Eike Ritter
University of Birmingham

2-hour Lecture Course in Semester 1 2003/2004
Tuesday, 9am
Wednesday, 10.00am

Examination:
100 % written examination in May

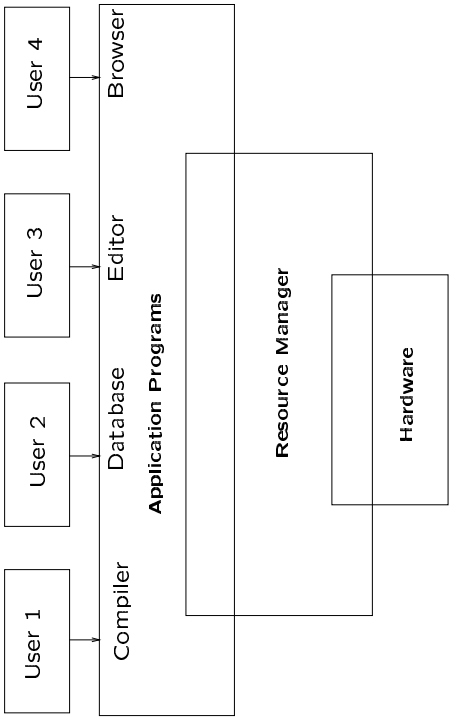
For advanced MSc students, examination will contain additional material

In addition, there will be exercise sheets every two weeks

A. Silberschatz, P.B. Galvin, G. Gagne. *Operating Systems Concepts*, Sixth edition, Addison Wesley, 2001.

A.S. Tanenbaum. Distributed Operating Systems. Prentice-Hall 1995.

A.S. Tanenbaum, *Modern Operating Systems*. Prentice-Hall, 1992.



Functions:

1.) Implement multi-user programming:

- several processes ready to be executed; OS chooses next one
- Have to simulate parallelism on a sequential machine
  ⇒ Must avoid Starvation, Deadlock and achieve fairness.
- Protection of processes from each other
  ⇒ Separation of logical and physical address spaces.

2.) Memory Management

Main memory is fast but expensive
Disc storage is slow but cheap

⇒ Only part of memory needed for process execution in main memory
⇒ OS manages memory allocation

View from process: One large address space ("virtual memory").

3) Input/Output

OS manages highly complex interaction with I/O-devices:

real-time constraints have to be observed

4.) Distributed Computing

Data or programs can be on different computer
Aim: Transparency (same interface for local and remote access)
⇒ need to support protocols for file transfer, remote login as part of the OS

Hardware prerequisites

OS relies on hardware to ensure protection of processes from each other.

Need at least two different execution modi for hardware:

1. Kernel or Supervisor mode: Allows unrestricted access to all resources

2. User Mode: Only instructions not affecting other users are allowed; sanity checks are enforced.

Instructions allowed only in Supervisor mode are called privileged instructions.

Examples of Operating Systems

- **Unix**: Started 1969 as Multi-user Time-sharing System
  Vital for development of the Internet in the 1980's
- **Linux**: Unix derivative, started in 1991 when PC-HW was powerful enough to run Unix
- **MS-DOS/Windows 95/98/ME**: Started in 1980's as OS for PC's
  Restricted Power of 1980's PC's meant very limited functionality
- **Windows NT/2000/XP**: RE-implementation started in the 1980's with extended functionality
- **Mac OS** OS with limited functionality for Apple Mac
- **Mac OS X** Unix-like re-implementation for Apple Mac

OS for embedded systems

Have variety of embedded systems
from special-purpose controllers to programmable chips
$\Rightarrow$ wide variety of OS's to satisfy very different needs systems
Due to limited resources of embedded systems re-emergence of OS-issues of 1970's and 1980's

Interrupts

OS requires feedback from hardware when operations are finished.

Standard mechanism: Interrupt:

- Hardware generates signal, which is transferred to processor
- Processor interrupts current activity
- Processor executes appropriate interrupt service routine
- Processor resumes previous activity

Short response time important, so interrupt service routines tend to be small

Interface to application programs

OS accessible only via specified procedures (system calls)

Execution of a system call:

- OS reads call parameters and checks appropriate privileges
- OS executes requested function in Supervisor mode
- OS returns result