Technische Hochschule Rosenheim

Modul - IT Systems (IT)

Bachelor Programme AAI

04 - Lecture: Operating Systems

Prof. Dr. Marcel Tilly

Faculty of Computer Science, Cloud Computing

Vers. 8.11.2021

Agenda

Technische Hochschule Rosenheim

We will look at the following in this lecture:

1. Introduction (basic concepts)
2. Architectural concepts
3. Activities, Processes
4. Threads
5. Scheduling

Learning objectives

Technische Hochschule Rosenheim

Students learn ...

- ... how to use basic tools
- ... automation of tasks
- ... interaction between applications and operating system(s)
- ... processes within operating systems
- ... Unix-like and Windows operating systems

Operating system

Technische Hochschule Rosenheim

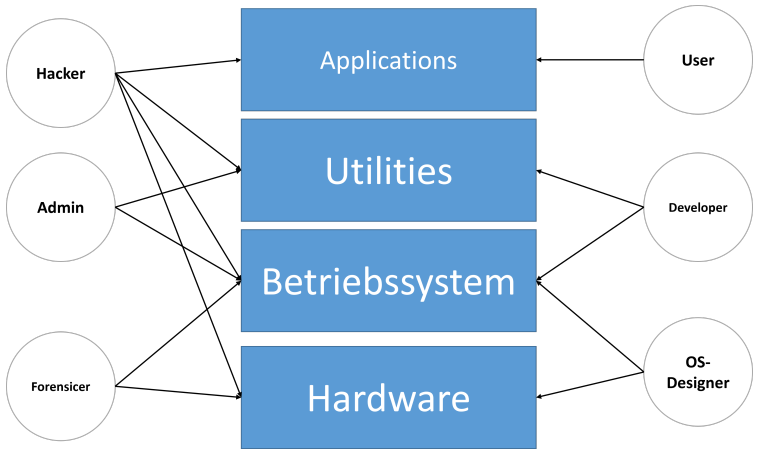
Why do we need an operating system?

- Providing services and the necessary abstractions
  - Process
  - File(s)
  - Device drivers, etc.
- Resource management including logging
- Coordination of parallel processes
- Basis for protection and security

Perspectives on operating systems

There are different roles in the use of a system (OS):

- User
- Administrator
- System programmer
- Driver programmer
- (Hostile) attacker, "hacker
- Forensic scientist
- OS Designer



What is an OS?

Technische Hochschule Rosenheim

Definition

An operating system (OS) is the link between the hardware of a computer on the one hand and the user on the other. It comprises programs which, together with the properties of the computer, "form the basis of the possible modes of operation of this system and, in particular, control and monitor the execution of programs".

Requirements

- "optimal" utilisation of the available operating resources
- Fulfilment of user requirements, e.g. control of priorities

What does an OS do?

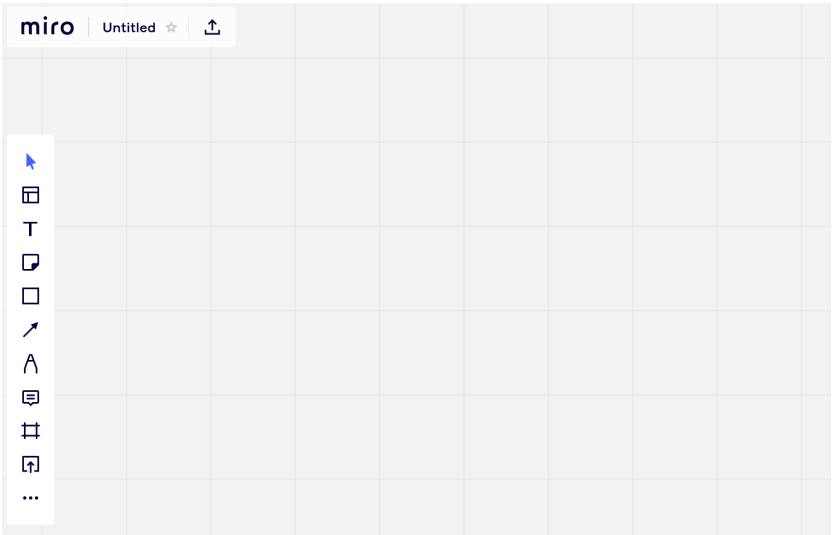
Technische Hochschule Rosenheim

- Boots up the computer system
- Adapting the performance of the hardware to the needs of the users (abstraction of the hardware).
  - The operating system extends the hardware functionality, e.g. through file management
- Organisation, management and control of the entire operational sequence in the system
- Management and, if necessary, appropriate allocation of operating means (resources) to different execution units
- Control and enforcement of protective measures (e.g. access rights), in particular to ensure the integrity of data and programmes, especially in multi-user operations.
- Logging of the entire sequence of events in the system, collection and analysis of performance data for system optimisation

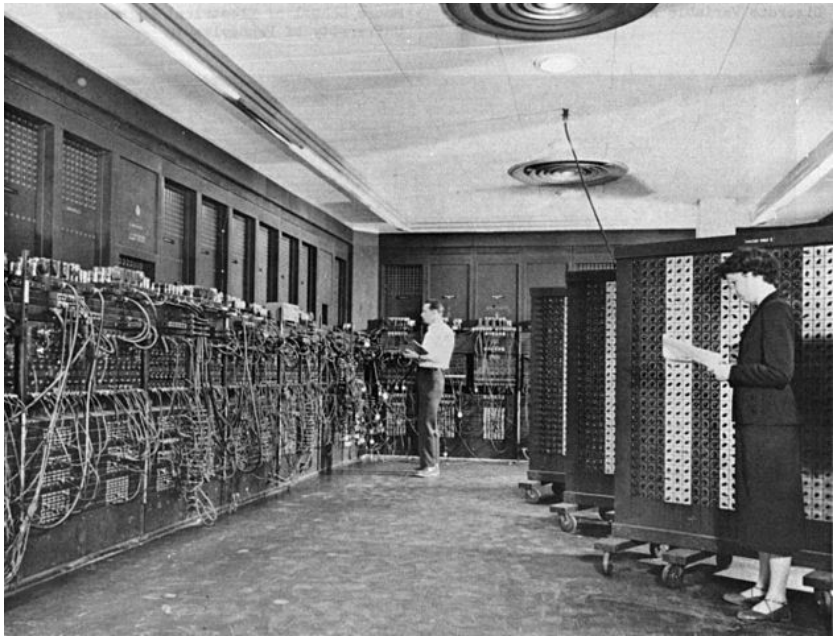
Which system programmes do you know?

Small survey!

Miro board: https://miro.com/app/board/o9J_lm7fCk8=/



Once upon the time...



History of operating systems

Technische Hochschule Rosenheim

- First computers [ca. 1945]
  - no OS
- Mainframes [ca. 1960 . . . today]
  - IBM System/360: OS/360 family
  - currently z/OS for today's IBM mainframes
- Minicomputer [ca. 1970 . . . 1985]
  - DEC PDP 11: UNIX [1970 . . . today]
  - DEC VAX 11/780: VMS [1977 . . . today]
- Workstations [ca. 1982 . . . 1995]
  - HP, Sun, Silicon Graphics etc. m
  - UNIX, i.e. , HP-UX, SunOS, IRIX, . . .
  - largely disappeared today (replaced by high-end PCs)

Personal computer

Technische Hochschule Rosenheim

- Home computer [ca. 1978 . . . 1994]
  - Commodore PET, C64, Atari ST, Commodore Amiga, . . .
  - Homebrew-OS ("ROM-BIOS"), CP/M, AmigaOS, . . .
  - largely extinct today
- Apple computers [from 1984]
  - macOS - first commercial OS with graphical user interface
  - OS X (basis: Darwin, based on BSD UNIX 4.4)
- IBM PC (XT, AT, 386 . . .) [from 1981]
  - MS-DOS [1981 . . . 1994]
  - Windows [as of 1985]
  - Linux [as of 1991]

Mobile phones, smartphones, tablets

Technische Hochschule Rosenheim

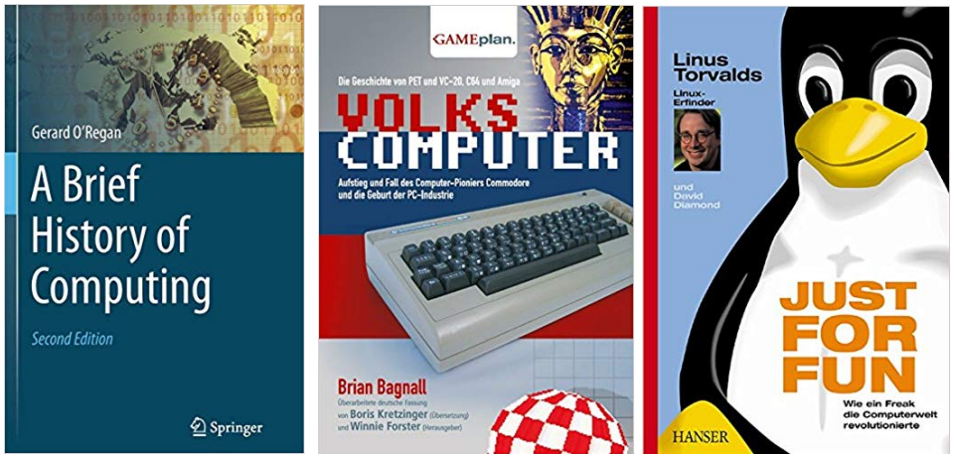
- "classic" mobile phones
  - Nokia, Ericsson, Motorola (and many more)
  - each its own BS
- Symbian smartphones [1997 . . . 2012]
  - SymbianOS
  - first OS for smartphones and PDA: Nokia Communicator, Ericsson P900, . . .
- Apple iPhone [2007 onwards]
  - iOS
  - similar to macOS based on UNIX (XNU kernel)
- Android devices [from 2008]
  - Android, based on Linux
  - most widespread BS of all

Recommended reading

Gerard O'Regan: *A Brief History of Computing*. Springer Verlag, 2012

Brian Bagnall: *Volkscomputer.Aufstieg und Fall des Computer-Pioniers Commodore und die Geburt der PC-Industrie*, Gameplan,2011

Linus Torvalds, David Diamond: *Just for Fun: Wie ein Freak die Computerwelt revolutionierte.*, dtv, 2002



Which OS do you use?

Technische Hochschule Rosenheim

Small survey!

Pingo: https://pingo.coactum.de ... **ID: 050521**

Resultate

Type of OS

- Desktop Operating System
- Server Operating System
- Mobile Operating System
- Supercomputer System
- Embedded , industrial and controller OS
- Cloud based OS

Let's revisit: https://miro.com/app/board/o9J_lm7fCk8=/

Classification according to interaction types

- **Batch processing**: Processing a sequence of batch jobs. A batch job is compiled by the user with all the necessary programmes, data and job control language (JCL) instructions. It is processed completely without user interaction.
  - Examples: *IBM OS/370, OS/390, MVS, BS 2000*.
- **Dialogue mode (interactive processing)**: Constant change between actions of the user (e.g. command inputs) and those of the system (e.g. command execution). The user can influence the workflow in the dialogue at any time.
  - Examples: *MS-DOS, MS Windows 95/98/ME/NT/2000/XP, UNIX/Linux*.

Classification according to operating modes

Technische Hochschule Rosenheim

- **Real-time processing**: Use of a computer system for the control and monitoring of technical processes, whereby the real-time operating system must primarily ensure timeliness (compliance with time conditions).
  - Examples: *VxWorks, VRTX, pSOS, LynxOS, Enea OSE, MS Windows CE, QNX, OSEK/VDX (especially in automotive engineering)*.
- **Distributed processing**: A distributed system consists of several interconnected computers. The operating system here primarily serves to distribute data, resources and workload.
  - Examples: *Amoeba, CHORUS, MACH, Novell Netware, UNICOS, SPP-UX, KSR-OS*.

Classification by user

Technische Hochschule Rosenheim

- **Single User System**: Such a system does not perform recognition or management of multiple users.
  - Examples: *MS-DOS*, *MS Windows 3.X/95/98/ME*.
- **Multi-user system**: It is available (simultaneously) to several users, e.g. via several terminals.
  - Examples: *UNIX*, *IBM OS/390*, *OS/400*, *BS2000*, *OpenVMS*, *Linux*, *Windows 10*.

Classification according to processes

Technische Hochschule Rosenheim

- **Single-process system (single-tasking system)**: The system can process only one task at a time. Another task is only accepted after the current one has been completed.
  - Examples: *CP/M, MS-DOS*.
- **Multi-process system (multi-tasking system)**: It can manage several different jobs simultaneously and, if necessary, process them in parallel or at least quasi-parallel (time-nested). A modified variant is found in multi-threading systems.
  - Examples: *MS Windows 95/98/ME/NT/2000/CE/XP, UNIX/Linux, IBM OS/390, OS/400, OS/2, BS2000, OpenVMS, VxWorks*.

Main components

Technische Hochschule Rosenheim

Concepts for operating systems:

- System architecture
- Interfaces
- Processes
- Memory management
- Access protection
- Scheduling and resource management

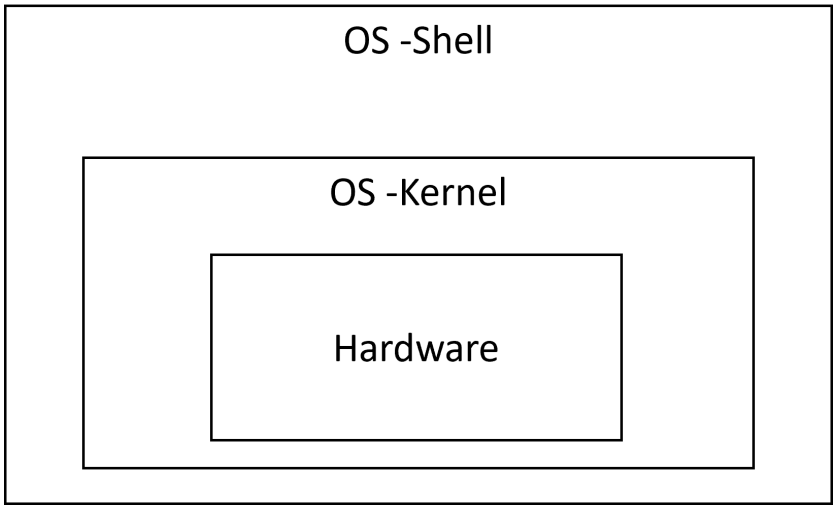
Definition

System software *includes all programmes that enable the efficient and comfortable use of a computer. In addition to the operating system, this includes supplementary, hardware-independent service and auxiliary programs.*

Kernel Architecture

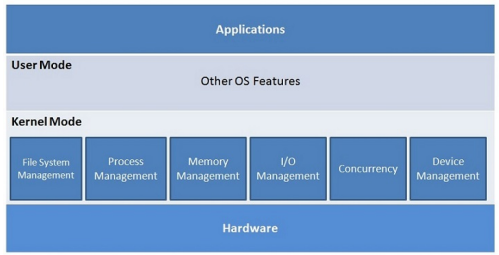
Technische Hochschule Rosenheim

- Kernel is the core of the operating system. Kernel controls everything in a computer.
- There are various kernel architectures:
  - Monolithic architecture
  - Microkernel architecture
  - Hybrid Kernel architecture



Monolithic Architecture

- In Monolithic kernel mode, operating system runs in a single address space.
- Monolithic kernel has all the operating system functions or services within a single kernel.
- This single kernel will run as a single process in a single address space in memory.
- Monolithic architecture enables higher performance however less flexible for modifications to add new features or enhance existing features.



Monolithic Kernel Architecture

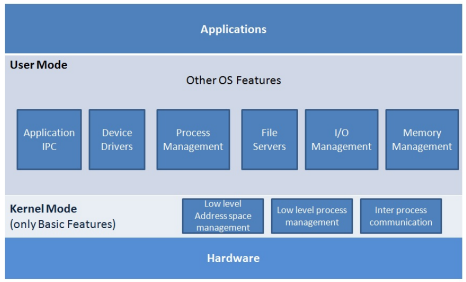
Examples:

- Windows 1, Windows 2, Windows 3, Windows 95, Windows 98, Windows ME
- All Linux distribtions - CentOS, Ubuntu, Red Hat's Fedora, Red Hat Enterprise, Open SUSE and all other linux distributions
- Android uses a modified Linux Kernel

Microkernel Architecture

Technische Hochschule Rosenheim

- Microkernel architecture is an architecture with kernel having the basic interaction with hardware and the basic Inter-Process Communication mechanisms.
- All the other Operating System services exist outside the Kernel.
- Microkernel provides the flexibilities to add new features or modify existing features while slightly affecting performance as it increases amount of interactions between kernel and user mode features.



Micro Kernel Architecture

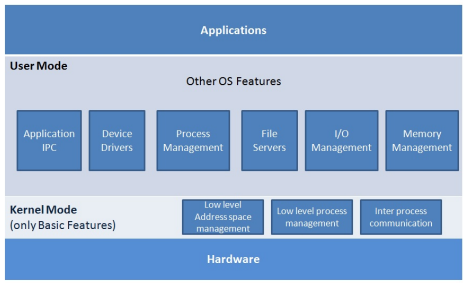
Examples:

- Mach, OKL4, Codezero, Fiasco.OC, PikeOS, seL4, QNX

Hybrid Kernel architecture

Technische Hochschule Rosenheim

- Hybrid architecture tries to get the best features of both monolithic kernel as well as microkernel.
- Hybrid kernel aims to have an optimal performance and the flexibility to modify and upgrade kernel services.



Micro Kernel Architecture

Examples:

- Windows NT, Windows 2000, Windows XP, Windows Vista, Windows 7, Windows 8, Windows 8.1, Windows 10
- Apple's macOS for dektop machines
- Apple's iOS for mobiles uses hybrid (XNU)

Architecture of Linux OS

Technische Hochschule Rosenheim

- Linux architecture has kernel mode and user mode.
- Linux has a monolithic Kernel layer which interacts with the hardware.
- There are other system libraries and device drivers.
- System libraries help Software applications to pass instructions to the Kernel or the Device drivers.
- Device drivers are softwares that interact (take input or provide output or both) with the firmware of the particular device.
- User mode has applications or tools which in turn interact with Kernel directly or indirectly via system libraries, device drivers.
- Kernel space is part of the RAM where the Kernel is loaded.
- Kernel has code which is unique to the hardware platform and code common to all platforms.
- Kernel also has an interface to handle system calls from user layer.

Kernel Architecture of Windows NT

Technische Hochschule Rosenheim

- Windows NT has hybrid Kernel architecture.
- It has the ability to handle Uni Processor and Symmetric Multi Processors(SMP).
- In the kernel mode, Windows NT architecture has a Hardware Abstraction Layer(HAL), a simple kernel, and a collection of services called Executive Services.
- In the user mode, windows NT architecuture has environment subsytem and Integral subsystem.

Windows NT

Technische Hochschule Rosenheim

Environment subsytem

- In the older versions of Windows NT, Environment subsystem had Win32 subsystem, OS/2 subsystem, and POSIX subsystem.
- In the newer versions of Windows NT, Environment subsystem has Win32 subsytem and Windows Subsystem for Linux (WSL)

Integral subsystem

- Integral subsystem has security subsystem, workstation service and server service.
- Security subsystem grants or denies users access to file or other resources and takes care user login authentication and logout.
- Workstation service lets the OS to access file and printer across the network.
- Server service lets the files and printers and resources in the machine to be accessed by external systems

Architecture of Android

Technische Hochschule Rosenheim

Androd OS includes the following components:

- Linux Kernel: Android has modified Linux kernel which has several hardware drivers one for each device.
- Hardware Abstraction Layer: HAL provides a way for developers to link Android OS to access the hardware of choice.
- Android Runtime (ART) is the application runtime of the Android operating system. ART converts the bytecode into native instructions.
- Native C/C++ libraries
- Java API framework
- System Apps

Process

Technische Hochschule Rosenheim

Definition

*A **Program (Program)** is a static sequence of instructions in a programming language using data. It is used to code an algorithm and is generally in the form of a file.*

- Process can be defined as a program in execution.
- Program is a passive entity but process is an active entity.
- System consists of many processes such as operating system processes and user processes.
- Operating system processes execute system code where as user processes execute user code.

Process

Technische Hochschule Rosenheim

Definition

A **process (also: job or task)** is a dynamic sequence of actions (changes of state) that comes about through the execution of a programme on a processor. A process is characterised in particular by its time-varying state. It is created in the operating system as a result of a job._

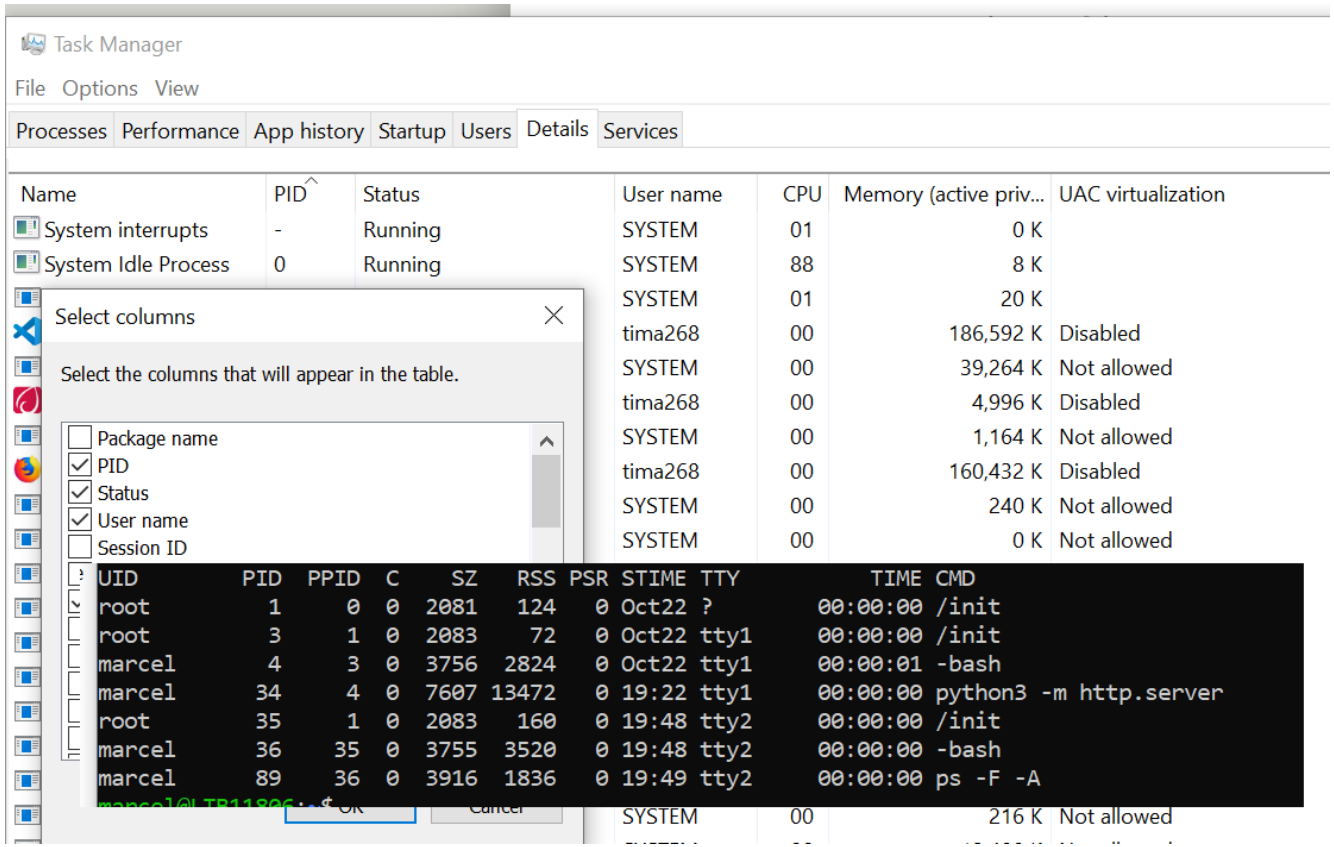
Process control block

Technische Hochschule Rosenheim

In operating system, each process is represented by a Process control block.

- **Process state**: current state of the process. It can be any one of the five states (new, ready, running, waiting and terminate).
- **Program counter** contains address of the next instructions is being to be executed for this process.
- **Process number**: Unique identification of the process.
- **CPU registers**: There are different CPU registers that are used by the process during execution.
- **CPU-scheduling**: contains information about process priority and scheduling.
- **Memory management**: contains the information of the base and limit register, page table and segment tables, depending on the memory system used by the operating-system.
- **Account Information**: contains information about the amount of CPU and real time used, time limits, account numbers, job or process numbers, and so on.
- **I/O Status Information**: contains information of I/O devices allocated to the process, a list of open files, and so on.

Process control block



Process states

Technische Hochschule Rosenheim

At any one time, only as many processes can actually progress in the system as there are processors available. Process management is therefore based on a state model:

- **New State**: As soon as a new process is created, process is in the initial state.
- **Ready State**: Process is waiting to be assigned CPU time. Operating system allocates the processor time to the ready process. Process can come into this state either after the new state or when interruption occured during the running state of the process.
- **Active (also computing, running)**: The process in question has been allocated a processor and is progressing (it is "running").
- ready (also ready to run): The process is only waiting for the allocation of a processor (no further waiting conditions).
- **Waiting (also blocked, suspended, waiting)**: The process must wait for the fulfilment of (at least) one waiting condition.
- **Terminat State**: The process has finished execution.

Process change

Technische Hochschule Rosenheim

The change of processor allocation practically means a task switch or context switch.

1. the active process either voluntarily gives up the processor or is deprived of it. This process is controlled by the process switch (dispatcher).
2. the selection of the process to which the processor is allocated next is made by the scheduler based on a chosen strategy.

Dispatcher and scheduler use the underlying state model as central components of the operating system kernel.

A process change can usually only take place after an interruption (hardware interruption/interrupt or software interruption or system service call).

Process change

Technische Hochschule Rosenheim

A process change requires the following actions in the operating system, among others:

1. Saving the entire context of the interrupted process.
2. Changing the state of the interrupted process to "ready" or "waiting" depending on the reason for the interruption.
3. Scheduling: Selection of the next process to be activated.
4. Change the state of this process from "ready" to "active".
5. Restoring the (saved) context of this process. Loading the processor registers continues this process.

System Service	Unix	Windows (Win32 API)
Create Process	fork(); exec()	CreateProcess()
End Process	Exit()	ExitProcess()

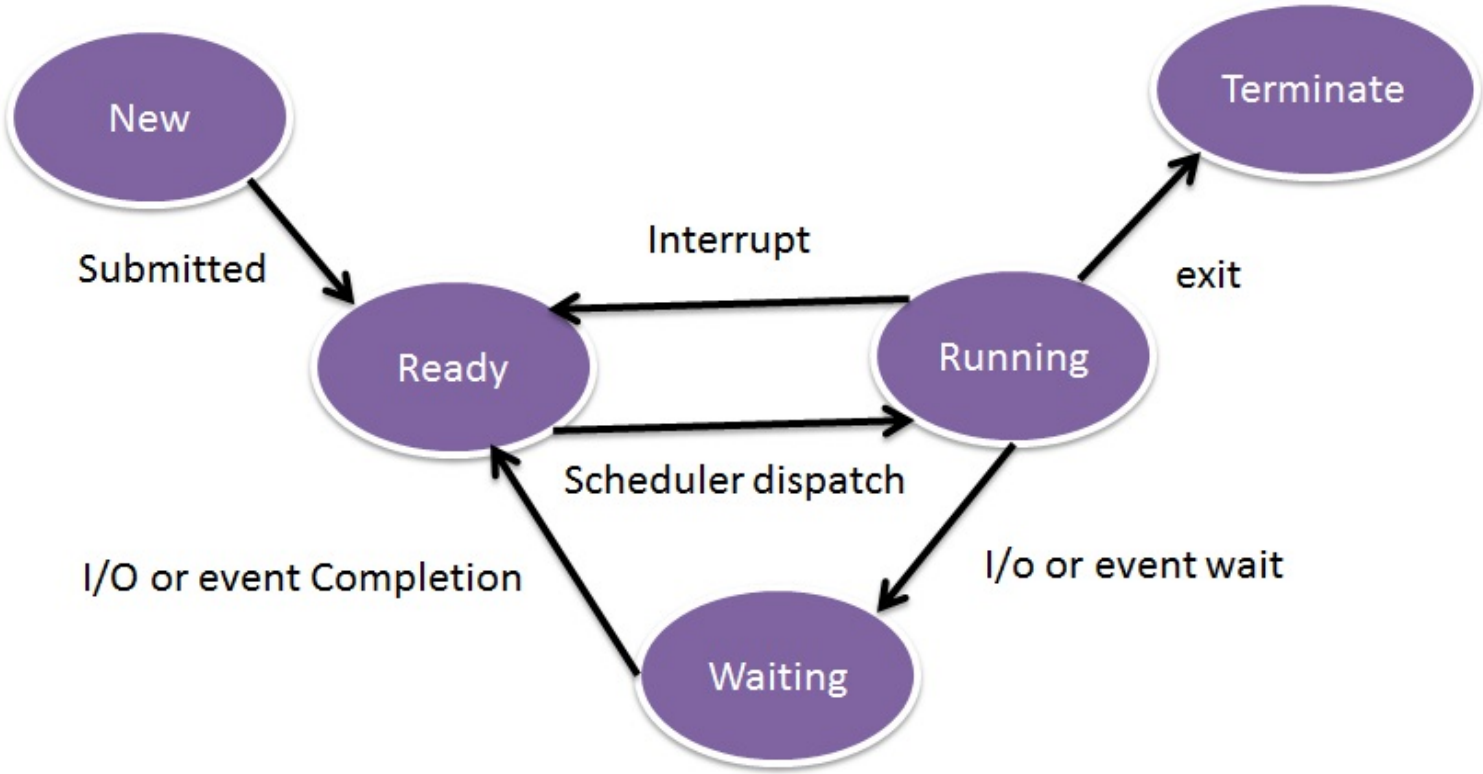
Exercise

Technische Hochschule Rosenheim

Create the state diagram for processes:

- **nonexistent -> New**: A process is created.
- **New -> Ready**: All waiting conditions for the process have been met, it becomes ready to run and awaits processor allocation.
- **Ready -> Running**: The process is allocated a processor (scheduling) and can proceed in its flow.
- **Running -> Ready**: An interrupt happend, the active process is displaced, its processor is withdrawn in favour of another process.
- **Running -> Waiting**: A waiting condition occurs in the process flow (e.g. waiting for input data), it must therefore wait for this condition to be fulfilled and loses the processor.
- **Runnnig -> Terminate**: The process has finished its work and logs off to the operating system, it exits.
- **Waiting -> Ready**: The I/O event occured the process can get back to ready state.

Exercise


New
Terminate
Submitted
Interrupt
exit
Ready
Running
Scheduler dispatch
I/O or event Completion
I/o or event wait
Waiting

Scheduling

Technische Hochschule Rosenheim

Scheduling is organizing OS processes and CPU time. That is, picking one process from pool of processes and getting it executed on the CPU.

Processes are also called as Jobs.

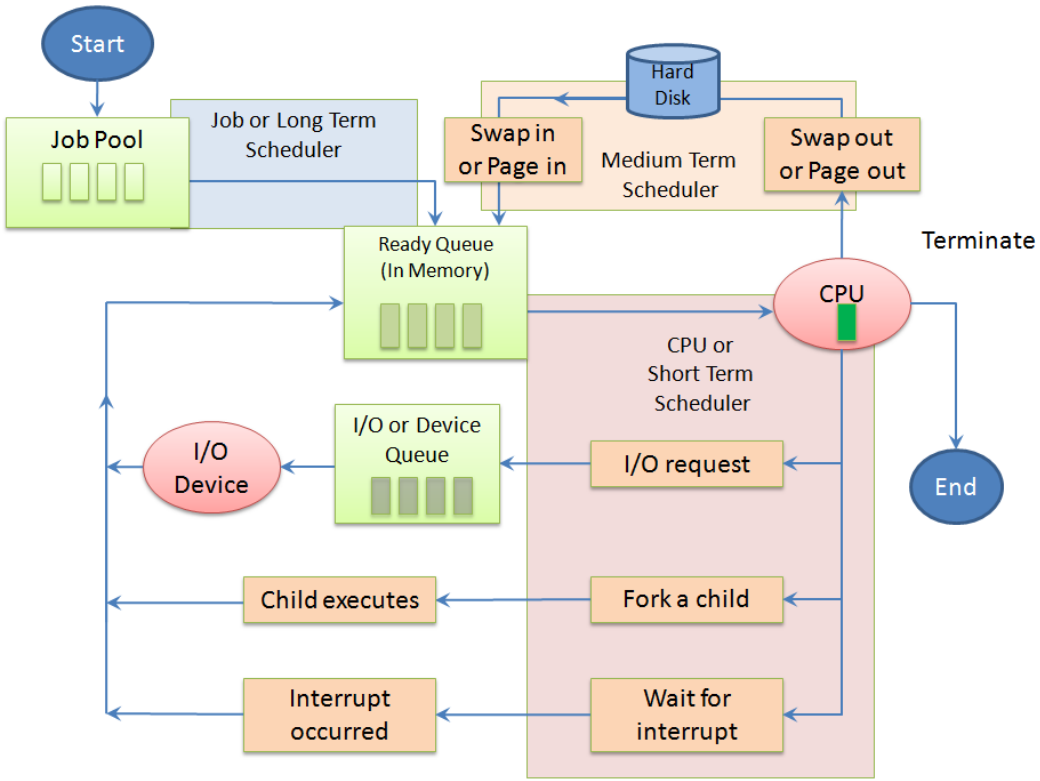
How does scheduling work?

To make scheduling work, it is split into 3 activities.

- Job scheduling is used to select the job to be executed
- CPU scheduling is used to allocate CPU time for a process or job
- Swapping is used to replace one process with another on the CPU

In order to move the jobs between the 3 activities, they are maintained in pools or queues.

Scheduling Queues



Scheduling

Technische Hochschule Rosenheim

Non-preemptive scheduling, also: run-to-completion procedure (non-displacing):

- Old method in which a process is not interrupted until it is finished.
- Not suitable for concurrent users in dialogue mode and also not suitable for real-time processing
- Example: MS-DOS

Preemptive scheduling (displacing)

- Also called priority interruption
- Processes that are ready to compute are suspended
- Suitable for concurrent users
- Time-slice technique required

Retention Time

Technische Hochschule Rosenheim

Definition

Retention time is the total time that a process needs until completion. This time applies from the time it arrives or is called up in the system until it is processed._

Example:

- There are 3 processes A (execution time: 10ms), B (5ms) and C (20ms).
- If the process is processed in the order of arrival, A is processed first, then B and C.
  - A for 10ms
  - B for 10ms+5ms=15ms
  - C for 10ms+5ms+20ms=35ms
- The average retention time is the sum of the individual retention time divided by the number of processes.
  - in the example: 60ms/3 = 20ms

Scheduling

Technische Hochschule Rosenheim

Typical strategies for (processor) scheduling:

- **First Come First Served (FCFS)**: non-displacing strategy with allocation in the order of arrival of processes or jobs.
- **Shortest Job First (SJF), Shortest Processing Time (SPT)**: If the expected processor occupancy time is known, allocation is made to the process with the shortest processing time (e.g. in batch mode).
- **Round robin (RR), time-sharing (or time-slice) method**: In turn, each process receives the processor for a certain time interval (time slice). After this period, the process is displaced. If it requires further processor time, it is again placed at the end of the waiting list. Typical for dialogue mode.
- **Fixed External Priorities (FEP, Priority Scheduling)**: Allocation of priorities (priority numbers) to processes. Resources are allocated to the process with the highest priority. Low-priority processes can be "starved" (starvation).

Scheduling in real-time systems

Technische Hochschule Rosenheim

Fast and predictable reactions to upcoming events are required.

A distinction is made between **hard real time** systems and **soft real time** systems:

- The former must react quickly and be *predicable*.
- In the latter, a delay is possible

A distinction is made between static and dynamic scheduling algorithms:

- In the former, the scheduling decision is already known before the system is started.
- The latter make their scheduling decisions at runtime.

Threads

Technische Hochschule Rosenheim

Because of their extensive context, processes are often **heavyweight** (e.g. in UNIX). In order to achieve finer parallelism, especially within an application, "lightweight" processes or **threads** have been developed.

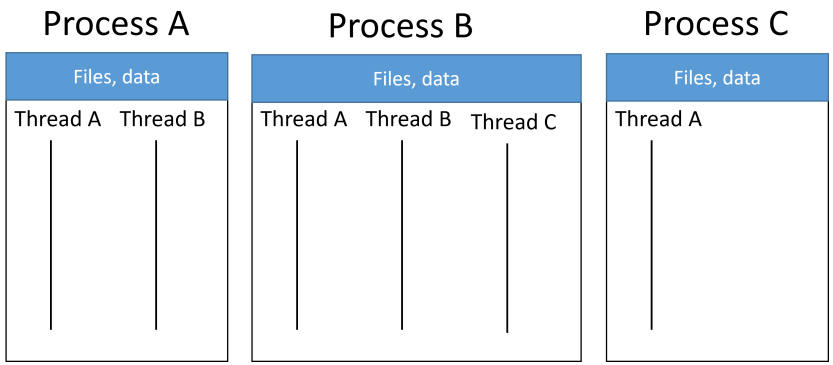
Definition

A **thread** is an activity with minimal context (stack and register) within a process. In this case, each process has at least one (initial) thread. All threads belonging to one and the same process share the same address space as well as other resources of this process.

Threads

Technische Hochschule Rosenheim

- The shared environment considerably reduces the effort required to switch between threads within the same process.
- On the other hand, accesses of threads within a process can no longer be controlled on the basis of address space boundaries, which is usually accepted in favour of higher parallelism.
- Additional synchronisation is often required. Some systems allow the agreement of local memory areas (Thread Local Storage TLS).
- However, switching between threads belonging to different applications still requires an "expensive" address space change and thus higher effort.



Deadlocks

Technische Hochschule Rosenheim

Definition

A **deadlock** is a condition in a system of parallel processes in which some processes are mutually waiting for each other in such a way that none of them can proceed._

The permanent waiting condition in deadlocks results from the fact that each of the stuck processes is waiting for conditions that could only be satisfied by other processes, which in turn are waiting for such conditions, often in the context of resources.

Conditions for deadlocks to occur:

1. resources are used exclusively (critical sections)
2. resources that have already been used cannot be withdrawn
3. processes demand resources without releasing the occupied ones.
4. the processes request resources in a different order so that a closed chain (circle) is created.

Deadlocks

Technische Hochschule Rosenheim

- Prevention of deadlocks: Measures aimed at preventing the occurrence of one of the deadlock conditions by imposing restrictions on the processes.
  - Allocate resources to only one process.
  - Use resources indirectly, e.g. a printer (spooling).
- Avoidance of deadlocks: For each resource request, the operating system analyses whether, after allocation, only safe, deadlock-free states would occur in the further course (in the worst case). If this cannot be guaranteed, the current request is not fulfilled "for safety's sake", even if this would be possible at the moment.
- **Detect and delete: The operating system cyclically checks the state of all processes (and resources) for possible deadlocks.
  - Deadlock can only be resolved by prematurely terminating at least one stuck process.

Summary

Technische Hochschule Rosenheim

We looked at the following in detail:

- Operating systems
- Architectures
- OS Categories
- Processes and Threads
- scheduling
- Deadlocks

Final thought