

WEEK 1

582497 Operating Systems, 8 op

Tiina Niklander

University of Helsinki
Department of Computer Science

Course

- Structure
 - 12 weeks
 - 2 exams
- Prerequisites:
 - Bachelor level studies, at least our courses
 - Computer Organisation I
 - Concurrent Programming

<http://www.cs.helsinki.fi/u/niklande/opetus/kj/>

Goal

- To understand the fundamental principles and methods of Operating System both from the user's and implementor's view points
- You need to be able to explain the general principles and some alternative implementation algorithms to them
- Examples Operating Systems:
 - Unix = Linux, Windows XP ja Solaris
- After the course you are expected to be able to
 - Describe the main parts of the OS, their functionality, dependences and their interactions.
 - Study how one specific OS implements certain feature (if necessary by reading the source code)

Course book (must have!)

- [Stal05] William Stallings, *Operating Systems, 5th. ed.*, Prentice-Hall, 2005
- Alternative books:
 - [SGG07] A. Silberschatz, P. Galvin ja G. Gagne: *Operating System Concepts with Java*, seventh edition. Wiley, 2007
 - [Tane01] A.S. Tanenbaum, *Modern Operating Systems, 2nd. ed.*, Prentice-Hall, 2001.
 - [DDC04] H.M. Deitel, P.J. Deitel, D.R. Choffnes, *Operating Systems, 3rd ed.*, Prentice-Hall 2004
 - *Any (good quality) book with the title: Operating Systems*

Course structure

- Part 1: OS structure, Processes, threads
 - Myös käyttöjärjestelmän rakenne ja tehtävät
- Part 2: Memory Management, virtual memory, scheduling
- Part 3: I/O and file management
- Part 4: Distribution and security

How to do the course

Alternative 1: participate to the course

- Enroll to the small group session in ILMO
- Read the material (and/or go to lectures)
- Prepare for the sessions by solving the problems in advance
- Group projects
- Exams
- Weekly sessions, projects and exams give points for passing the course. They are used in grading also

How to do the course

Alternative 2: Separate exam

- Enroll to the exam in ILMO
- Inform the teacher that you need to have English questions (at least a week before)
- Exam based on the preceding course (material information from that course)
- *Questions quite often similar to the weekly exercises*

Group projects (group tasks)

- Three projects during the course (1-4 points each)
- Optimum group size 2 or 3 students
- Tasks are mainly
 - Solving a larger problem and writing a report
 - Explaining the functionalities of certain features in the report (how something works or is implemented)

WEEK 1

Read for next week:

Chapters 1 & 2

(+ solve the problems presented in the web-page)

Hardware

CPU, Execution cycle

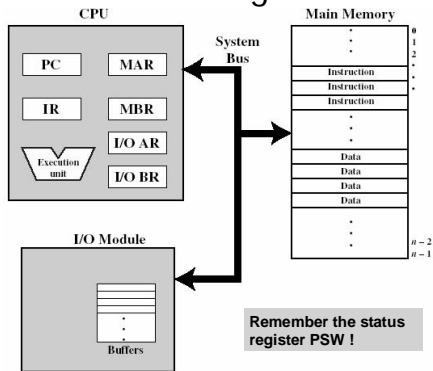
Interrupts and exceptions

I/O, Memory

Stallings, Chapter 1

(known from Computer Organisation I)

Central Processing Unit Fig 1.1 [Sta105]



Memory Management Unit - MMU

- Transforms the program's internal address to physical memory address

Registers for address and content

- MAR, Memory Address Register
- MBR, Memory Buffer Register

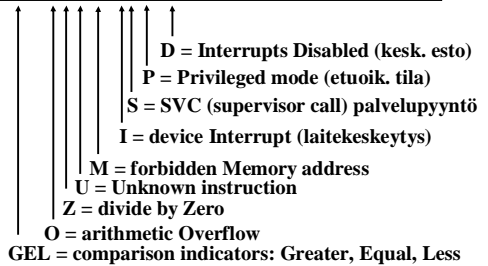
MMU

- No virtual memory:
 - Base Register: physical location start here
 - Limit Register: last allowed physical address or the size of the address space of the process
- Virtual memory (example):
 - PTR, Page Table Register
 - Physical address location of the start of the page table
 - TLB, Translation Lookaside Buffer
 - Cache for the page table entries (most recently used)

PSW – Program status word

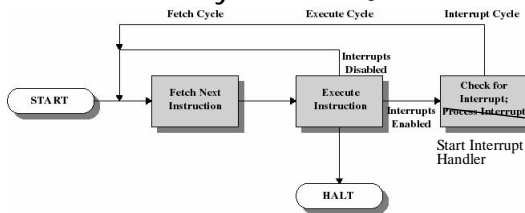
32 bits (each has a value 0 or 1)

GEL OZUM IS P D . . .



Instruction cycle

Fig 1.7



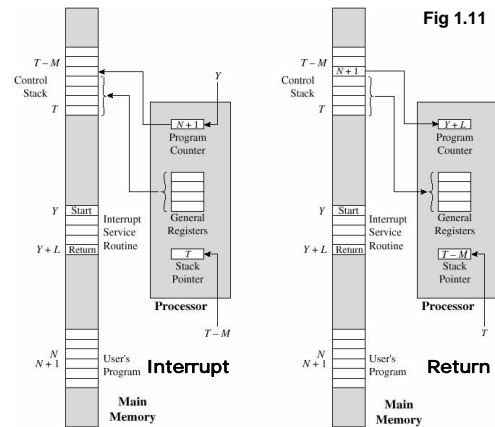
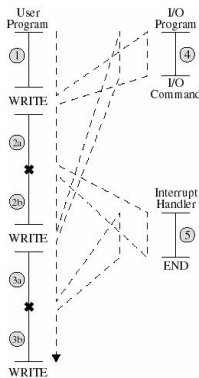
Interrupts

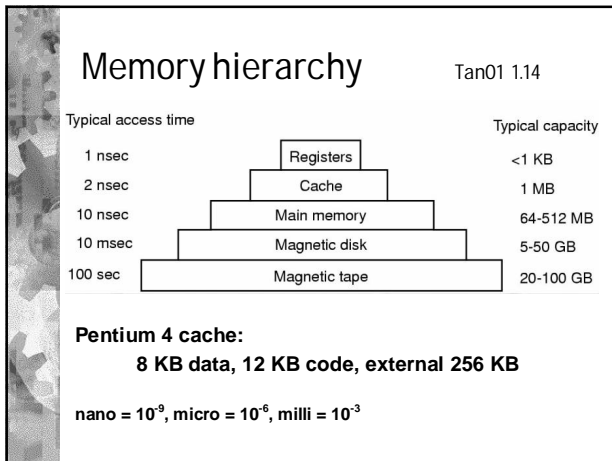
Table 1.1 Classes of Interrupts

Program	Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, and reference outside a user's allowed memory space.
- error	
- System call	
Timer	Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.
I/O	Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions.
Hardware failure	Generated by a failure, such as power failure or memory parity error.

I/O & Interrupts

Fig 1.5b [Stal05]





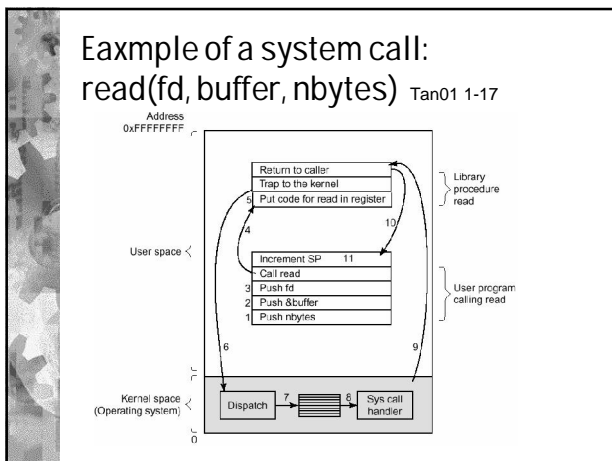
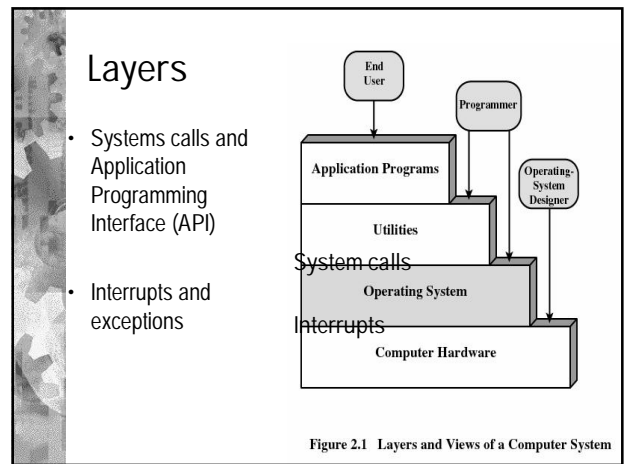
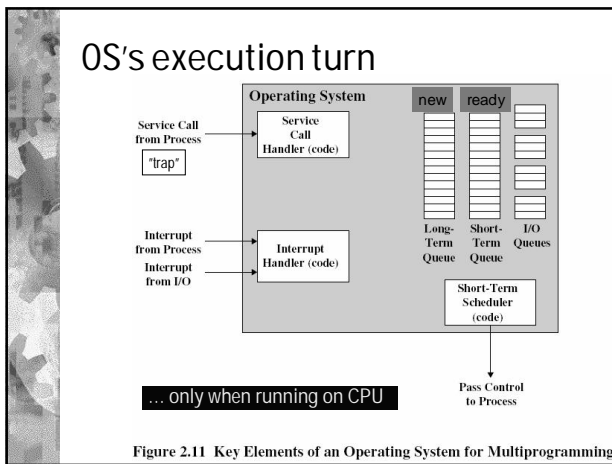
Locality of reference

Spatial and temporal locality:

- For example in a loop a small set of instructions is executed several times
- Certain part of code only uses a small set of variables (data)

⇒ When program makes a memory reference (data or instructions), it is likely to refer again to the same location or a location near by

- This is the principle behind the usage of caches



Example Tan01 1-19

```

#define TRUE 1
while (TRUE) {
    type_prompt( ); /* repeat forever */
    read_command(command, parameters); /* display prompt on screen */
    /* read input from terminal */

    if (fork() != 0) { /* fork off child process */
        /* Parent code. */
        waitpid(-1, &status, 0); /* wait for child to exit */
    } else {
        /* Child code. */
        execve(command, parameters, 0); /* execute command */
    }
}
    
```

Simple shell

WEEK 1

OS structure

Chapter 2

Computing System

- OS offers:
 - Process management
 - Memory management
 - I/O management
 - File management

Fig 1.1 [SGG07] –
Silberschatz, Galvin and Gagne: Operating Systems Concepts with Java, Wiley, 2007

OS structure

Services

- User interface
- Utility programs
- Program execution
- Memory management
- I/O
- File systems
- Protection
- Accounting (statistics)

Services

- Failure handling/ management
 - Hardware faults and failures
 - Software faults and failures
 - Missing resources
- Support for recovery
 - Status information (to the process)
 - Retries
 - Killing a process

Hierarchical structure: OS device-independent services

Level	Name	Objects	Example Operations
13	Shell	User programming environment	Statements in shell language
12	User processes	User processes	Quit, kill, suspend, resume
11	Directories	Directories	Create, destroy, attach, detach, search, list
10	Devices	External devices, such as printer, displays and keyboards	Create, destroy, open, close, read, write
9	File system	Files	Create, destroy, open, close, read, write
8	Communications	Pipes	Create, destroy, open, close, read, write

Hierarchical structure: OS device-dependent services

Level	Name	Objects	Example Operations
7	Virtual Memory	Segments, pages	Read, write, fetch
6	Local secondary store	Blocks of data, device channels	Read, write, allocate, free
5	Primitive processes	Primitive process, semaphores, ready list	Suspend, resume, wait, signal P / V

Memory management: must consider the structure of MMU
 Device controllers: must consider the structure of the devices
 Scheduler: synchronization primitives, registers copy/restore

Hierarchical structure: Device layers Brown, Denning 1984

Level	Name	Objects	Example Operations
4	Interrupts	Interrupt-handling programs	Invoke, mask, unmask, retry
3	Procedures	Procedures, call stack	Mark stack, call, return
2	Instruction Set	Evaluation stack, micro-program interpreter, scalar and array data	Load, store, add, subtract, branch
1	Electronic circuits	Registers, gates, buses, etc.	Clear, transfer, activate, complement

What hardware features are needed to support OS functionalities?

Difficult parts in OS implementation

- Synchronisation / timing
 - Occasionally must wait until something else has happened
 - Priorities of the interrupts
 - Signals from devices and messages from program must not be lost or duplicated
- Mutual exclusion
 - Some resources can be allocated only for one user at a time
 - Shared data, shared file, printer
- Deadlock, Starvation
 - Difficult to detect
 - Occasionally must wait for others (which cannot happen)
 - Too low priority, no access to services (everybody goes before)

Concurrency
Ch 5, 6.1-6 [Stal05]

Part of the 'Concurrent Programming' course. Not covered in this course.

Short history of OS

Old (basic) models:

- Batch System
- Multiprogramming, multitasking
- Time-Sharing

More complex models:

- Multiprocessor
- Networked systems
- Distributed system
- Client-Server

ALL OFFER SIMILAR FUNDAMENTAL SERVICES

Hardware support ⁽¹⁾

- Memory protection (MMU)
- Interrupt mechanism
 - Detection and initiation by hardware, continue in sw
- Clock interrupts
 - Needed to allow OS to run on certain periods (otherwise the program running on CPU could continue forever)

ALSO (often provided features)

- Privileged instructions
- Supervisor and user modes

Multiprogramming

- More OS features: scheduling, memory allocation

