Operating systems

ELTE IK.

Dr. Illés Zoltán

zoltan.illes@elte.hu

Financed from the financial support ELTE won from the Higher Education Restructuring Fund of the Hungarian Government

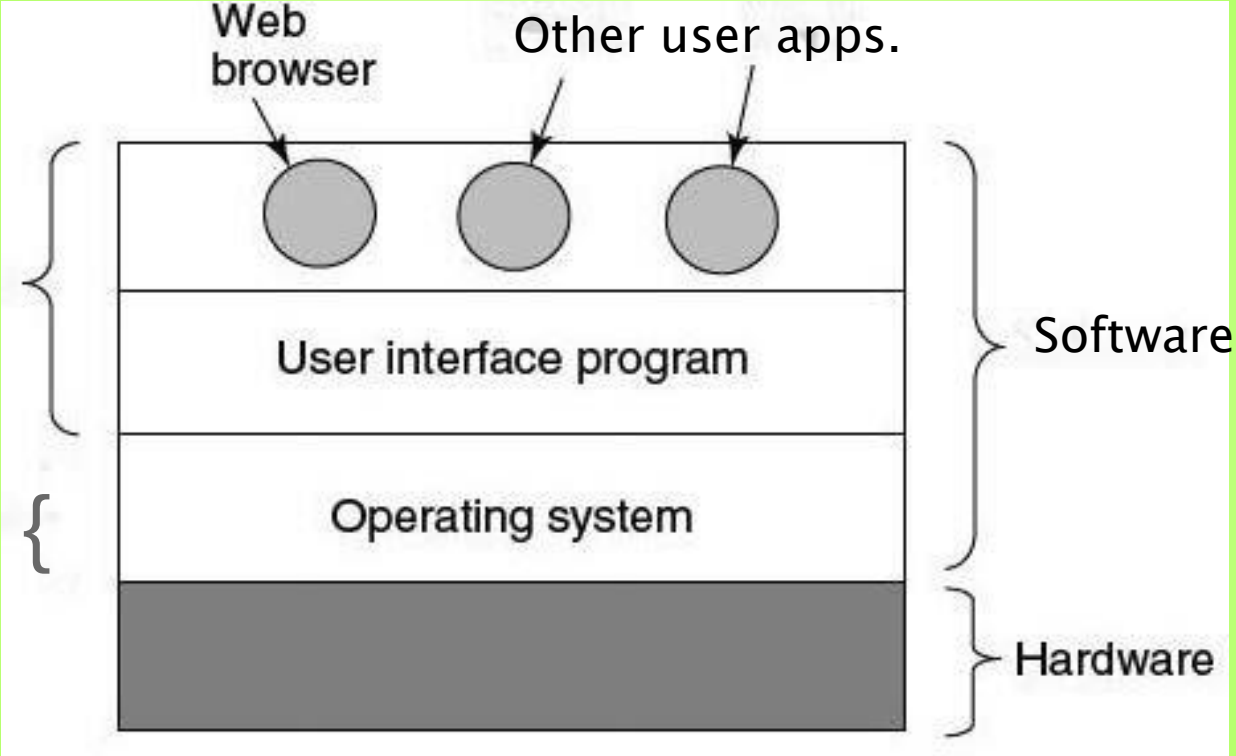
Introduction

- Review (Introduction to Comp. Technology)
- Computer Architecture I.(HW)
- Computer Architecture II. (SW)
- Definitions of the „Operating systems”
- History, evolution of Operating systems
  - Past, Present, Future?
- Notions, names, conventions in an operating system
- System calls
- Structures of operating systems

An overview of a computer

User mode

Kernel mode



Review

- As we finished „Introduction Computer technology” …
- Script programs
  - Best friend of the system admin istrator
    - Shell script
    - PowerShell
- Client-server architecture
  - HW differences
- Client-server services
  - Administration
  - SW differences

Computer Architecture (HW)

- **Computer components**
  - **Hardware side**
    - Stored programs, commands, data are stored (binary, why?) in the memory, same way.
    - Central processing Unit (CPU), arithmetical-logical unit (ALU) are controls the commands execution and base arithmetical instructions.
    - Input/Output handling, this is the communication between CPU and outside world.
    - Neumann principals.
  - **Base parts: Processor, Memories, Pheripherals, data storage**
    - Connecting element: Bus (data, address, control)

Processor commands

- All parts are intelligent, no direct communication, main role: processor
- Registers: special memories inside of the processor
  - Register groups (general,segment, status,…)
- Groups of proc. commands
  - Data movement commands (registers-memories)
  - Jump commands (absolute-relativ)
  - I/O port read/write,
  - Interrupt handling etc.

Processor execution levels

- Intel 80286 – one level (real mode, 20 address lines– protected mode – not often used)
- Intel 80386 – starts in real mode– switch to protected mode
  - In protected mode there are 4 level (level 0, level 1, level 2, level 3)
  - Level 0 – kernel mode
  - Level 1, 2– not used
  - Level 3 – user mode
- Level 0 functionality
  - Interrupt handling (IDT)
  - I/O port management
  - Memory management
- Sofware interrupt handling (trap) is same as a hardware interrupt handling
- Interrupt masking – NMI

Using processor commands

- Commands, data are in the memory, the CPU executes the commands
  - Mov al, ‘F’
  - Mov ah,’T’
  - Mov bl,’C’
  - etc.
- Is it fun?
  - Sure--- (FTC:)… //FTC is a football team ☺
  - …

Computer Architecture (SW)

- **Execution levels, sw levels**
  - **Logical circuits- no sw**
  - **CPU, microprogram, microarchitecture level**
  - **Assembly,hw devices**
  - **Operating system**
  - **System application (on user level too)**
    - Written in low level, in assembly
    - Written in high level language (usually in c,c++)
  - **User applications**
    - Like awk, Pasians etc.

Operating system definitions

- Operating system: Such a program (group) which gives to the user an easy to use workplace, hiding the computer(system) elements.
- Op. system as a virtual machine
  - Do not care how to copy, how to play a movie (film), only i need player or a „copy machine”.
- Op. System as a resource manager
  - Printing spool manager
  - Memory manager
- Kernel mode
- User mode
- Special controlled mode-embedded systems

Operating system main task

- Giving an easy to use, efficient user interface!
  - 0. generation: spec. Hw. Switching table
  - Early systems (1940-70): Special terminals
    - The architecture was the same as today..
  - Beginning of 80th : microcomputers (ZX81 etc), Basic
  - PDP compatible TPA1140, serial terminals
  - MS DOS character terminal
  - Unix_X Window system, Xerox, MacOS
  - Windows 3.1, 95,98,Mill,2000,XP, Win7
- Are they good user interfaces?

Peripheral communication

- Continuous checking (polling)
  - I/O port continuous reading
  - Often used technic, typically used in synchronic software calls.
- Interrupt usage
  - There is no continuous reading, the program waits for an event (external, data is ready), the handler will read the data.
  - Asynchronic method
- DMA, direct memory access
  - eg. direct memory addressing: 0xb800:0

Application libraries

- **Computer command levels:**
  - **Machine code**
    - E.g.:intel x86, mov ax, ‘F’, mov eax, ‘T’, jmp address
  - **Normal user API, Application Programming Interface**
    - C64 ROM Basic
    - DOS (IBM, MS) , IO.sys, msdos.sys, interrupt table
    - Windows 98,…Windows 7, Win32 API
    - Unix-Linux system libraries, C language
  - **Script programming (BASH, PowerShell)**
    - We saw it in 1th semester! (Fundaments of Computers)

User API's

- Layered architecture
- Divided into 2 groups:
  - Kernel level calls
    - Pheripheral communication
  - User level calls
    - Wide library support
- Which programming language is supported?
- Sure, the C language! And more? C++- etc…☺
- Compatibility!

What is POSIX?

- POSIX = Portable Operating System Interface for uniX
- Official standard: IEEE 1003 – ISO 9945
- The POSIX is a minimal , standard API
- POSIX 1, 1a, 1b,1c …versions
- ANSI C compatibility
- Every OS has POSIX compatibility
- Windows compatibility, until Windows 8
  - Windows Services for Unix

Main POSIX API features

- File, directory management
- Process control
- Signals
- Pipes
- Standard C library
- Clocks,Timers
- Semaphors
- Synchron, asynchron I/O
- Threads
- etc.

Function groups

- Math functions: e.g. sin, cos, tan,atan, atan2, log, exp etc.
- File management functions: e.g. creat, open,fopen, close, read, write, unlink etc.
- Directory management functions: e.g. opendir, closedir, mkdir, rmdir, readdir etc.
- Character, string management: strcpy,strlen strcmp,strcat,strchar,strstr etc.
- Memory management: malloc, free, memcpy etc.
- Communication functions: msgsnd, msgrcv,shmat, semop, signal, kill, pipe etc.

How can we use in practice?

- Our Op. system: Suse Linux Enterprise server
  - oprendszerek.inf.elte.hu
- Text editor: vi, mcedit
  - Or using locally a graphics editor, and ftp.
- Help: man
  - E.g.: man exit, man strlen
- Compile: cc -c first first.c
  - Try to resolve warnings too!

Operating systems API's

- So much API's as Op. systems
- Typical API's
  - Open VMS
  - OS/400
  - System V, BSD , common part: POSIX
  - Win32 API
  - Mac OS API
  - Windows Mobile, CE API
  - Palm OS
  - Nokia S40, S60, S80 API
  - Android x, WP7,8, etc.

Firmware – Middleware

- We saw: Hardware – Software
- Hardware is not only the physical device
  - Eg: HDD firmware.
  - BIOS calls.
- Firmware: An intagrated software is build into hw by manufacturer
- Middleware: A system layer is above Op. System.
  - E.g.: JVM

Operating systems generations I.

- Historical generation: Charles Babbage (1792-1871)
  - Fully mechanics machine, no op.system
    - Operator job
  - Later, one of programmer was Ada Lovelace (Lord Byron's (poet) doughter) (Ada language)
- First generation, 1940-1955, switching table, rele, electrical vacuumtube
  - Neumann János, Institute for Advanced Studies, Princeton
  - Customised comp. machines
  - Machine code, simple math calculations
  - Punched cards as a data, command holder

Operating systems generations II.

- Second generation 1955-1965, transistors systems
  - Trusty, better computer components
  - Mainframes, computer centers
  - Distinct planning, manufacturing, programming, operation
  - Punched cards, tapes systems, batch systems
  - Fortran language
  - Op. system
    - FMS, Fortran monitor system
    - IBM 7094 machine, 1401 input – 7094 evaluation-1401 output unit

Operating systems generations III.

- Third generation, 1965-1980, appears integrated circuits
  - IBM 1401 and 7094 next one: System/360 family
  - Same architecture, construction, compatibility
  - Developing OS/360 , Fits to all System/360 HW, it results a big, tricky, complicated op. system.
  - Multiprogramming, multitask
    - There are more tasks in the memory at the same time.
  - Spooling, time sharing systems
  - No general on-line access, work

Operating systems generations III.

- First time sharing system: M.I.T-en CTSS (CompatibleTime Sharing System)
- MULTICS, Multiplexed Information and Computing System
  - AT&T Bell labs, General Electric support
  - PL/1 language
- Bell Labs, Ken Thompson, Simplifying Multics , PDP 7->UNIX
- Two main developing stream:
  - Berkeley University – Berkeley Software Distribution
  - AT&T Bell Labs, System V Unix

Operating systems generations IV.

- Since1980 till now, personal computers, MS Windows
- LSI (large scale integration) circuits, CPU developement
- Z80- CP/M (Control Program for Microcomputers)
  - ZX-81, ZX-Spectrum- Basic
- Intel x86 family, IBM PC- DOS, MS DOS
  - Command line desktop
- GUI- X Window, Mac OS X, MS Windows
- Network, distributed systems

MINIX 3

- From the beginning the UNIX source was free, open by AT&T .
- UNIX – not open, free from AT&T 7. version
- MINIX – MINI Unix, open source code
  - A.Tanenbaum, Vrije Univ. Amszterdam
  - Written in C language
- Linus Torvalds, ‚Tanenbaum’s student’
  - MINIX modification, 1994, LINUs uniX->LINUX
  - Open source
  - LAMP-Linux-Apache-Mysql-Php

System calls

- System call: An operating system library call which connects the user level call to the kernel level.
- Two category:
  - Task or process handling call
  - File management call
- The best friend of the programmer: man, …

Process management

- **Process- a program loaded into the memory and executing it**
  - **Own address space**
  - **Process table**
    - Address, register, workfile datas, metadatas
  - **Process start, stop**
    - Shell, child processes
  - **Process pause**
    - Memory map + process table save
  - **Process communication**
    - Signals

File management

- Only one directory, /
  - Tree structure
  - Two registry entry: file, directory
- Dir. operations: create, copy, delete, open, read, write
- Access rights: rwx, - denied the given right
  - SETUID, SETGID, Sticky bit
- File system connecting (mount), disconnecting (unmount)
- Specific files:
  - Character, block files, /dev directory
- Special file: pipe

Major process handling calls

fork – To create a new process
exec – To execute a new program in a process
wait – To wait until a created process completes its execution
exit – To exit from a process execution
getpid – To get a process identifier of the current process
getppid – To get parent process identifier
nice – To bias the existing priority of a process
brk – To increase/decrease the data segment size of a process

Major signal handling functions

sigaction - Define action to take on signals
sigreturn - Return from a signal
sigprocmask - Examine or change the signal mask
sigpending - Get the set of blocked signals
sigsuspend - Replace the signal mask and suspend the process
kill - Send a signal to a process
alarm - Set the alarm clock
pause - Suspend the caller until the next signal

Major file management functions

open – Create a file or open an existing file
close – Close a file
read – Read data from a file
write – Write data to a file
lseek – Move the file pointer
stat , fstat – Get various file attributes
fcntl – File locking

Major directory management functions

mkdir – Create a new directory

rmdir – Remove an empty directory

link – Create a link to a file

unlink – Delete the link

mount, umount – Mount/ unmount a filesystem

chdir – Change the current working directory

Operating system structures

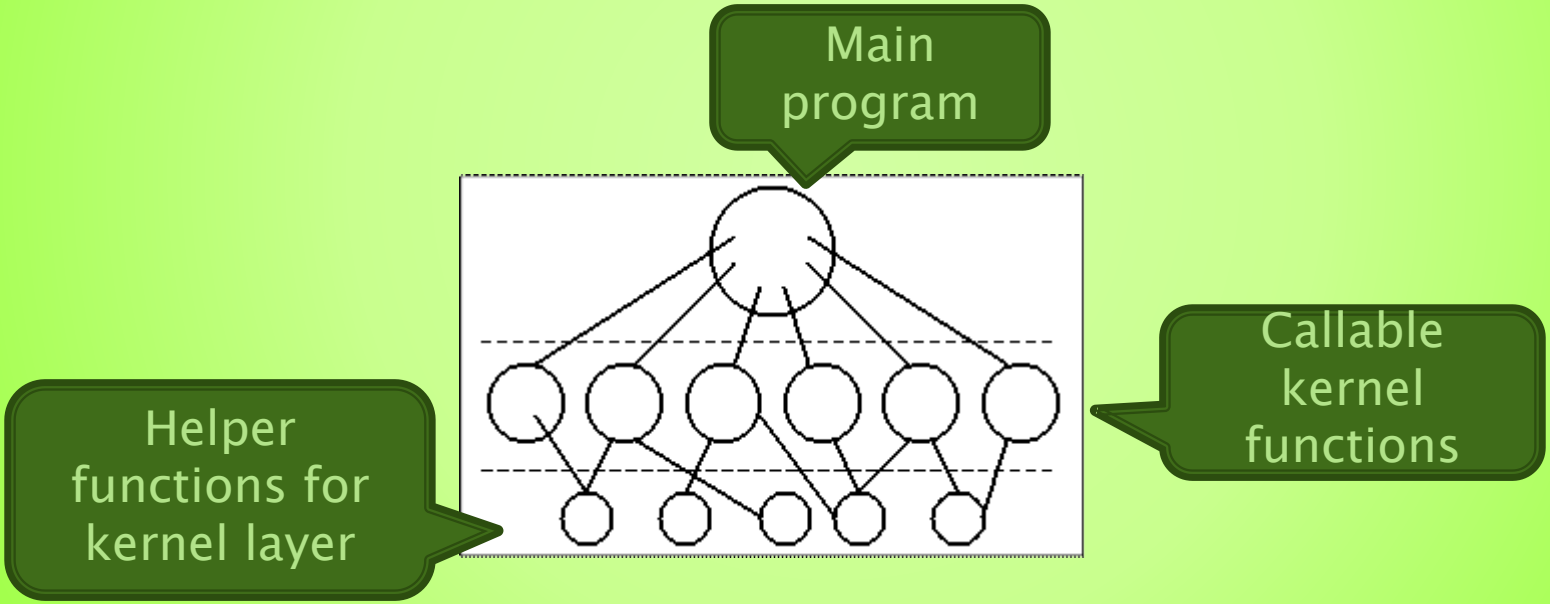
- Monolitics systems
- Layered systems
- Virtual machines
  - Exokernel
- Client – Server model

Monolitic systems

- Generaly: no defined structure, but…
- System library is one big system, so every process can use everything.
  - No information hide.
- Exist software moduls, planning modul groups
  - The predefined entry points(functions) can be called!
- During a system call the execution level is often switched to kernel mode (CPU execution level 0)
  - Parameters in registers
  - Trap

Monolitic system model

- **Monolitic system: tipically a 2 layers support**



Layered architecture

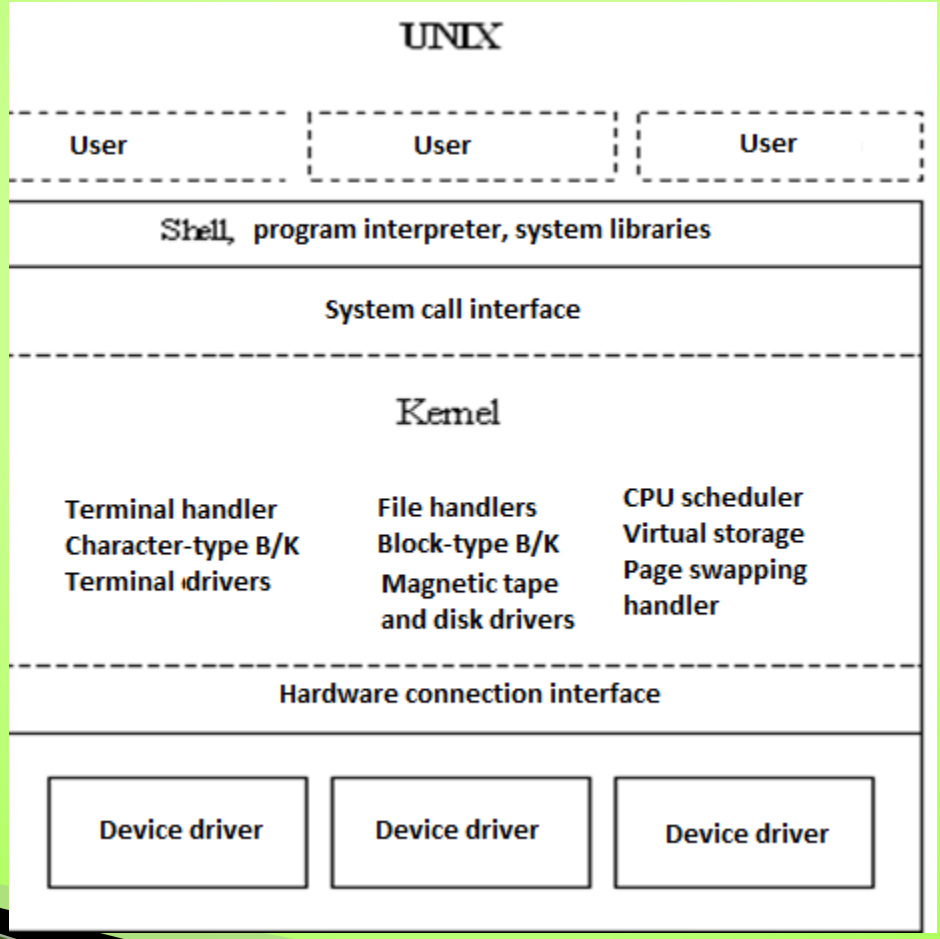
- **Planned by E.W. Dijkstra : THE (1968)**

5.	The operator
4.	User applications
3	I/O management
2	Machine-process
1	Memory management
0	CPU management and multiprogramming

- **In MULTICS more generally**
  - **Round, ring architecture**

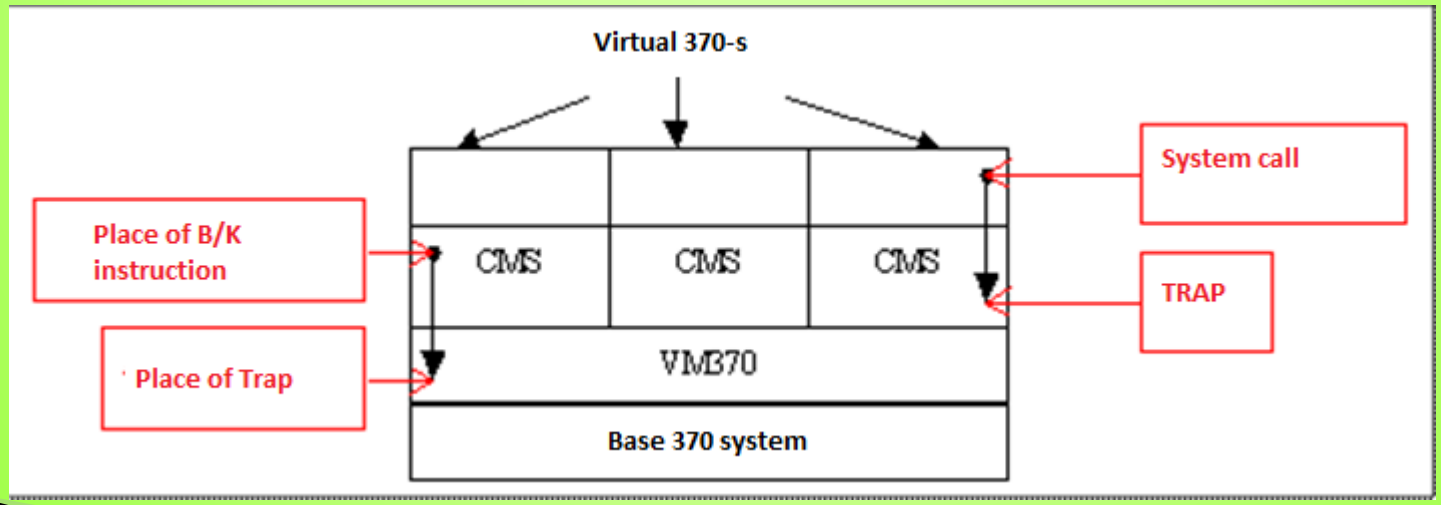
Typical layered structure

- **The UNIX typical layered (round) structure.**



Virtual machines

- The idea comes from IBM
- It was implemented on VM/370 system
- Virtual machine Monitor: gives a „copy” of the hardware

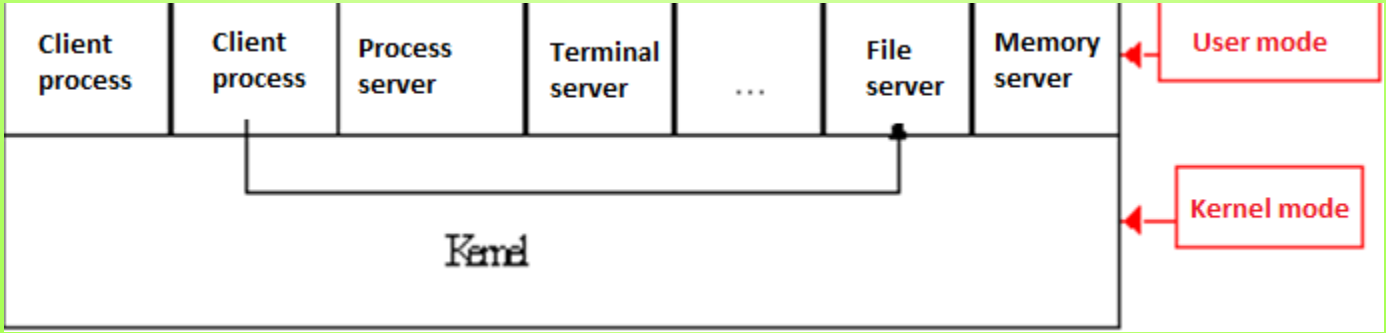


Virtual machines today

- VMWare – under Unix- Linux platform
  - Under Windows too
- MS Virtual Server, Virtual PC
  - Todays processors support virtualization
- Microsoft- Hyper-V
- LINUX – XEN-KVM
- Other virtualization:
  - JVM
  - .NET

Client-Server model

- Improvement of the idea of vm/370
  - To divide the task to much more parts.
- User application: client program
- Service application: server program
- Both runs in user mode
- Less and less functions stays in the kernel



Operating system expectations I.

- Efficiency, an efficien management of the real resources
- Reliability, Build a reliable system
  - Working all time without problems
  - Archive datas
  - 3-4 nine… reliability (99.9%, 99.99%)
  - Failover systems
    - Redundant systems(both HW and SW too), Server Cluster

Operating system expectations II.

- Security
  - Safe work from outside „attacks”, firewall
  - Data security
- Compatibility
  - Data, program exchange between systems.
  - Role of standards (POSIX)
- Low energy occupation
  - Not only for mobile devices

Operating system expectations III.

- Flexibility
  - Flexibil resource management(memory, processor, today cloud technology)
- Manageability
  - On support and user level too.
- Can we offer all expectations?
  - All manufacturers answer: yes….☺
- We'll see later!

Thanks for your attention!

zoltan.illes@elte.hu