Operating Systems SOFTENG 370

- Lecturers
  - Robert Sheehan (r.sheehan@auckland.ac.nz)
    - Rm 303.409
    - office hours - Whenever
  - Meng-Fen Chiang (meng.chiang@auckland.ac.nz)
    - Room and office hours to be announced
- Textbook
  - *Operating System Concepts: Global edition (10th edition)* – Silberschatz, Galvin and Gagne.
  - The printed textbook and E-Text can be purchased from just $65AUD from Wiley: https://www.wiley.com/en-au/Silberschatz%27s+Operating+System+Concepts%2C+10th+Edition%2C+Global+Edition-p-9781119455868
- Test (10%)
  - Thursday 3rd September 6:30pm
- Exam (60%)
  - TBA
- Three Assignments (30%)
- You have to get an overall pass (>=50%). N.B. no theory/practical requirement this year

Tutor and tutorials

- Tutor - unknown
- Tutorial - Tuesday 12 noon LibB10/109-B10
    - Start 4th of August
    - Not compulsory
    - How-to sessions or help with assignments
        - First session - setting up a virtual machine and getting it ready to program in C

What is an Operating System?

- https://www.youtube.com/watch?v=V5S8kFvXpo4 - Computer Chronicles from 1984
  - Some things to note
    - Have you heard of CP/M?
    - How long to boot a *microcomputer* in 1984?
- Examples
  - macOS
  - Windows
  - Linux
  - UNIX
  - Plan9
  - Amoeba
  - OpenVMS (Virtual Memory System)
  - VM/CMS (Conversational Monitor System)
  - z/OS (IBM)
  - Android
  - iOS
- The software which makes the computer usable.
  - It is *impossible* to use modern machines without an OS.
- The collection of software sold (or freely available) as an OS.

Are these things part of the OS?

- file system
- communication system
- process manager
- security manager
- memory manager
- graphical user interface
- backup system
- web browser
- media player
- compiler
- Java (or .Net) environment
- Virtual Machine Monitor

Extreme approaches

- Minimalist understanding

- OS software is the minimum amount of software required to allow the computer to function.
- kernel – usually in memory always
- process/thread management
- communications
- memory management
- file management
- monolithic and micro-kernels.

- Maximalist understanding

- All the software which comes with a standard release of the OS.
- many utilities and programs

Usable vs Efficient

Some OSs are designed for specific needs

- factory control systems
- aircraft control
- database servers
- phones

Others are general purpose

- desktop computers
- phones

Usable – for whom?

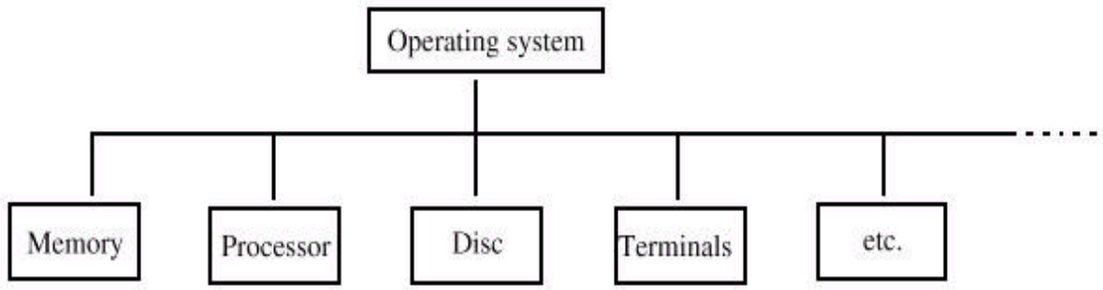
- the developer of the system
- a software engineer or computer scientist
- a data entry operator
- a person with a disability
- an “ordinary” user

Efficient

- real-time systems
- dealing with many thousands of transactions a second
- battery life

OS Themes

Manager model

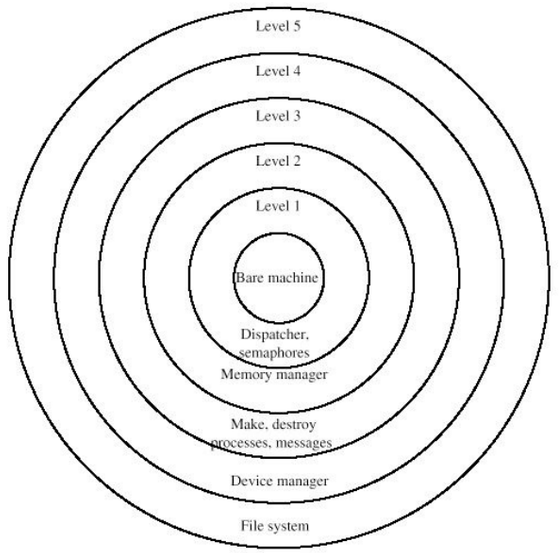


- The OS is a collection of managers.
- It prevents improper use of devices.
- Each manager is independent
    - and maintains tables of information

Onion model

https://www.youtube.com/watch?v=_bMcXVe8zIs

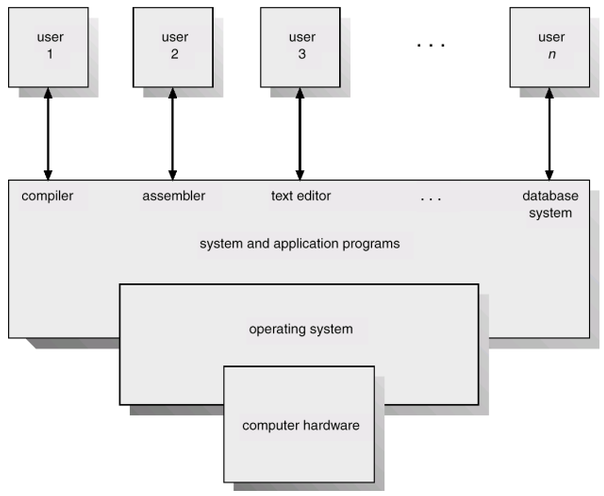
- The OS is a series of layers.
- Outer layers can access resources contained in inner layers.
- But not vice-versa.



OS Themes (cont.)

Resource allocator model

- Related to the manager model.
- The emphasis is on providing the services programs need.
- Must be fair. (Whatever that means.)



Dustbin model

- This sees the OS as all the bits no one else wants to do.

Getting work done model

- We only use computers to do something else:
- write an essay
- calculate a mortgage repayment
- find information
- download a song
- play a game
- make a phone call
- The OS has to help us to get our work done.

Things I expect you to know.

- You should have some idea of
  - interrupts
  - security & protection
  - file systems
  - virtual memory
  - processes and threads
- The first assignment will require C programming. You need to get comfortable with C and Linux.

Things you should be able to do by the end.

Describe the stages of OS development and comment on them.

Discuss different implementations of the process and thread model.
Describe different types of process scheduling and apply different scheduling algorithms. Write programs with interprocess communication.

Reason about a concurrency problem and the common concurrency constructs.
Apply a suitable construct to solve a problem involving concurrency.

Discuss different deadlock solutions.

Discuss file system operations. Able to describe the implementation of a simple file system.
Compare different file systems including distributed systems.

Compare the main methods of managing memory.
Describe in detail how virtual memory works.

Apply simple page replacement algorithms.

Discuss the main problems associated with protection and security and provide a variety of solutions.
Interpret access matrices.

Describe different methods of incorporating device drivers into OSs. Discuss low-level disk scheduling algorithms.

OS DESIGN

All in one – all OS components can freely interact with each other.

- MS-DOS
- Early UNIX

Separate layers – see the Onion model.

- This simplifies verification and debugging.
- Very hard to get the design correct.
- Can be inefficient – lots of layers to go through to get work done.
- THE
- OS/2

Modules – like the all-in-one but only loaded when necessary

- Linux, Windows

Microkernels

- Use client/server model.
- Many modern general-purpose OSs use this approach (although that doesn't mean they are microkernel OSs)
- Mach (basis for MacOS X)
- QNX RT-OS
- Exokernels - more radical microkernels

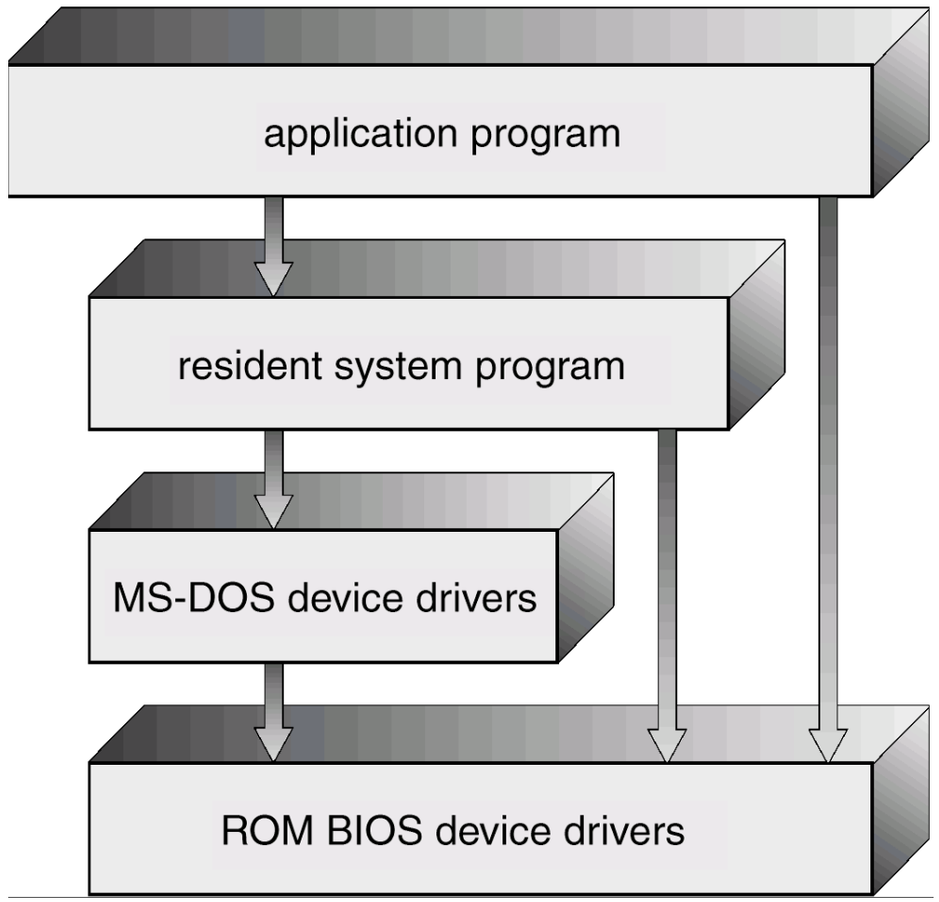
Virtual Machines

- VM/CMS
- Java

MS-DOS

Written to provide the most functionality in the least space

- not divided into modules
- Although MS-DOS had some structure, its interfaces and levels of functionality were not well separated
- This approach comes back in a modern form in exokernels.



Early UNIX

- https://www.youtube.com/watch?v=JoVQTPbD6UY - Ken Thompson and Dennis Ritchie

The UNIX OS consists of two separable parts.

- Systems programs
- The kernel
- Consists of everything below the system-call interface and above the physical hardware
- Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level.

<table>
<tr><td colspan="3">(the users)</td></tr>
<tr><td colspan="3">shells and commands
compilers and interpreters
system libraries</td></tr>
<tr><td colspan="3">system-call interface to the kernel</td></tr>
<tr><td>signals terminal handling
character I/O system
terminal drivers</td><td>file system
swapping block I/O system
disk and tape drivers</td><td>CPU scheduling
page replacement
demand paging
virtual memory</td></tr>
<tr><td colspan="3">kernel interface to the hardware</td></tr>
<tr><td>terminal controllers
terminals</td><td>device controllers
disks and tapes</td><td>memory controllers
physical memory</td></tr>
</table>

THE Multiprogramming system

A layered design was first used in the THE operating system.
Its six layers were:

layer 5:	user programs
layer 4:	buffering for input and output
layer 3:	operator-console device driver
layer 2:	memory management
layer 1:	CPU scheduling
layer 0:	hardware

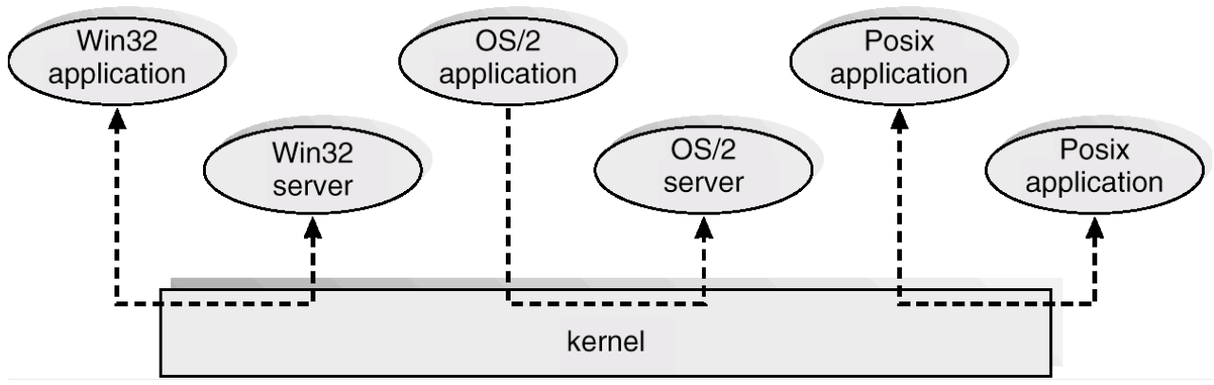
Windows NT client/server

- https://www.youtube.com/watch?v=J2GV_bCfnCw - Dave Cutler

Windows NT provided environmental subsystems to run code written to different OS APIs.

Windows NT (and successors) is a hybrid system.

- Parts are layered but some of the layers have been merged to improve performance.
- Many OS services are provided by user-level servers
  - e.g. the environmental subsystems.
- Now in Windows 10 there is the Windows subsystem for Linux.



Before the next lecture

Read textbook

- What Operating Systems Do 1.1
- Operating System Structure 2.8
- Linux 20.1, 20.2
- Windows 10 21.1, 21.2

Preparation for next time

- 1.2 Computer-System Organization
- 1.3 Computer-System Architecture
- 1.4 Operating-System Operations