

CA321 OS Design and Implementation

Darragh O'Brien

`dobrien@computing.dcu.ie`

Office L2.35
School of Computing
Dublin City University

September 29, 2009

Course Home Page

- `http://www.computing.dcu.ie/~dobrien/ca321`
- Announcements
- Lecture notes
- Assignments
- Results
- Further reading
- Useful links

Lectures

Two 1-hour lectures per week

- Tuesday, 1500-1600, HG.22
- Friday, 1000-1100, Q1.19

Labs

One 2-hour lab per week

- Thursday, 0900-1100, L114
- Unsupervised, this time has been booked for you to work on CA321 assignments
- There are no weekly lab exercises

Marking

Continuous assessment 30%

- Usually 2 assignments
- Usually POSIX threads and Unix systems programming in C

Examination 70%

- Usually 5 questions, answer 4

How To Pass

- Your final mark is calculated by combining your continuous assessment and exam results
- To pass the module your final mark must be 40+
- Failed components must be repeated
- Repeat examinations take place in August
- Repeat continuous assessment due in August

Overview

- This module follows on from last semester's CA216 *Operating Systems*
- Since an understanding of the material covered by CA216 is assumed you may occasionally find it beneficial to look over those notes

Approach

- The OS we will focus on is Linux (time permitting we will also look at Windows Vista and/or Symbian OS)
- Knowing how issues are handled in at least one OS will help you understand how they are solved in others
- All practical work will be implemented on a Linux platform and includes some POSIX threads and Unix systems programming in C
- Your favourite Linux and C programming references will be useful (C programming skills are assumed)

Textbooks

Required reading

- *Modern Operating Systems*, 3rd edition by Tanenbaum
- *Operating Systems Internals and Design Principles*, 6th edition by Stallings

Supplementary reading

- *Operating Systems Concepts*, 7th edition by Silberschatz, Galvin and Gagne
- *Operating Systems*, 3rd edition by Nutt
- *Linux Kernel Development*, by Love
- Resources linked from the course home page

What is an Operating System?

An OS basically does two things:

1. It makes the machine easier to use
2. It attempts to share the resources on the machine fairly between users and applications

What is an Operating System?

Some definitions

- Most fundamental piece of software running on computer
- Program that runs automatically computer is switched on
- Software that provides a user-friendly interface between application programs and the hardware: it takes the pain out of dealing directly with hardware by providing a virtual programming environment
- Software that handles resource requests from application programs and that prevents applications from interfering with one another

Operating System Interfaces

- The OS sits between userland applications and hardware and thus has two interfaces: hardware and software
- Like applications make talking to the OS more user-friendly so an OS makes the hardware more user-friendly
- Applications talk to the OS by invoking the system services it provides; this is “systems programming” i.e. writing code that makes system calls (using the software interface)
- The hardware interface is where the OS meets the hardware and carries out I/O on our behalf

Why Study Operating Systems?

- There is a minimum amount of knowledge of how an OS works that is required by anyone seriously involved the study of computer science
- Knowledge gained is useful for guiding OS selection for a particular environment and for tuning OS performance

Why Study Operating Systems?

- When solving problems it is useful to be aware of services made available to you by the OS: they may help you in solving your problem
- An OS is a very large piece of software which solves many problems: you will come across similar problems later in your programming career and so will become a better programmer by studying operating systems

Sample Syllabus

- Processes, threads, scheduling
- Multiple processor systems and virtualisation
- Security
- Memory management
- Operating system design
- Case studies: Linux, Windows Vista, Symbian OS