Introduction to Operating Systems
Vorlesung 19525 / SS 2001

Dr. Richard S. Hall
rickhall@inf.fu-berlin.de

Torsten Fink
tnfink@inf.fu-berlin.de

Important Times

- Vorlesung
  - Tuesday 12 - 14, SR 005 (Hall)
  - Tuesday 12 - 14, SR 005 (Hall)
- Übungen
  - Tuesday 10 - 12, SR 049 (Fink)
  - Thursday 14 - 16, SR 049 (Fink)
- Sprechstunden
  - Wednesday 10 - 12, 106 (Hall)
  - Wednesday 14 - 16, 107 (Fink)

Purpose of This Course

- Discuss the underlying concepts and principles of modern operating systems
  - Abstract concepts and approaches that are relevant to all operating system implementations
  - Specific concepts and approaches taken by specific operating system, such as Linux
- Give students the opportunity to experiment with low-level system programming concepts by providing exercises that involve programming in C and modifying the Linux kernel

Expectations

- Schein
  - Exercises (Übungen)
    - Will be collected and graded
      - One assignment may be dropped
    - Presentations of exercises will be necessary
    - Small project
  - Klausur (Final exam)
  - Exercises and Klausur have equal weight
- All students will be assigned an overall “Note” based on the standard FUB grading system

Übungen

- Read the Web site for news
- There is an exercise for this lecture
  - It is an introduction to C
  - It is due April 26, 2001
- For smoother start-up, Lab A should join Lab B for the introduction this week because there will only be a partial introduction to C for Lab A next week

Reading List

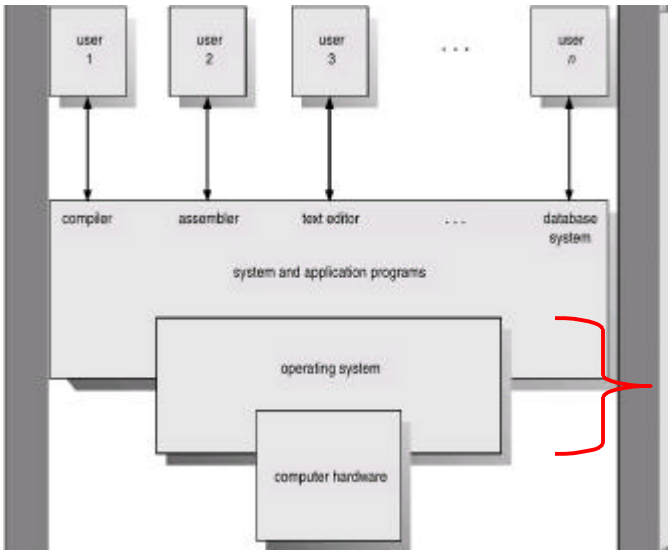
- **Applied Operating System Concepts (First Edition)**
  - Abraham Silberschatz, Peter Galvin, Greg Gagne, 2000.
- **Modern Operating Systems (Second Edition)**
  - Andrew Tanenbaum, 2001.

- Both books have slides available on the Web
  - The lecture slides will be based on the book slides, but will combine, edit, remove, and add slides as required

Why a Class on Operating Systems?

- Most likely, none of us will ever implement an operating system, so why study how they are implemented?
  - Operating systems contain examples of nearly all the issues you might ever encounter in programming
    - Concurrency, distribution, security, performance, efficiency
  - Understanding how the OS is implemented gives us better insight into how we should be solutions on top of it

Abstract Computer System Components



What is this?

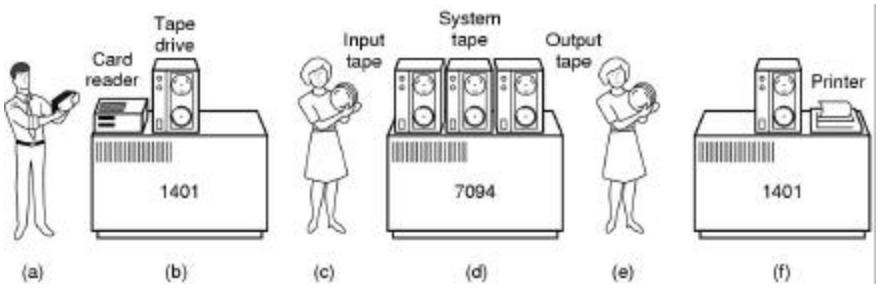
What is an Operating System?

- It is a program that is always running
  - Responsible for actually executing other programs
- It creates a virtual machine by
  - "virtualizing" the processor
  - "virtualizing" memory
  - "virtualizing" input/output
    - File systems, printers, etc.
- Why?
  - To provide higher levels of abstraction
  - To extend the functionality of the underlying hardware
  - To manages resources efficiently

The History of Operating Systems

- First generation 1945 - 1955
  - Relays, vacuum tubes, plug boards
- Second generation 1955 - 1965
  - Transistors, batch systems
- Third generation 1965 – 1980
  - ICs and multiprogramming
- Fourth generation 1980 – present
  - Personal computers

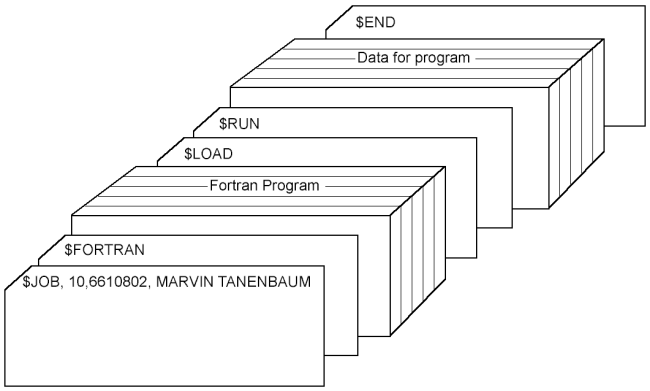
Second Generation Systems



Early batch system

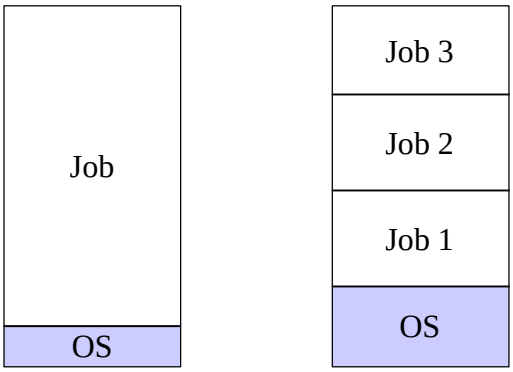
a) Bring cards to 1401
b) Read cards onto a tape
c) Put tape on 7094 which does computing (d)
e) Put tape on 1401 which prints output (f)

Second Generation Systems



Structure of a typical card file batch job for the Fortran Monitor System

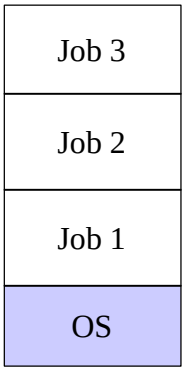
Third Generation Systems



- Second generation systems did not virtualize the hardware, a job was given the entire computer
- Third generation systems started to virtualize, allowing jobs to share the hardware

Third Generation Systems

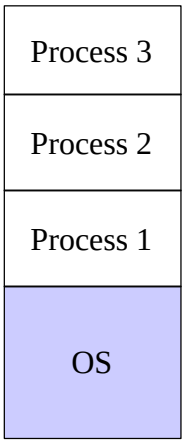
- Multiprogramming
  - All jobs are kept in a pool
  - OS chooses a subset of jobs to load into memory (job scheduling)
  - OS chooses a job to run and if that job has to wait (for i/o, etc.) then the OS chooses another job to run in the meantime (CPU scheduling)
  - OS/360 (IBM)



  - Allowing multiple jobs in memory at the same time requires special hardware support (MMU covered later)

Third Generation Systems

- Timesharing
  - OS loads multiple programs into memory (processes), processes that do not fit are swapped out to disk
  - OS executes one process for a short period of time, then quickly switches to a different process continuously
  - To the use it appears as if they have complete access to the computer
  - MULTICS (MIT, Bell Lab, GE 1965), UNIX (Bell Lab 1970)



Fourth Generation Systems

- Personal computers
  - Initially these systems lacked hardware support for advanced OS features; they were not multiuser or multitasking
  - PC operating systems have advanced as hardware cost went down and performance went up
  - CP/M, MS-DOS, Mac, Windows, Windows NT, Linux, etc.
  - We are still here…
  - Where are we going?
    - Network computers?
    - PDAs?
    - Embedded devices?

Current Domains of Operating Systems

- Mainframes (huge I/O capacity)
- Servers (shares resources over a network)
- Multiprocessors (variant of server OS)
- Personal computers (our computers)
- Real-time systems (monitoring sensors, etc.)
- Embedded systems (PDA, washing machines, etc.)
- Smart cards (severely constrained)