

EECS 211: Advanced System Software Lecture 8

Rainer Dömer

doemer@uci.edu

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

Lecture 8: Overview

- Assignment 1
 - Nachos setup, threads
 - Solution
- Assignment 2
 - Concurrency, Synchronization
 - Discussion
- Review of Chapter 6
 - “Condition Variables in Monitor”
- Memory Management
 - Paging, Segmentation

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

The diagram illustrates the Nachos system architecture as a stack of layers. At the top, a human icon represents the user, with two 'application' boxes. Below these is the 'shell' layer, labeled 'user programs'. The next layer is 'MIPS simulation'. Below that is the 'portable OS kernel' layer, which is divided into 'syscalls' and 'virtual memory' on the left, and 'address spaces' and 'file system' on the right. Below the kernel is the 'thread management' layer. The next layer is the 'machine-dependent OS layer', labeled 'hardware simulation'. The bottom layer is 'I/O device simulation'. The entire stack is labeled 'UNIX process' at the base.

EECS211: Advanced System Software, Lecture 8
(c) 2011 R. Doemer
3

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 8
(c) 2011 R. Doemer
4

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 8

(c) 2011 R. Doemer

5

Operating Systems Review

- Excerpts from chapter 6 of
“Operating System Concepts”, 8th Edition,
by A. Silberschatz, P. B. Galvin, G. Gagne,
John Wiley & Sons, 2009.
- Monitors
- Condition Variables

EECS211: Advanced System Software, Lecture 8

(c) 2011 R. Doemer

6

Memory Management

- Excerpts from chapter 8 of
“Operating System Concepts”, 8th Edition,
by A. Silberschatz, P. B. Galvin, G. Gagne,
John Wiley & Sons, 2009.
- Memory Management
 - Paging
 - Segmentation