

Kernel Assignments

CS 256/456
Dept. of Computer Science, University of Rochester

2/9/2005 CSC 256/456 - Spring 2005 1

Introduction to Nachos: Our Objective

- Provide a basic understanding of the structure of Nachos
 - not a replacement of your own practice and learning
- Jump start you on Programming Assignment #3

2/9/2005 CSC 256/456 - Spring 2005 2

Motivation for Nachos

- Motivation: need good projects for teaching operating systems
- Assignments so far are all at user level
 - OK with understanding OS concepts
 - Not real OS development experience
- Develop an OS from scratch on a real hardware (e.g., Intel IA32)
 - Real hardware is too complex to work with
 - Very hard to support even the most basic user programs
- Making enhancement/adjustment of an existing OS (e.g., Linux)
 - Modern OSes are too complex to understand
 - Not complete view about OS development
- Nachos is a toolkit for teaching operating systems; not an operating system itself

2/9/2005 CSC 256/456 - Spring 2005 3

Overall Nachos Structure

The diagram illustrates the overall Nachos structure. On the left, 'Normal computer structure' shows a flow from 'user programs' through 'System calls' to the 'OS kernel', which then interacts with 'I/O devices' and 'CPU & mem' on the 'Hardware machine'. On the right, 'Nachos structure' shows 'Nachos user programs' interacting with the 'Nachos kernel' via 'Nachos syscalls'. The 'Nachos kernel' runs on a 'Host OS kernel' which interacts with the 'Host hardware machine' (I/O devices, CPU & mem) via 'Host Syscalls'. The 'Nachos kernel' also interacts with 'I/O devices' and 'MIPS CPU & mem' on 'Simulated Nachos HW'. 'Host user programs' are also shown running on the host OS kernel.

2/9/2005 CSC 256/456 - Spring 2005 4

Nachos Kernel and User Programs

The diagram illustrates the Nachos structure. At the top, 'Nachos user programs' are shown in a box. Below them, 'Nachos syscalls' and 'Nachos kernel' are connected by a bidirectional arrow. The 'Nachos kernel' is further connected to 'I/O devices' and 'MIPS CPU & mem', which are grouped as 'Simulated Nachos HW'. Below this, 'Host Syscalls' and 'Host OS kernel' are connected by a bidirectional arrow. The 'Host OS kernel' is connected to 'I/O devices' and 'CPU & mem', which are grouped as 'Host hardware machine'. Arrows indicate the flow of control and data between these layers.

- The Nachos kernel runs directly as a user program on the host machine.
- Kernel code written in C++.
- User programs run on the Nachos kernel and the simulated hardware (with a MIPS CPU).
- User programs access simulated I/O devices through Nachos syscalls.
- User programs written in a stripped-down C, compiled into MIPS executables.

Nachos structure

2/9/2005 CSC 256/456 - Spring 2005 5

Simulated Nachos Hardware

This diagram is identical to the one on slide 5, showing the Nachos structure. It highlights the 'Simulated Nachos HW' components: 'I/O devices' and 'MIPS CPU & mem'.

- Simulated MIPS CPU
 - A MIPS instruction interpreter
- Simulated memory
 - Basically an array of words
- Simulated I/O devices
 - Console terminal
 - Disk
 - Timer
 - *Network interface*

Nachos structure

2/9/2005 CSC 256/456 - Spring 2005 6

Nachos as An Instructional Tool

- The Nachos toolkit contains:
 - The simulated hardware (MIPS CPU, memory, I/O devices)
 - A cross-compiler/linker that compiles your C user programs into Nachos executables
 - instructions are in MIPS
 - the segment layout and header format can be recognized by the Nachos kernel
 - A skeleton OS kernel supporting
 - a single system call: halt
 - limited synchronization primitives
 - running a single user program at a time
 - no inter-process communication
- Your job is to augment the OS step by step
 - don't touch the simulated hardware
 - shouldn't need to touch the compiler

Nachos structure

2/9/2005 CSC 256/456 - Spring 2005 7

The Most Common Confusion

- Nachos kernel and Nachos user programs do not run on the same machine (real or virtual)
- Implications:
 - they don't run on the same CPU (real or virtual)
 - they don't share the same memory (real or virtual)
- This is a big weakness
 - why not let Nachos kernel run on the simulated hardware?
 - the projects become too challenging
 - a pain to develop code on the simulated hardware, e.g., can't run gdb

Nachos structure

2/9/2005 CSC 256/456 - Spring 2005 8

Nachos Assignments

Start from a given skeleton OS, filling the missing pieces step by step

- Assignment #3: threads and synchronization
- Assignment #4: supporting multiple user programs
- Assignment #5: virtual memory
- Assignment #6: file system and disk scheduling

2/9/2005

CSC 256/456 - Spring 2005

9

Assignment #3: Threads and Synchronization

- Become familiar with Nachos and the behavior of a working (but incomplete) thread system
- Use what is supplied to study concurrent programming
- Threads you use for this assignment are:
 - Nachos threads in kernel
 - user-level threads from the host machine's perspective
- Synchronization primitives are used for synchronization between Nachos threads in kernel

2/9/2005

CSC 256/456 - Spring 2005

10

Learning about Nachos

- You cannot learn all about software systems from textbooks
 - read the source code for systems that other people have written
- For Nachos:
 - it is not sufficient to just read the textbook, lectures, and information on the assignment Web pages
 - read over the Nachos source code
 - try to understand where the various pieces of the system live, and how they fit together
- It will take a while to develop an understanding. Start early and be patient!

2/9/2005

CSC 256/456 - Spring 2005

11

Traversing the Nachos Files

- starting from the Makefiles
- threads/
 - support threads and synchronization
 - the threads support is fully functional, though some of the synchronization primitives have not been implemented.
- userprog/
 - support the loading and execution of user programs
 - basic memory management
- vm/
 - virtual memory subsystem

2/9/2005

CSC 256/456 - Spring 2005

12

Traversing the Nachos Files

- test/
 - some simple Nachos user programs
 - Makefile for compiling these programs and converting them to NOFF
- machine/
 - support machine simulation.
 - You need to read some header files to know how to access the simulated machine.
 - It might also be instructive to look at the implementation of the machine simulation.
 - But **you shouldn't have to modify** anything here.

2/9/2005

CSC 256/456 - Spring 2005

13

Administrative Issues

- group assignments
 - group of two or three
 - don't form the same group for all four assignments
- C/C++ programming
 - high-level languages (e.g., Java, perl) are not suitable for operating system development
- managing TA for each assignment
- start early
 - you can't cut corners on this
 - mandatory meeting with the TA before the due time

2/9/2005

CSC 256/456 - Spring 2005

14

An alternative Assignment Track

- Nachos-based assignments:
 - Nachos is specifically designed for OS course projects.
 - The kernels you develop on it are not "real".
- Xen-based assignments
 - Xen, on the other hand, is a "true" virtual machine.
 - You will be working with slightly modified Linux on it.
- Why not choose Xen?
 - More realistic, but much more challenging (the Nachos assignments will be challenging enough).
 - The first time we introduce the Xen assignments so there might be problems and you need to be very patient with us.

2/9/2005

CSC 256/456 - Spring 2005

15

Xen: An "True" Virtual Machine

- Real operating systems (with slight modifications) can run on Xen.
 - we use a modified Linux 2.6.10
- For Xen assignments:
 - you will spend most of your time reading, understanding the Linux code.
 - and you will spend more time on Linux administration.
 - even debugging becomes very challenging.
 - you will get help from us, but you are expected to figure out lots of things on your own.

2/9/2005

CSC 256/456 - Spring 2005

16

Xen Assignments

Make adjustment to a fully functional kernel

- Assignment #3: write a system call
- Assignment #4: threads and synchronization
- Assignment #5: CPU scheduling
- Assignment #6: memory management