

EECS 211: Advanced System Software Lecture 5

Rainer Dömer

doemer@uci.edu

The Henry Samueli School of Engineering
Electrical Engineering and Computer Science
University of California, Irvine

Lecture 5: Overview

- Operating Systems Review
 - Prerequisite Quiz Discussion
- The Nachos System
 - Introduction
 - Overview
- Assignment 1
 - Introduction to Nachos, Threads
- Assignment 2
 - Concurrency and Synchronization

Operating Systems Review

- Prerequisite Quiz Discussion
 - Results
 - Overall quite positive
 - Most seem to be well-prepared
 - Some may need improvement in C/C++ programming...
 - Solution
 - `PrerequisiteQuiz_Solution.pdf`

The Nachos System

- Introduction
 - *“Not Another Completely Heuristic Operating System”*
 - Instructional operating system
 - designed by Th. Anderson, UC Berkeley (in early 90's)
 - designed for teaching (undergraduate level!)
 - Simple, but working system
 - Concepts are learned by hands-on experimentation
 - Covers most significant concepts of a modern OS
 - threads and process synchronization
 - multi-programming
 - file systems
 - virtual memory
 - networking
 - Usable in regular Unix environment
 - Well-documented source code freely available

The Nachos System

- Documentation
 - Text book, 7th edition, Appendix D
 - Local copy available on course web site
 - Source code (!)
 - Well-commented C/C++ code
 - Additional online resources
 - Nachos home page
 - Nachos roadmap
 - Wikipedia entry
- Why not Linux?
 - Huge size and complexity
 - Development and test environment (“naked PCs”?)
 - Debugging (nightmare!)

EECS211: Advanced System Software, Lecture 5

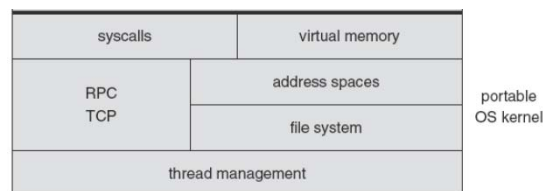
(c) 2011 R. Doemer

5

The Nachos System

- Overview

- Kernel:
 - Compiled C/C++ code
 - normal (debug'able) Unix process



EECS211: Advanced System Software, Lecture 5

(c) 2011 R. Doemer

6

The Nachos System

- Overview
 - User code:
 - Cross-compiled C/C++ code
 - emulated by MIPS simulator
 - Kernel:
 - Compiled C/C++ code
 - normal (debug'able) Unix process

The diagram illustrates the Nachos system architecture. At the top, a human icon represents the user, with two boxes labeled 'application' below it. Below the applications is a 'shell' layer, which is part of the 'user programs' category. Underneath the shell is the 'MIPS simulation' layer. The 'portable OS kernel' consists of several layers: 'syscalls' and 'virtual memory' are grouped together; 'RPC' and 'TCP' are grouped together; 'address spaces' and 'file system' are grouped together; and 'thread management' is at the bottom of this section. Below the portable OS kernel is the 'machine-dependent OS layer', which is part of the 'hardware simulation' category. At the very bottom is the 'I/O device simulation' layer. The entire system is labeled as a 'UNIX process' at the bottom.

UNIX process

EECS211: Advanced System Software, Lecture 5 (c) 2011 R. Doemer 7

The Nachos System

- Overview
 - User code:
 - Cross-compiled C/C++ code
 - emulated by MIPS simulator
 - Kernel:
 - Compiled C/C++ code
 - normal (debug'able) Unix process
 - I/O System:
 - simulated by Unix process I/O

This diagram is identical to the one on slide 7, but it includes an additional layer in the hardware simulation section. Below the 'machine-dependent OS layer' is the 'I/O device simulation' layer. The 'UNIX process' label is now positioned below the 'I/O device simulation' layer.

UNIX process

EECS211: Advanced System Software, Lecture 5 (c) 2011 R. Doemer 8

Assignment 1

- Introduction to the Nachos System
 - Task 1: Read the overview chapter
 - Text book, Appendix D (contents online)
 - Task 2: Install the software
 - Setup environment, copy tar-ball, unpack, compile, test
 - Task 3: *Understand* the Nachos system!
 - Read documents and source code
- Deliverables
 - Log output for `./nachos -rs 42`
 - What is the purpose of `SWITCH()`?
 - Why is `SWITCH()` not implemented in C/C++?
 - Why does the execution order change with `-rs` option?
- Due by email to **doemer@uci.edu**
 - Wednesday, January 26, 2011, at 2pm (sharp!)

EECS211: Advanced System Software, Lecture 5

(c) 2011 R. Doemer

9

Assignment 2

- Thread Synchronization in Nachos
 - Task 1: Implement the missing locks and condition variables
 - files `synch.h` and `synch.cc`
 - Task 2: Test the locks and condition variables
 - file `threadtest.cc`
(based on `threadtest.cc.W11templateA2`)
 - implement *safe* scheduling of strictly alternating threads
 - *no change in execution order due to any `-rs` value!*
- Deliverables
 - code for locks, condition variables, and safe test case
 - log file of test runs with five different random seeds
 - Email to **doemer@uci.edu**
- Due by email to **doemer@uci.edu**
 - Wednesday, February 2, 2011, at 2pm (sharp!)

EECS211: Advanced System Software, Lecture 5

(c) 2011 R. Doemer

10