SOFTENG370 Notes 2017

Theodore Oswandi

July 26, 2017

1 Lecture 1

1.1 Generics

Operating System The software that makes the computer usable. Using modern computers without an OS is "impossible"

Examples: Windows, OSX, Linux, Unix, iOS, Android, etc...

1.2 Approaches to Understanding

Minimalist

- mostly going to be using this one
- OS contains minimum amount of software to function
- archlike

Maximalist

- All software comes with standard OS release.
- Contains many utilities and programs.
- ubuntuish

1.3 Usable vs Efficient

- make sure you make OS suited for needs
- either specialised or more general purpose
- Think of who you expect to use the system
- If creating a realtime system with potentially thousands of operations in a short amount of time, have to consider efficiency
- Same with battery life if you expect the system to be used in a mobile setting.

1.4 OS themes

Manager Model

- OS is colleciton of managers, ensuring proper use of devices.
- Managers are independent.
- look out for everything associated with computer
- tie in with hardware. Current state of HW lets OS do more/less things

Onion Model

- Onions have layers (Abstractions)
- resources contained in lower layers.
- Lower layers can't access higher level layers but other way around possible
- Very difficult to get these layers 'right'
- can use in terms of security. Very good idea

Resource Allocator Model

- similar to manager model
- emphasis on fairness and providing services

Dustbin Model

- contains middleware that not considered part of OS
- Sees OS as bits no-one wants to do

Getting Work Done Model

- Idea of it is we use computers to do something else.
- Goal for OS is to help be able to get it all done.

1.5 OS design

1.5.1 Themes

All in one

- All OS components freely interact with each other
- MS-DOR and Early Linux

Separate Layers (Onion Model)

- Simplify verificiation and debugging
- Correct design difficult to get

Modules

- All in one with modules for some features
- Linux and Windows.

Microkernels

- Client/Server model
- make OS as small as possible
- **Exokernel** puts kernel outside. OS's job only need to authenticate people to use hardware.

VMs

- Java is an example of this

1.5.2 MS-DOS

- Written to provide the most functionality in the least amount of space
- not divided into modules
- Something exokernels trying to do. Make application program access hardware directly.

1.5.3 Early Unix

- UNIX OS in 2 parts. **Kernel** and **System Programs**
- Provides:
  - File System
  - CPU sheduling
  - Memory management
  - Other OS functions

- Ken Thompson and Dennis Ritchie
- Make OS as simple as possible.
- Simple 2 letter commands.
- Ideas of pipelining and process communication

1.5.4 THE Multiprogramming System

- THE was the first to use the layered system
- Contains 6 layers:

  5 User programs
  4 Input/Output buffering
  3 Operator-Console device driver
  2 Memory Management
  1 CPU scheduling
  0 Hardware

1.5.5 WinNT and Client/Server

- WinNT still being still run
  - Win10 now has Windows Subsystem for Linux
- NT provide env subsystem to run code written for differnt OS
- NT and successors are hybrid systems. Parts are layered but some merged to improve performance.