OS Structure

Module 04

OS Components

As we've seen, the Operating System has many jobs:

- The **Scheduler** allocates the CPU to processes
- The **Memory Manager** ensure protection and fair allocation to processes in memory
- The **File System** provides abstractions to allow programmers to use files instead of blocks/cylinders, and the like
- Other important components include **user management** and **network communication**.

Operating Systems are *big*

- Linux and Windows are both on the order of 5,000,000 lines of code
  - Over 70,000,000 if you include device drivers!

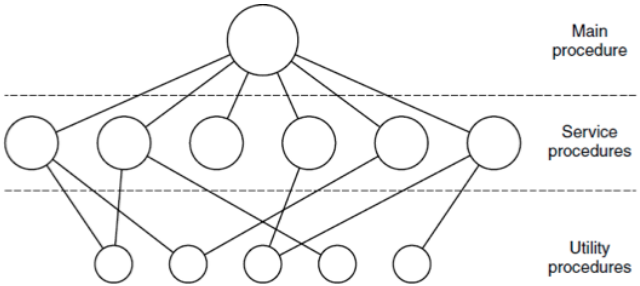
- Code in operating systems last a long time
- **Software Design** of an operating system is meaningful!

Languages?

- Early operating systems were written mostly in assembly code
  - IBM System OS/360
  - Millions of lines, complex, limited, and BUGGY!
- Modern Operating Systems are written in C (and a bit of C++)
  - May contain small portions of Assembly code
- The languages primarily provide *structure*

Monolithic Programs

- Run as a single process
- Collection of procedures, compiled into one executable, running entirely in kernel mode
- All procedures are free to call any others



- Advantage: Very efficient
- Disadvantage:
  - Changes require full re-compile
  - Crashes bring down *everything*
  - Very difficult to maintain

Layered Systems

Layered designs shares much with monolithic - however:

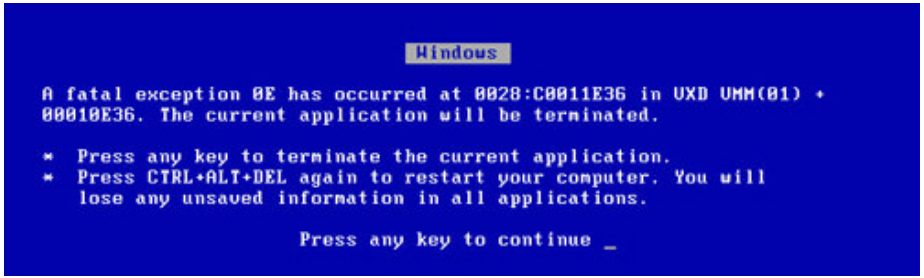
- Procedures in the bottom layers cannot call “up” to other layers
- This organization makes for easier maintenance, but requires a lot of discipline

About code...

- **All code has bugs** - yes, even operating systems
- Studies consistently show that *serious industrial system programs* have between 2-10 bugs per 1000 lines.
- Bigger programs = More Bugs!

What happens when you have a bug in an OS?

Nothing!



… or the process crashes…

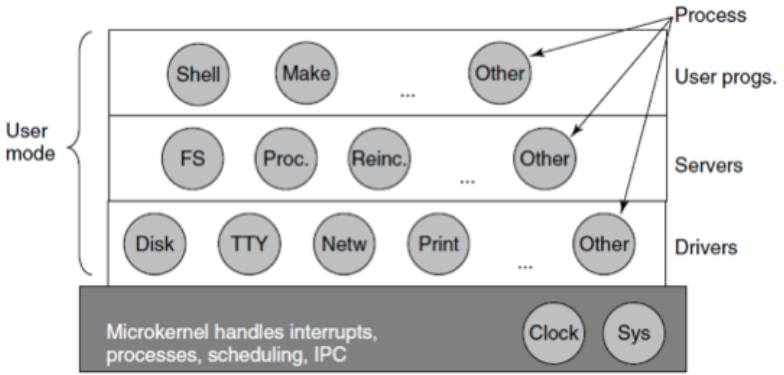
or a security vector is opened

Microkernels

Break the operating system up into **separate programs**.

- They each run at various security levels
- They each can start/stop without disrupting the system
- Pass standardized messages between each other instead of compiling against eachother

Microkernels



The system can be monitored, and crashed microkernels can be restarted without user intervention

Since communication is between processes (IPC), it’s slower than monolithic/layers

Next time

Please Read Chapter 1.8

We will discuss programming in C, and the general POSIX environment