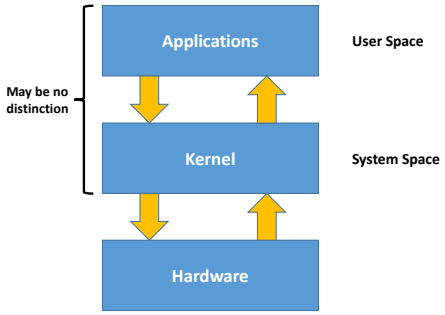
Operating System Designs

Dr Andrew Scott

a.scott@lancaster.ac.uk

1

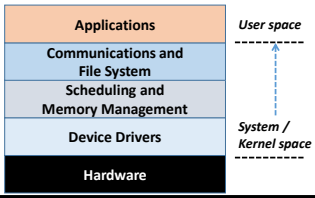
Monolithic



2

Monolithic

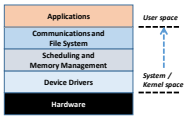
- Traditional approach
  - Tight coupling of non-application code
- ***At most*** two protection levels (system and user)



3

Monolithic

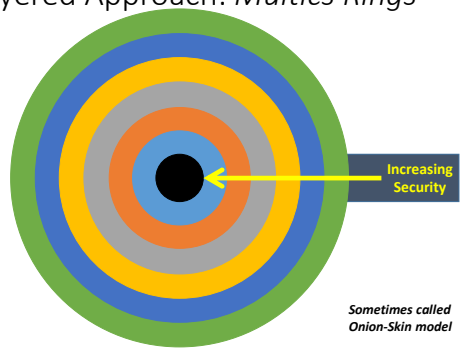
- All OS code running with full privileges
  - In some systems may not be distinct user/ system levels
- Difficult to impose security or protection
- Efficient call structure
  - Shared access to resources
- Difficult to maintain
  - Code boundaries frequently blurred



Embedded OSs, DOS based MS Windows, initially Linux

4

Layered Approach: *Multics Rings*

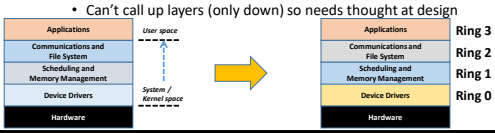


Sometimes called Onion-Skin model

5

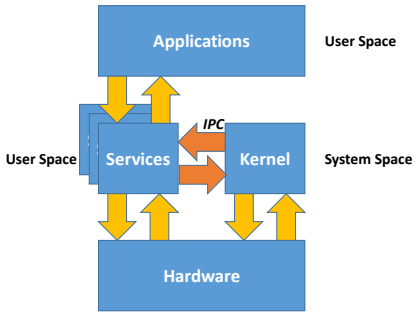
Layered Approach

- Realisation that we have identifiable layers in system
  - High-level components built on lower ones
  - Impose tighter security as move from hardware
    - **Performance penalty with every layer crossed**
  - Hard to manage if complex inter-layer call chain
    - Can't call up layers (only down) so needs thought at design



6

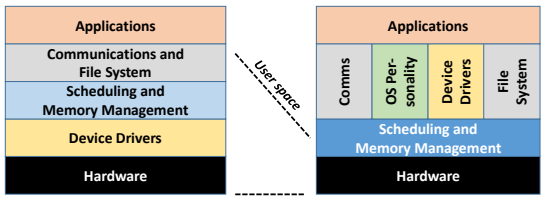
Microkernel



7

Monolithic vs. Microkernel OS

- Far less code running with elevated privileges
  - Most code subject to security constraints



8

Micro-Kernel

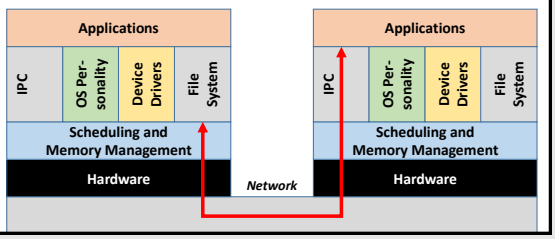
- Minimal amount of code has 'system' privileges
  - Most code in unprivileged 'user space'
  - Far more secure than other approaches
- Components communicate by message passing
  - Main function of kernel in this model is *Inter-Process Communication* (IPC)
  - Can be inefficient
  - Most real implementations break model (hybrid approach)
- Extensible and have minimal system dependencies
  - Clean APIs make porting or extending OS much easier

Mach, Minix, initially Microsoft (Windows) NT

9

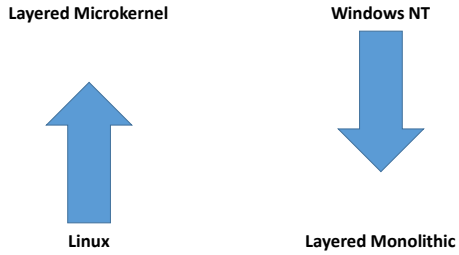
Networked Microkernel OSs

- Microkernels Inter-Process Communication based
  - Easy to extend over network (with performance penalty)



10

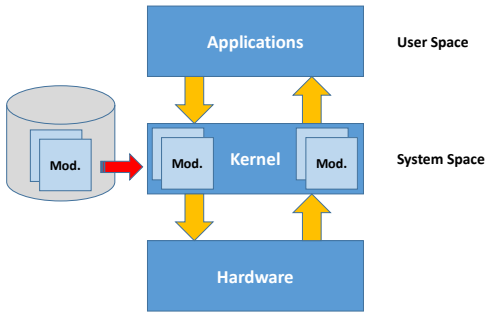
Relative Evolution of NT and Linux



c.f. Famous Tanenbaum-Torvalds Usenet debate

11

Modular



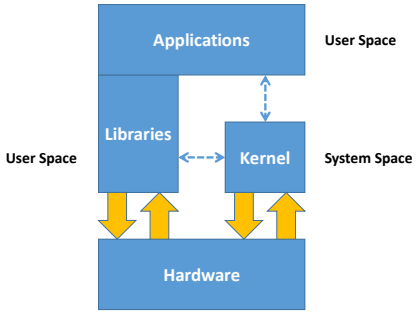
12

Modular OS

- Core kernel loads modules as needed
  - Core relatively small
    - More manageable, less scope for code problems
- Kernel modules run with elevated privileges
  - Reliant on API and structure for kernel integrity
    - Modules can still break things if they don't follow rules
  - 'Object Oriented-style' approach
- Naturally restricted to module API
  - Not limited to microkernel style message passing
  - Can access and share resources across modules

13

Library OS/ Uni-Kernels



14

Library OSs and Uni-kernels

- Kernel configures security at lowest level
  - Disk blocks, for example
- Libraries provide
  - Additional (user-/ resource-centric) security
  - OS Structure and personality
    - Could have Linux and Windows libraries on same system
- Applications more aware of hardware
  - OS personality bound into each application via libraries
  - Can better match hardware constraints, features, …
  - Less consistency in applications, APIs, etc.

Microsoft have demonstrated a fairly complete Library-OS re-implementation of Windows 7

15