ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN 2

BỘ MÔN: HỆ ĐIỀU HÀNH

Giảng viên hướng dẫn:

ThS. Lê Giang Thanh

ThS. Nguyễn Thanh Quân

20120307 – PHẠM GIA KHIÊM
20120519 – NGUYỄN THỊ THÚY LIỄU
20120540 – VÕ HOÀNG THẢO NGUYÊN

Thành phố Hồ Chí Minh - 2022

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN 2

BỘ MÔN: HỆ ĐIỀU HÀNH

Giảng viên hướng dẫn:

ThS. Lê Giang Thanh

ThS. Nguyễn Thanh Quân

Thành phố Hồ Chí Minh - 2021

LỜI CẢM ƠN

Trong quá suốt quá trình học tập môn học Hệ điều hành và thực hiện đồ án này, nhóm chúng em đã nhận được nhiều sự hướng dẫn góp ý, giúp đỡ tận tình từ thầy cô, bạn bè.

Nhóm em rất cảm ơn ThS. Nguyễn Thanh Quân và ThS. Lê Giang Thanh, khoa Công nghệ thông tin trường Đại học Khoa học Tự nhiên, Đại học Quốc gia TPHCM đã hướng dẫn và truyền đạt rất nhiều kiến thức bổ ích với chúng em trong suốt thời gian qua.

Cảm ơn chân thành những lời góp ý của bạn bè và sự góp sức của bản thân các thành viên trong nhóm hoàn thành đồ án.

Trân trọng cảm ơn!

MỤC LỤC

TỔNG QUAN VỀ NHÓM

I. THÔNG TIN NHÓM.

1. Danh sách thành viên.

STT	Họ và tên	Mã số sinh viên	Email
1	Phạm Gia Khiêm	20120307	20120307@student.hcmus.edu.vn
2	Nguyễn Thị Thúy Liễu	20120519	20120519@student.hcmus.edu.vn
3	Võ Hoàng Thảo Nguyên	20120540	20120540@student.hcmus.edu.vn

2. Bảng phân công công việc.

STT	Họ và tên	Mã số sinh viên	Phân công
1	Phạm Gia Khiêm	20120307	➢ SystemCall: SC_Create và SC_Remove. ➢ Chương trình: createfile, cat, copy.
2	Nguyễn Thị Thúy Liễu	20120519	➢ SystemCall: SC_Open và SC_Close. ➢ Chương trình: delete, concatenate. ➢ Không để user làm sụp hệ thống.
3	Võ Hoàng Thảo Nguyên	20120540	➢ SytemCall: SC_Read và SC_Write và SC_Seek. ➢ Sao chép vùng nhớ kernel – user. ➢ Test lỗi các SystemCall và các chương trình.

II. Bảng tự đánh giá mức độ hoàn thành.

Chức năng	Người thực hiện	Tự đánh giá (%)
SystemCall Create	Khiêm	100
SystemCall Open	Liễu	100

SystemCall Close	Liễu	100
SystemCall Read	Nguyên	100
SystemCall Write	Nguyên	100
SystemCall Seek	Nguyên	100
SystemCall Remove	Khiêm	100
Sao chép vùng nhớ kernel – user	Nguyên	100
Chương trình createfile	Khiêm	100
Chương trình cat	Khiêm	100
Chương trình copy	Khiêm	100
Chương trình delete	Liễu	100
Chương trình concatenate	Liễu	100
Không để user làm sụp hệ điều hành	Liễu	100
Báo cáo	All	100

TỔNG QUAN THIẾT LẬP

I. CÁC CÀI ĐẶT CÓ SẴN:

- **progtest.cc** kiểm tra các thủ tục để chạy chương trình người dùng.
- **syscall.h** SystemCall interface: các thủ tục ở kernel mà chương trình người dùng có thể gọi.
- **exception.cc** xử lý SystemCall và các exception khác ở mức user, ví dụ như lỗi trang, trong phần mã chúng tôi cung cấp, chỉ có 'halt' SystemCall 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
- **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

II. CÁC CÀI ĐẶT CỦA PHẦN TRƯỚC:

- **SC_ReadString**: Đọc chuỗi kí tự từ hệ thống.
- **SC_PrintString**: In chuỗi kí tự ra màn hình.
- Hàm **System2User:** Sao chép dữ liệu từ Hệ thống đưa cho Người dùng.
- Hàm **User2System:** Sao chép dữ liệu từ Người dùng đưa cho Hệ thống.

III. CÁC GIÁ TRỊ THANH GHI:

- ***Register 2***: Lưu mã syscall và lưu kết quả trả về của mỗi syscall nếu có.
- ***Register 4***: Lưu tham số thứ nhất.
- ***Register 5***: Lưu tham số thứ hai.
- ***Register 6***: Lưu tham số thứ ba.
- ***Register 7***: Lưu tham số thứ tư.

CÀI ĐẶT SYSTEMCALL THAO TÁC VỚI FILE

1. Cài đặt SystemCall Create:

- Mô tả cài đặt ***SC_Create***:
  - Input: char* filename.
  - Output: Thành công: 0, Lỗi: -1.
  - Used: Tạo một file rỗng.
- **Ý tưởng:** Chuyển dữ liệu filename từ system to user, kiểm tra xem filename hợp lệ không, nếu không → báo lỗi, nếu hợp lệ tạo file mới nếu thành công return 0, nhưng nếu không đủ vùng nhớ hoặc tạo gặp lỗi thì báo lỗi return -1.
- **Cách cài đặt:**
  - Bước 1: Chuyển đổi vùng nhớ từ System thành User (dùng hàm System2User) lấy được filename.
  - Bước 2: Kiểm tra xem filename có hợp lệ hay không. Nếu không thì đến bước 4.
  - Bước 3: Tạo file với tên là filename, nếu thỏa mãn các ràng buộc → file được tạo, nếu không thì file không được tạo.
  - Bước 4: Trả về giá trị -1 nếu file không được tạo, 0 nếu file được tạo.

2. Cài đặt SystemCall Open:

- Mô tả cài đặt **SC_Open**:
  - Input: char* filename, int type.
  - Output: Thành công: 0, Không thành công: -1.
  - Used: Mở file.
- **Ý tưởng:** Chuyển dữ liệu filename từ system to user, với type là cách mở (type = 0: Mở để đọc và ghi; type = 1: mở để đọc). Kiểm tra filename hợp lệ hay không. Sau đó tiến hành mở file, nếu mở được thì tiếp tục tùy chọn cách mở file dựa trên type của đầu vào.
- **Cách cài đặt:**
  - Bước 1: Đọc dữ liệu từ thanh ghi register4 và register5 và lưu vào 2 biến filenameAddr và fileType.
  - Bước 2: Từ filenameAddr chuyển đổi vùng nhớ từ System sang User và lưu vào filename. Sau đó kiểm tra tính hợp lệ của filename. Nếu không hợp lệ thì return.
  - Bước 3: Chọn file với tên là filename để mở, nếu chọn mở không thành công thì return -1.

- Bước 4: Tìm vị trí trống trong bảng des[1], nếu không còn index trống cấp cho file thì return -1.
- Bước 5: Mở file với type đã chọn. Nếu type = 0 thì mở để đọc và ghi file, type = 1 thì mở chỉ để đọc. Trong quá trình mở nếu không thành công thì return -1, còn thành công thì return 0.

3. Cài đặt SystemCall Close:

- Mô tả cài đặt **SC_Close**:
  - Input: fileID.
  - Output: Thành công: 0, Không thành công: -1.
  - Used: Đóng file.
- **Ý tưởng:** Lấy fileID từ thanh ghi, nếu fileID < 0 hoặc fileID > số lượng file đã được định nghĩa ban đầu thì đóng không thành công. Trong trường hợp fileID thỏa mãn, nếu không xóa fileID ra khỏi bảng des được thì đóng file không thành công.
- **Cách cài đặt:**
  - Bước 1: Đọc fileID từ thanh ghi register4.
  - Bước 2: Kiểm tra fileID có nằm trong khoảng [0, MAX_FILE_NUM[2]] hay không nếu có thì ta sẽ đóng fileID và return 0, nếu đóng không được thì return -1; nếu không nằm trong khoảng [0, MAX_FILE_NUM] thì return -1.

4. Cài đặt SystemCall Read:

- Mô tả cài đặt **SC_Read**:
  - Input: char* buffer, int size, OpenFileID id.
  - Output: Số lượng kí tự đọc thành công.
  - Used: Đọc các kí tự trong file vào buffer với kích thước size.
- **Ý tưởng:** Đọc các tham gia số từ các thanh ghi 4, 5 và 6 tương ứng với bufferAddress, size, và fileID. Xét fileID, nếu fileID = 0 thì chuyển sang đọc input từ console; Nếu fileID = 1, đây là consoleOUT nên ta không thể đọc → báo lỗi; Những trường hợp còn lại, ta sẽ tìm kím vùng nhớ của file trong file descriptor table, đọc nội dung file vào buffer, chuyển dữ liệu về lại bufferAddress qua hàm system2user và trả về số byte đọc được nếu FileID hợp lệ, ngược lại báo lỗi.
- **Cách cài đặt:**
  - Bước 1: Đọc bufferAddress, len, fileID lần lược từ register4, 5, 6. Xét điều kiện hợp lệ của bufferAddress, len, fileID. Nếu hợp lệ thì sẽ ghi vào register2 số byte đọc được, nếu không thì báo lỗi.

[1] Table gồm 20 phần tử chứa danh sách file opening. Nếu index còn trống để cấp cho file thì return index, không thì index = -1.
[2] Số lượng phần tử tối đa mà Table chứa được. Trong cài đặt thì MAX_FILE_NUM = 20.

- Bước 2: Khởi tạo cntChar = -1.
- Bước 3: Xét điều kiện của fileID. Nếu fileID = 0, thì sẽ làm tương tự như cách làm ReadString. Nếu fileID = 1, thì bỏ qua vì đây là consoleOUT. Các trường hợp fileID còn lại ta sẽ qua bước 4.
- Bước 4: Tạo một char* buffer có kích thước là len, sau đó cntChar sẽ được tính bằng cách đọc buffer với kích thước len ở fileID.
- Bước 5: Nếu cntChar <= 0 thì báo lỗi file rỗng, nếu không thì, buffer[cntChar] = 0 và đưa dữ liệu buffer vào vùng nhớ ảo bufferAddress với kích thước là cntChar.
- Bước 6: Return cntChar.

5. Cài đặt SystemCall Write:

- Mô tả cài đặt **SC_Write**:
  - Input: char* buffer, int size, OpenFileID id.
  - Output: Số lượng kí tự ghi thành công.
  - Used: Ghi các kí tự trong buffer vào file.
- **Ý tưởng:** Tương tự như SC_Read nhưng fileID = 0 thì lỗi do là consoleIN, fileID = 1 thì cài đặt tương tự PrintString.
- **Cách cài đặt:** Khá tương đồng cách cài đặt ở SC_Read.
  - Bước 1: Tương tự bước 1 của SC_Read.
  - Bước 2: Khởi tạo cntChar = -1.
  - Bước 3: Nếu fileID = 0 thì kết thúc return cntChar = -1. Nếu fileID = 0 thì ta bỏ qua vì đây là consoleIN.Nếu fileID = 1, chuyển sang chế độ ghi dữ liệu output lên console Các trường hợp còn lại thì ta tiếp tục ở bước 4.
  - Bước 4: Lấy dữ liệu từ vùng nhớ bufferAddress, độ dài len và gán vào char* buffer (Dùng hàm User2System) và xác định file mở để ghi thông qua fileID.
  - Bước 5: Kiểm tra kiểu file mở, nếu mở để chỉ đọc thì báo lỗi return -1.
  - Bước 6: Ghi dữ liệu từ buffer vào file nếu hợp lệ, ngược lại báo lỗi.
  - Bước 7: Return cntChar.

6. Cài đặt SystemCall Seek:

- Mô tả cài đặt **SC_Seek**:
  - Input: int position, OpenFileID id.
  - Output: Vị trí trong file.
  - Used: Đặt con trỏ file tại vị trí posiotion ở fileID = id.
- **Ý tưởng:** Kiểm tra nếu position = -1 thì trả về cuối file. Nếu position không nằm trong khoảng [0, lenFileSpace] thì lỗi và position = -1 (Trả về cuối file).
- **Cách cài đặt:**

- Bước 1: Đọc từ register4 và register5 lần lược là pos và fileID. Kiểm tra tính hợp lệ của fileID. Nếu không hợp lệ thì lỗi.
- Bước 2: Mở fileID, kiểm tra con trỏ file có hợp lệ không, nếu không thì lỗi.
- Bước 3: Nếu pos = -1, thì pos = lenFileSpace (cuối file).
- Bước 4: Nếu pos không nằm trong khoảng [0, lenFileSpace] thì báo lỗi và pos = -1. Nếu không thì đặt con trỏ file ở vị trí pos.
- Bước 5: Return pos.

7. Cài đặt SystemCall Remove:

- Mô tả cài đặt **SC_Remove**:
  - Input: char* filename
  - Output: Thành công: 1, Lỗi: 0
  - Used: Xóa một file
- **Ý tưởng:** Chuyển dữ liệu filename từ system to user, sau đó ta chuyển dữ liệu từ system to user, kiểm tra xem filename có đang mở hay không, nếu đang mở thì lỗi, nếu không thì xóa file.
- **Cách cài đặt:**
  - Bước 1: Sau khi người dùng nhập filename vào từ bàn phím, chuyển đổi vùng nhớ từ System thành User (dùng hàm System2User).
  - Bước 2: Kiểm tra xem file có đang mở hay không, nếu có thì báo lỗi. Sau đó chuyển đến bước 4.
  - Bước 3: Xóa file, xóa file không thành công thì lỗi.
  - Bước 4: Trả về giá trị thành công hoặc không thành công.

CÁC CHƯƠNG TRÌNH THAO TÁC VỚI FILES

1. Chương trình createfile:

- **Yêu cầu chương trình:** Viết chương trình **createfile** để kiểm tra SystemCall Create. Bạn sẽ dùng tên file cố định, hoặc cho người dùng nhập vào từ console từ ReadString.
- **Cách cài đặt:**
  - Bước 1: Cho người dùng nhập vào tên file.
  - Bước 2: Kiểm tra tên file đã tồn tại hay chưa. Nếu đã tồn tại thì cho người dùng nhập tên file khác.
  - Bước 3: Tạo file.
- **Hình ảnh demo chương trình:**

```
(base) giakhiem@pzakhim:~/nachos/NachOS-4.0/code/build.linux$ ./nachos -x ../test/createfile
Nhap ten file muon tao: helloNachos4.txt

Tao file thanh cong !!Creating file 'helloNachos4.txt' successful !!

 Shutdown, initiated by user program.Machine halting!

Ticks: total 1773616881, idle 1773614713, system 2100, user 68
Disk I/O: reads 0, writes 0
Console I/O: reads 17, writes 46
Paging: faults 0
Network I/O: packets received 0, sent 0
```

Hình 1. Chương trình createfile thành công

```
(base) giakhiem@pzakhim:~/nachos/NachOS-4.0/code/build.linux$ ./nachos -x ../test/createfile
Nhap ten file muon tao: helloNachos4.txt
< ERROR > File da ton tai !! Vui long nhap ten khac
Nhap ten file muon tao:
```

Hình 2. Chương trình createfile khi người dùng nhập tên file đã tồn tại

2. Chương trình cat:

- **Yêu cầu chương trình:** Viết chương trình **cat** , yêu cầu nhập filename, rồi hiển thị nội dung của file đó.
- **Cách cài đặt:**
  - Bước 1: Cho người dùng nhập vào tên file.
  - Bước 2: Mở file và lấy độ dài của file bằng cách seek đến vị trí cuối file. Nếu mở file không thành công thì kết thúc.
  - Bước 3: Sau khi lấy được len thì ta đọc file với độ dài là của chuỗi đọc được là len và in ra màn hình.
- **Hình ảnh demo chương trình:**

```
(base) giakhiem@pzakhim:~/nachos/NachOS-4.0/code/build.linux$ ./nachos -x ../test/cat
Nhap ten file ban muon doc: cat.txt
cat.txt
Noi dung file:
Thay oi cho em 10 diem nha !
Chung em cam on thay nhieu a !!! ^^

 Shutdown, initiated by user program.Machine halting!

Ticks: total 274559551, idle 274555282, system 4140, user 129
Disk I/O: reads 0, writes 0
Console I/O: reads 8, writes 116
Paging: faults 0
Network I/O: packets received 0, sent 0
```

Hình 3. Chương trình cat đọc nội dung của file cat.txt

```
(base) giakhiem@pzakhim:~/nachos/NachOS-4.0/code/build.linux$ ./nachos -x ../test/cat
Nhap ten file ban muon doc: weloveNachos.txt
< ERROR > Khong the mo file de doc !!

 Shutdown, initiated by user program.Machine halting!

Ticks: total 971301259, idle 971298482, system 2730, user 47
Disk I/O: reads 0, writes 0
Console I/O: reads 17, writes 65
Paging: faults 0
Network I/O: packets received 0, sent 0
```

Hình 4. Chương trình cat đọc file không tồn tại

3. Chương trình copy:

- **Yêu cầu chương trình:** Viết chương trình **copy**, yêu cầu nhập tên file nguồn và file đích và thực hiện copy
- **Cách cài đặt:**
  - Bước 1: Cho người dùng nhập vào src[3] và dst[4].
  - Bước 2: Mở src, nếu không mở được thì lỗi và kết thúc.
  - Bước 3: Tính độ dài file của src, và đọc src như chương trình cat. Sau đó ta sẽ mở dst.
  - Bước 4: Nếu không mở dst được thì tạo một dst mới, nếu tạo không được thì dừng chương trình.
  - Bước 5: Ghi dữ liệu đã đọc từ scr sang dst và kết thúc chương trình.
- **Hình ảnh demo chương trình:**

[3] File nguồn.
[4] File đích.

```
(base) giakhiem@pzakhim:~/nachos/NachOS-4.0/code/build.linux$ ./nachos -x ../test/copy
Nhap ten file nguon: cat.txt
Nhap ten file dich: helloNachos4.txt

len: 64
Copy du lieu thanh cong !!

 Shutdown, initiated by user program.Machine halting!

Ticks: total 908692206, idle 908688664, system 3400, user 142
Disk I/O: reads 0, writes 0
Console I/O: reads 25, writes 77
Paging: faults 0
Network I/O: packets received 0, sent 0
```

Hình 5. Chương trình copy từ file nguồn vào file tồn tại

```
(base) giakhiem@pzakhim:~/nachos/NachOS-4.0/code/build.linux$ ./nachos -x ../test/copy
Nhap ten file nguon: cat.txt
Nhap ten file dich: helloNachos.txt

len: 64

Tao file dich thanh cong !!
Copy du lieu thanh cong !!Creating file 'helloNachos.txt' successful !!

 Shutdown, initiated by user program.Machine halting!

Ticks: total 1481742862, idle 1481738384, system 4310, user 168
Disk I/O: reads 0, writes 0
Console I/O: reads 24, writes 105
Paging: faults 0
Network I/O: packets received 0, sent 0
```

Hình 6. Chương trình copy từ file nguồn vào file không tồn tại

4. Chương trình delete:

- **Yêu cầu chương trình:** Viết chương trình **delete** để kiểm tra SystemCall Remove.
- **Cách cài đặt:**
  - Bước 1: Cho người dùng nhập tên file muốn xóa.
  - Bước 2: Xóa file và kết thúc chương trình.
- **Hình ảnh demo chương trình:**

```
(base) giakhiem@pzakhim:~/nachos/NachOS-4.0/code/build.linux$ ./nachos -x ../test/delete
Nhap ten file can delete: helloNachos4.txt

Xoa file thanh cong.

 Shutdown, initiated by user program.Machine halting!

Ticks: total 1531796833, idle 1531794652, system 2130, user 51
Disk I/O: reads 0, writes 0
Console I/O: reads 17, writes 47
Paging: faults 0
Network I/O: packets received 0, sent 0
```

Hình 7. Chương trình delete một file tồn tại

```
(base) giakhiem@pzakhim:~/nachos/NachOS-4.0/code/build.linux$ ./nachos -x ../test/delete
Nhap ten file can delete: weloveNachos.txt

Xoa file khong thanh cong.

 Shutdown, initiated by user program.Machine halting!

Ticks: total 623857033, idle 623854652, system 2330, user 51
Disk I/O: reads 0, writes 0
Console I/O: reads 17, writes 53
Paging: faults 0
Network I/O: packets received 0, sent 0
```

Hình 8. Chương trình delete một file không tồn tại

5. Chương trình concatenate:

- **Yêu cầu chương trình:** Viết chương trình **concatenate** để nối nội dung của 2 file, yêu cầu nhập tên file nguồn 1 và file nguồn 2.
- **Cách cài đặt:**
  - Bước 1: Cho người dùng nhập vào src1[5] và src2[6].
  - Bước 2: Mở src1, nếu mở không được thì kết thúc chương trình. Nếu không thì đọc dữ liệu src1.
  - Bước 3: Mở src2, nếu mở không được thì kết thúc chương trình. Nếu không thì đọc src2.
  - Bước 4: Tạo result[7], nếu tạo thất bại thì kết thúc chương trình. Nếu không thì viết lần lược nội dung scr1 và src2 vào result và kết thúc chương trình.
- **Hình ảnh demo chương trình:**

```
(base) giakhiem@pzakhim:~/nachos/NachOS-4.0/code/build.linux$ ./nachos -x ../test/concatenate
Nhap ten file nguon 1: file1.txt
Nhap ten file nguon 2: file2.txt



Noi file thanh cong
Creating file 'resultCon.txt' successful !!

 Shutdown, initiated by user program.Machine halting!

Ticks: total 909491154, idle 909487104, system 3040, user 1010
Disk I/O: reads 0, writes 0
Console I/O: reads 20, writes 68
Paging: faults 0
Network I/O: packets received 0, sent 0
```

Hình 9. Chương trình concatenate nối 2 file tồn tại

[5] File nguồn thứ nhất.
[6] File nguồn thứ hai.
[7] File lưu lại kết quả nối của 2 file nguồn.

```
(base) giakhiem@pzakhim:~/nachos/NachOS-4.0/code/build.linux$ ./nachos -x ../test/concatenate
Nhap ten file nguon 1: helloNachos.txt
Nhap ten file nguon 2: weloveNachos.txt

<ERROR> Mo file nguon 2 that bai !!

 Shutdown, initiated by user program.Machine halting!

Ticks: total 1060095446, idle 1060091514, system 3840, user 92
Disk I/O: reads 0, writes 0
Console I/O: reads 33, writes 82
Paging: faults 0
Network I/O: packets received 0, sent 0
```

Hình 10. Chương trình concatenate nối 1 trong 2 file không tồn tại

TÀI LIỆU THAM KHẢO

[1] Professor: Alan Smith, CS 162 Nachos Tutorial and Source Code, spring 2007.

[2] From FIT HCMUS, How to Install Nachos on Ubuntu 18.04.