CẤU HÌNH YÊU CẦU VÀ CÁC BƯỚC ĐỂ CÀI ĐẶT NACHOS

* Yêu cầu để cài đặt : Ubuntu 14.04 32-bit trên Linux(ưu tiên Redhat 9 hoặc Fedora core 3)
* Các bước cài đặt nachos:
  1. Cài đặt gói gcc/g++ bằng cách vào terminal và nhập lệnh ‘sudo apt -get install gcc’ và ‘sudo apt-get install g++’
  2. Unzip nachos.zip đã cho trên moodle vào một thư mục để xử lý trong thư mục đó
  3. Mở thư mục code và sửa file Makeup (‘gmake’ thành ‘make’) trong nachos/nachos-3.4/code
  4. Trong terminal nhập đường dẫn vào thư mục ‘code’ vào và dùng lệnh ‘make’ để biên dịch file
  5. Chạy thử chương trình :…/nachos-3.4/test/halt

\*Bỏ qua bước cài đặt cross compiler vì trong file đã cho nachos đã được cấu hình cross-compiler-mips

TÌM HIỂU CODE CHO TRƯỚC

* progtest.cc: kiểm tra thủ tục để biết rằng Nachos có thể load chương trình người dùng và thực thi nó. Khi khởi động nachos và yêu cầu chạy một chương trình bằng nachos -x, nó sẽ bắt đầu một quá trình thử sử dụng hàm *StartProcess*  trong protest.cc, ngoài ra còn chứa một số hàm có ích cho việc I/O
* syscall.h: Lưu các syscall mà người dùng có thể gọi, trong đó có các syscall cho việc thao tác*(SC\_Exit, SC\_Exec, SC\_Join)*, các syscall cho việc thao tác trên tập tin *(SC\_CreateFile, SC\_Open, SC\_Read, SC\_Write, SC\_Close)* và syscall cho multithreading *(SC\_Fork, SC\_Yield)*
* exception.cc: xử lý các syscall và các exception khác như lỗi, trong phần note còn có kèm quy định sử dụng thanh ghi:
  + Thanh ghi r2: chứa system call
  + Thanh ghi r4: chứa tham số thứ nhất
  + Thanh ghi r5: chứa tham số thứ hai
  + Thanh ghi r6: chứa tham số thứ ba
  + Thanh ghi r7: chứa tham số thứ tư
  + Kết quả thực hiện của system call cũng sẽ được lưu ở thanh ghi thứ 2
* bitmap.\*: gồm các thủ tục để quản lý bitmap trong đó gồm các hàm để thao tác trên bitmap(*Bitmap,Mark,Clear,Test,Find,NumClear,Print)*, bên cạnh đó còn lưu 2 hàm để đọc/ viết bitmap từ 1 file nachos*(FetchFrom,WriteBack)*
* filesys.\*:
  + gồm các thủ tục để quản lý hoạt động của hệ thống file sao cho:
    - Mỗi file gồm:
      * 1 file header(được lưu ở 1 sector trên đĩa)
      * Số lượng của khối dữ liệu
      * Một cổng vào thư mục hệ thống file.
    - Một hệ thống file bao gồm nhiều cấu trúc dữ liệu:
      * Cấu trúc dữ liệu bitmap với các disk sector trống
      * Cấu trúc dữ liệu thư mục với tên file và file headers.
  + Đối với các quá trình (Như là *Create/Remove/…*) điều chỉnh thư mục cùng hoặc không cùng với bitmap, nếu mà quá trình đó thành công, các thay đổi sẽ được ghi thẳng vào ổ đĩa trong khi các file vẫn còn mở. Nếu quá trình thay đổi thất bại do có vấn đề phát sinh, mặc dù chúng ta đã chỉnh sửa một phần của thư mục/bitmap, những thay đổi đó sẽ được thay xóa mà không lưu vào ổ đĩa
* openfile.h: gồm cấu trúc dùng để mở,đóng,đọc và viết trên một file
* translate.\*: Gồm các thủ tục để dịch địa chỉ ảo sang địa chỉ vật lý bằng cách đọc/viết từng bytes từ/vào bộ nhớ ảo bằng cách sử dụng một trong hai phương pháp ‘Linear Page Table’ hoặc là ‘Translation Lookaside buffer’, nếu có vấn đề phát sinh trong khi dịch thì sẽ RaiseException để xử lý
* machine.\*: Gồm cấu trúc dữ liệu để mô phỏng việc thi hành của các chương trình của người dùng đang chạy trên nachos, có thể hiện tình trạng của CPU và debug khi có phát sinh xảy ra
* mipssim.cc: Mô phỏng lại tập lệnh của MIPS R2/3000 processor
* console.\* gồm các cấu trúc dữ liệu dùng để mô phỏng lại hành vi của một thiết bị cuối cùng để nhập/xuất bao gồm một bàn phím và một màn hình bằng cách sử dụng UNIX files
* synchconsole.\*: Gồm cấu trúc dữ liệu chứa các hàm dùng để quản lý việc nhập/xuất trên Nachos
* …/test/\*:
  + halt.c: Dừng hệ điều hành lại
  + shell.c: Tạo shell
  + matmult.c:Tạo 3 ma trận a b c với kích thước [20][20] và nhân chúng lại với nhau
  + sort.c: tạo 1 mảng 1 chiều a[1024] với các giá trị từ 0-1024 bị xếp ngược, ta xắp xếp lại sử dụng biến tạm thời tmp
  + start.c: chạy chương trình