INSTALLING AND BUILDING NACHOS

Recommend: to use Linux distro such as: Redhat 9, Fedora core 3 or CentOS 6.0

1. Nachos Introduction:

- ❖ Nachos is instructional software for teaching undergraduate, and potentially graduate, level operating systems courses. The Nachos distribution comes with: an overview paper simple baseline code for a working operating system
- ❖ Nachos simulates a machine with a processor, registers, memory that roughly approximates the MIPS architecture. The simulated MIPS processor can execute arbitrary user programs. Nachos has two modes of execution, one of which is the MIPS simulator. The second mode corresponds to the Nachos “kernel”.
- ❖ The basic NachOS is designed as a single tasking operating system.
- ❖ NachOS subdirectory structure:
  - filesys/ : holds implementation of both stub and real file system
  - lib/ : holds the class library and debug routines
  - machine/ : holds the MIPS simulator that NachOS uses to execute the user program
  - network/ : includes the networks code for NachOS
  - test/ : holds all the test cases and environment to build new test cases. Follow the format in Makefile to make new tests
  - threads/ : holds the threading and related routines for NachOS
  - userprog/ : holds the beginnings of user program support and system calls handling
- ❖ Once Nachos is installed, you should try compiling it. The code for the nachos program itself. This program is written in C++ and implements the machine simulation and the operating system for the simulated machine. The second is a set of test programs in /code/test directory. These programs are written in C. They run on the simulated machine and are used to test its operating system. The third is the coff2noff program for converting a COFF (Common Object File Format) file to a NOFF file (Nachos Object File Format)
- ❖ The Nachos kernel and machine simulator run directly on the host machine, but user-level programs running on top of Nachos are simulated instruction-by-instruction. The simulator assumes MIPS object code, in little endian format. It would take much more work to complete the port and change the CPU simulator to simulate other instruction sets. Therefore, you need a cross-compiler. The cross-compiler runs on Linux and compiles user programs into MIPS format. GCC has been fixed to support this.

2. How to install and build NachOS:

Solution 1:

a. Cross-compiler and NachOS:

- Refer to: http://en.wikipedia.org/wiki/Cross-compiler
- A *cross-compiler* is a compiler that runs on one architecture but compiles to another architecture, e.g. a compiler that runs on x86 machines but builds MIPS programs.

b. Install Cross-compiler

Requirement: gcc-3.x was installed on Linux machine

Now we start with the current directory is **/root**

Step 1: Create a home directory for this assignment

% mkdir hdh

Download and copy binutils, gcc packages into **hdh** directory (should use WinSCP)

binutils-2.11.2.tar.gz

gcc-2.95.3.tar.gz

Extract these packages:

% cd hdh

% tar -xzvf binutils-2.11.2.tar.gz

% tar -xzvf gcc-2.95.3.tar.gz

Step 2: Create folder that the Cross-compiler will be installed

% mkdir cross-compiler

Build binutils first:

% cd binutils-2.11.2/

% ./configure --host=i686-pc-linux-gnu --target=decstation-ultrix --prefix=/root/hdh/cross-compiler

% make

% make install

You can see some new directories and files in /root/hdh/cross-compiler/decstation-ultrix/bin/ directory like that:

ar as ld nm ranlib strip

Step 3: Building gcc

% cd ..

% mkdir gcc-obj

Create a stub directory system-include:

% mkdir cross-compiler/decstation-ultrix/sys-include

In gcc-obj directory, modifying gcc configuration:

% cd gcc-obj

% ../gcc-2.95.3/configure --host=i686-pc-linux-gnu --target=decstation-ultrix --prefix=/root/hdh/cross-compiler --with-gnu-as --with-gnu-ld --with -

as=/root/hdh/crosscompiler/decstation-ultrix/bin/as --with -ld=/root/hdh/crosscompiler/decstation-ultrix/bin/ld --enable-languages=c –disable-multilib --disable-libgcj

Compile gcc:

```
% make
```

It's ok if there is an error in building libgc2, you don't need any of the libraries anyway. So just go ahead and install

```
% make -k install
```

After installing gcc successfully, you can see new file named **gcc** in /root/hdh/cross-compiler/decstation-ultrix/bin/ directory

c. Installing and building Nachos:

Requirement: already installed gcc-2.x or gcc-3.x on Linux and cross-compiler

Step 1: Download and extract the NachOS source code

Copy the following file into hdh directory

nachos-3.4.tar.gz

Now start with the current directory is **/root**

```
% cd hdh
% tar -xzvf nachos-3.4.tar.gz
```

Step 2: Modifying the path to Cross-compiler in Makefile

You can copy two folders: cross-compiler/decstation-ultrix/ and Nachos nachos-3.4 to another machine without recompile cross-compiler

```
# if you are cross-compiling, you need to point to the right
# executables and change the flags to ld and the build
# procedure for as
GCCDIR = ../../../cross-compiler/decstation-ultrix/bin/
LDFLAGS = -T script -N
ASFLAGS = -mips2
CPPFLAGS = $(INCDIR)
# if you aren't cross-compiling:
# GCCDIR =
# LDFLAGS = -N -T 0
# ASFLAGS =
# CPPFLAGS = -P $(INCDIR)
PATH = $(GCCDIR):/lib:/usr/bin:/bin
CC = $(GCCDIR)gcc -B../../../cross-compiler/
AS = $(GCCDIR)as
LD = $(GCCDIR)ld
CPP = gcc -E
INCDIR =-I../userprog -I../threads
CFLAGS = -G 0 -c $(INCDIR)
```

Compile NachOS

```
% cd nachos-3.4/code
% gmake all
```

d. Run the first test program:

Try asking Nachos to run a very simple test program called halt, which is found in the code/test directory. Type:

```
% ./userprog/nachos –rs 1023 –x ./test/halt
```

The halt program simply makes the "Halt" system call, which causes the machine simulator to halt and print out some statistics like this:

```
Machine halting!
Ticks: total 42 idle 0, system 30, user 12
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…
```

Solution 2:

Use the ready compiled directory of cross-compiler.

a. Copy 2 below directory into the same directory:

nachos-3.4

gnu-decstation-ultrix

b. Modifying file Makefile in nachos-3.4/code/test/

```
# procedure for as
    GCCDIR = < the fullpath to gnu-decstation-ultrix directory >/decstation-ultrix/bin
    LDFLAGS = -T script -N
    ASFLAGS = -mips2
    CPPFLAGS = $(INCDIR)
    # if you aren't cross-compiling:
    # GCCDIR =
    # LDFLAGS = -N -T 0
    # ASFLAGS =
    # CPPFLAGS = -P $(INCDIR)
    PATH = $(GCCDIR):/lib:/usr/bin:/bin
    CC = $(GCCDIR)gcc -B <the fullpath to gnu-decstation-ultrix directory>
    AS = $(GCCDIR)as
    LD = $(GCCDIR)ld
```

CPP = gcc -E
INCDIR =-I../userprog -I../threads
CFLAGS = -G 0 -c $(INCDIR)

c. **Compile Nachos:**
Move to nachos-3.4/code directory. Executing the following command:
% gmake all

❖ **The next task:**
- Research how to add a new system call into NachOS
- Read detailed about NachOS source subdirectories to implement more complex components.