tạo file mới
- + `SC_Close`: đóng file (thu hồi vùng nhớ đã cấp phát trên `file_table`)
- + `SC_Read`: đọc dữ liệu từ file
- + `SC_Write`: ghi dữ liệu vào file

Chi tiết cách cài đặt và xử lý logic đã được viết và comment chi tiết trong mã nguồn (tham khảo `syscall.h` và `exception.cc`). Dưới đây là một số **lưu ý** trong quá trình cài đặt:

- + Đối với tên file / đường dẫn file được truyền vào tham số của `Syscall`, mặc định thư mục hiện hành của file đó là `./code`. Ví dụ nếu ta truyền vào `Syscall Create` tham số `filename = quicksort.txt` thì sẽ tạo file `quicksort.txt` tại thư mục `./code`. Tương tự như vậy với các `File Syscall` còn lại
- + Dùng các hàm `User2System` và `System2User` để chuyển đổi vùng nhớ qua lại giữa `Kernel` và `User space`
- + Xử lý logic đúng với từng `openmode` (`READ_WRITE`, `READ_ONLY`, `STDIN`, `STDOUT`) của file. Ví dụ với `Syscall Read` cho file `stdout` thì báo lỗi, hoặc `Write` cho file `stdout` thì cần dùng `gSynchConsole` để xuất dữ liệu ra màn hình.
- + Tăng `Program Counter` cuối mỗi `Syscall`

A. *Viết chương trình người dùng*

Trong phần này ta sẽ viết các chương trình người dùng - `User Program` sử dụng các `Syscall` của hệ thống. Để đơn giản hơn về mặt xử lý logic và tập trung vào cấu trúc tổng quát, ta viết chương trình `char_io_test` để xử lý nhập xuất 1 kí tự từ người dùng. Sau đây là chi tiết các bước thực hiện:

- + **Bước 1:** Tại thư mục `/code/test`, tạo file `char_io_test.c`. Hình bên dưới là minh họa đơn giản về chương trình này

```
You, 4 hours ago | 1 author (You)
1  #include "syscall.h"
2
3  #define NULL 0
4  #define BACKSPACE 8
5  #define ESC 27
6  #define SPACE 32
7  #define DELETE 127
8  #define TAB 9
9
10 int main()
11 {
12     char c;
13     PrintString("*=====*\n");
14     PrintString("| Welcome to the Char I/O Test Suite |\n");
15     PrintString("*=====*\n");
16
17     PrintString("Enter a character: ");
18     c = ReadChar();
19     PrintString("You entered: ");
20     switch (c)
21     {
22     case NULL:
23         PrintString("NULL\n");
24         break;
25     case ESC:
26         PrintString("ESC\n");
27         break;
28     case SPACE:
29         PrintString("SPACE\n");
30         break;
31     case TAB:
32         PrintString("TAB\n");
33         break;
34     default:
35         PrintChar(c);
36         break;
37     }
38     PrintChar('\n');
39 }
```

You, 2 days ago • [code]: Write syscall for I/O Int, Char, String

Chương trình sẽ cho phép người dùng nhập vào 1 kí tự và in ra kí tự vừa nhập, xử lí lỗi nếu nhập nhiều hơn 1 kí tự. Chương trình sử dụng `syscall ReadChar` để đọc 1 kí tự từ thiết bị nhập của người dùng và `PrintChar` để in ra kí tự ra màn hình người dùng

Lưu ý: Khi viết các chương trình người dùng, các biến sử dụng trong chương trình tất cả nên được khai báo trước các lệnh xử lí để tránh gặp lỗi trong quá trình biên dịch. Ví dụ như trong `quicksort.c`, tất cả các biến liên quan đến mảng số nguyên, thuật toán `quicksort` và xử lí `file` được khai báo trước khi xử lí logic như hình minh họa bên dưới

```
1  #include "syscall.h"
2
3  #define MAX_SIZE (100)
4
5  int main()
6  {
7      int a[MAX_SIZE]; // Array to be sorted
8      int n;           // Number of elements in the array
9      int order;       // Order of sorting
10
11     /* Quicksort */
12
13     int stack[MAX_SIZE]; // Stack for storing left and right indices of sub arrays
14     int top;             // Top of the stack
15     int left;            // Left index of the sub array to be sorted
16     int right;           // Right index of the sub array to be sorted
17     int pivotIndex;      // Index of the pivot element
18     int pivot;           // Pivot element
19     int i, j;            // Loop variables
20     int temp;            // Temporary variable for swapping
21
22     /* File handling */
23     int openFileId; // File descriptor for the file to be written
```

- + **Bước 2:** Mở **Makefile** trong thư mục **code/test** và thêm dữ liệu vào file như hình minh họa bên dưới để chương trình có thể hiểu và biên dịch

```

38  #-----
39  all: halt char_io_test string_io_test int_io_test float_io_test help ascii quicksort mergesort
40  # -----
41  start.o: start.s ../userprog/syscall.h
42      $(CPP) $(CPPFLAGS) start.c > strt.s
43      $(AS) $(ASFLAGS) -o start.o strt.s
44      rm strt.s
45
46  # -----
47  char_io_test.o: char_io_test.c
48      $(CC) $(CFLAGS) -c char_io_test.c
49  char_io_test: char_io_test.o start.o
50      $(LD) $(LDFLAGS) start.o char_io_test.o -o char_io_test.coff
51      ../bin/coff2nooff char_io_test.coff char_io_test
52  # -----

```

- + **Bước 3:** Chuyển hướng đến thư mục **code**, biên dịch bằng lệnh **make all**. Nếu biên dịch thành công, gõ lệnh sau để chạy chương trình **char_io_test**
./userprog/nachos -x ./test/char_io_test
 Hình bên dưới là minh họa khi chương trình chạy thành công khi ta nhập **TAB**

```

→ code git:(main) ./userprog/nachos -x ./test/char_io_test
*=====*
| Welcome to the Char I/O Test Suite |
*=====*
Enter a character:
You entered: TAB

Machine halting!

Ticks: total 1300399758, idle 1300397890, system 1790, user 78
Disk I/O: reads 0, writes 0
Console I/O: reads 2, writes 160
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
→ code git:(main) █

```

V. Hình ảnh demo chương trình

A. *help*

Chương trình *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*

```

mota.txt
1 Course: Operating System
2 Class: 22CLC06
3 Project 2 - Nachos Operating System
4 Group members:
5 22127022 - Vo Hoang Anh
6 22127154 - Nguyen Gia Huy
7 22127210 - Pham Anh Khoi
8 22127413 - Pham Hoang Tien
9
10 Usage (At code directory):
11 + make all
12 + ./userprog/nachos -x test/[program_name]
13
14 There are 4 programs for user to run:
15 1. help: Print information about us and available commands from [mota.txt]
16 2. ascii: Print ASCII table to console and write it to [ascii.txt]
17 3. quicksort: Sort an array of integers using quicksort algorithm,
18 then print the sorted array to console and write it to [quicksort.txt]
19 4. mergesort: Sort an array of floats using mergesort algorithm,
20 then print the sorted array to console and write it to [mergesort.txt]
21
22 Note: The output files will be created in [code] directory
23
24 THANK YOU FOR READING!
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
258
```

C. quicksort

Chương trình **quicksort** cho phép nhập vào n số nguyên từ bàn phím ($0 < n < 100$), sử dụng thuật toán **QuickSort** để sắp xếp và lưu kết quả vào tập tin **quicksort.txt**

```

nachos > nachos-3.4 > code > quicksort.txt
00000000  FC FF FF FF FE FF FF FE FF FF FF 00 00 00 00
00000010  01 00 00 00 03 00 00 00 03 00 00 00 04 00 00 00
00000020  05 00 00 00 05 00 00 00

binary  00000101      octal  005
uint8   5            int8    5
uint16  5            int16   5
uint24  5            int24   5
uint32  5            int32   5
uint64  End of File   int64   End of File
ULEB128 5            SLEB128 5
float16  2.980232238769531e-7 bfloat16  4.5917748078
float32  7.006492321624085e-45 float64  End of File
ASCII    ☐            UTF-8    ☐
UTF-16   ☐            GB18030  ☐
BIG5     ☐            SHIFT-JIS ☐
☑ Little Endian

code git:(main) x ./userprog/nachos -x ./test/quicksort
Enter number of elements in the array: 10
+ Enter element [0]: -4
+ Enter element [1]: 5
+ Enter element [2]: -2
+ Enter element [3]: 3
+ Enter element [4]: 4
+ Enter element [5]: 5
+ Enter element [6]: 1
+ Enter element [7]: 0
+ Enter element [8]: -2
+ Enter element [9]: 3
Initial array: [-4, 5, -2, 3, 4, 5, 1, 0, -2, 3]
Sorted array: [-4, -2, -2, 0, 1, 3, 3, 4, 5, 5]
Write sorted array to file quicksort.txt successfully
Machine halting!
Ticks: total 1854294252, idle 1854283727, system 6860, user 3665
Disk I/O: reads 0, writes 0
Console I/O: reads 26, writes 469
Paging: faults 0
Network I/O: packets received 0, sent 0
Cleaning up...
code git:(main) x
code git:(main) x

```

D. mergesort

Chương trình **mergesort** cho phép nhập vào n số thực từ bàn phím ($0 < n < 100$), sử dụng thuật toán **MergeSort** để sắp xếp và lưu kết quả vào tập tin **mergesort.txt**

```

mergesort.txt
1  -4.28000
2  -2.28000
3  0.400000
4  4.900000
5  5.100000
6  5.200000
7

code git:(main) x ./userprog/nachos -x ./test/mergesort
Enter number of elements in the array (n, 1 <= n < 100): 6
+ Enter float element [0]: -2.3
+ Enter float element [1]: 4.9
+ Enter float element [2]: 5.2
+ Enter float element [3]: -4.3
+ Enter float element [4]: 5.1
+ Enter float element [5]: 0.4
-> Initial array: [-2.29, 4.90, 5.20, -4.29, 5.10, 0.40]
=> Sorted array : [-4.28, -2.28, 0.40, 4.90, 5.10, 5.20]
-> Write sorted array to file mergesort.txt successfully
*****
| Thank you for using this program in Nachos! |
*****
Machine halting!
Ticks: total 2062657562, idle 2062646998, system 7300, user 3264
Disk I/O: reads 0, writes 0
Console I/O: reads 28, writes 587
Paging: faults 0
Network I/O: packets received 0, sent 0
Cleaning up...
code git:(main) x
code git:(main) x

```

VI. Tài liệu tham khảo

- [1]. Ths. Lê Viết Long, *Tài liệu Project 2 - 20240315*
- [2]. Hào, H. C. [@hachihao1604]. (2021, October 12). FIT HCMUS | Hệ điều hành | Đồ án 1 NachOS. Youtube. <https://www.youtube.com/watch?v=H0w6OdM1H-M>
- [3]. Nguyễn, T. C. [@thanhchungnguyen2618]. (n.d.). Lập trình nachos HCMUS. Youtube. Retrieved March 31, 2024, from https://www.youtube.com/playlist?list=PLRgTVtca98hUgCN2_2vzsAAXPiTFbvHpO