Systems Software and Distributed Systems
Summary of Key concepts

Tobias Famos

June 8, 2019

Exercise 1 Takaways

Definitions that occurred in the Exercise 1:

Programs Executables that define the ways in which system resources are used to solve computing problems of Users. One can distinguish between **System Programs** and **Application Programs**. System Programs are the ones that come with the Operating system and application programs are manually installed.

Operating System Is the Part that controlls and coordinates the use of hardware among various applications and users.

Kernel The Kernel is the most central piece of an operating system. It handles and controlls all the other parts of the operating system.

Bootstrap program is also known as firmware. The firmware initializes all aspects of a system. It also loads the operating system kernel and starts the execution.

Interrupts An interrupt is a signal to the processor indicating that an event needs immediate attention. E.g. the I/O devices send an interrupt when they are done handling input.

Traps A trap is a special kind of interrupt emitted by software indicating that an error or a user request has occurred. A trap is sent for example after a division by zero.

System Call A system call is the way with which a program requests a service from the kernel. This could be a hardware related service (like accessing the hard disk), the creation of new processes or similar things.

Multiprocessing Systems A Multiprocessing System is a system with multiple processors. They can execute multiple processes simultaneously. There are two different types: **Symmetric** and **Asymmetric**:

Symmetric multiprocessing architecture Here all the CPU's are symmetric. Meaning: They all have the same rights and communicate with each other. They execute tasks from the same shared queue and have shared memory. They all run tasks of the Operating System.

Asymmetric multiprocessing architecture Here there is one master processor and one or more slave processors. CPU's are not Symmetric. Only the master processor runs the OS tasks. In case the master process fails, a other slave gets to be the master.

Kernel mode Is a mode in the access mode mechanism. When a process runs in kernel mode, the process has unrestricted access to all the hardware. It is reserved for the most trusted OS processes.

User mode Is the second mode in the access mode mechanism. A process running in user mode does not have access to the underlying hardware. It must use the system API to use the Hardware (e.g. to read from memory). Most code executed on a computer will run in user mode.

Exercise 2 Takeaways

Definitions and concepts that occurred in Exercise 2:

Linux File structure - usr is Unix System Resources NOT USER. contains files related to user such as application files and

Monolithic Kernel A monolithic kernel consists of everything below the system-call interface and above the physical hardware. It provides a file system, CPU scheduling, memory management and other operating system functions. This is a large number of functions for one level and is not that reliable since a lot of code runs on kernel level.

Micro Kernel A micro kernel is the a kernel where a lot of the functionalities are lifted up to the user space. Things like device drivers, protocol stacks and the file system are removed form the kernel and run as separate processes (often in the user space). They communicate using message passing. This leads to an easier to extend kernel, that can also easier be ported to new architectures. Furthermore is it more reliable and more secure. The downside is a performance loss due to the communication from the processes to the kernel.

Hybrid Kernel A hybrid kernel (also called a **Macro Kernel**, is a mix of the two previous kernel models. It has some of the services in the user space and some in the kernel space. It is not strictly defined which ones belong where. This leads to performance improvements since less communication has to take place.

BIOS **B**asic **I**nput **O**output **S**ystem is a non-volatile firmware used to perform hardware inizialisation during the booting process

CMOS **C**omplementary **M**etal-**Oxide**-**S**emiconductor is a technology for constructing integrated circuits used for example in microprocessors.

EPROM **E**rasable **P**rogrammable **R**ead-**O**nly **M**emory is a non-volatile (keeps data when powered off) memory chip.

Exercise 3 Takeaways

Definitions and concepts that occurred in Exercise 3:

States of a Process :

New: The process has is being created. After creation it changes to ready
Ready: The process is waiting to be assigned to a processor. When it is assigned it changes to running
Running: The process is executing instructions. From here it can either go back to ready, via a interrupt, go to terminated, when it has done all its work or go to waiting, when doing I/O or some other reason for waiting.
Waiting: The process is waiting for some event to occur. Can go back to ready from here.
terminated The process has finished execution. The used resources are reallocated by the OS.

Race condition A Race condition occurs if two a system attempts to perform two or more operations at the same time, but the operation must be done in proper sequence to be done correctly.

Starvation Starvation is the state where a process is denied its resources it needs to work. It can be caused by e.g. an error in scheduling.

Fork The fork system call creates a new child process that is equal to the parent process (except the Process ID)

Pipe A Pipe is a mechanism for inter process communication. Data written to a pipe can be read by another process.

SIGINT This is the **interrupt signal**. It requests an interrupt of the process. E.g.: when pressing crtl+c during execution in terminal, SIGINT is sent to the process.

SIGKILL This is the **killing signal**. It kills the process immediately and cant be caught or ignored.

SIGSTOP This is the **stop signal**. It instructs the operating system to stop a process for later resumption.

Exercise 4 Takeaways

Definitions and concepts that occurred in Exercise 4:

Response Time The Response time is the time until a process sends his first response. It is (I think) The same as the turnaround time, when the first response is sent at the end (which is done quite often).

Round-Robin Schedule A round-robin schedule is a schedule where, based on a time quantum (typically 10-100ms) processes get the CPU. They get it in turn so that every process gets its time.

CPU Burst A CPU burst is the execution of an instruction on a CPU.

Turnaround Time The Turnaround Time is the time needed until a process can finish. It is the waiting Time plus the actual execution time.

Exercise 5 Takeaways

Definitions and concepts that occurred in Exercise 5:

1 Exercise 7

External vs internal Fragmentation Fragmentation is the unused memory space generated when allocating memory to a process. The difference between internal and external fragmentation is whether this unused memory is assigned to a process or not. In **Internal Fragmentation** there has been too much memory assigned to a process that is now unused or wasted. In **External Fragmentation** the memory between assigned parts of memory is not used. External Fragmentation is unused memory that could, theoretically still be assigned.

Exercise 6

dirty bit

Exercise 9

Middleware

Distributed Operating System

Network Operating System

Transparency of a Distributed System

Openness of a Distributed System

Scalability of a Distributed System

Code Mobility Is the ability in a distributed system to migrate (move) code from one node to an other to execute it on the second node.

Thin Clients A thin client is a client in a client server model, that does not need to have extensive computation power. It only needs to execute some minor parts, while the intense parts are executed by the server.

Overlay Networks Is a network that has been build on top of another network.

Forward Pointer A forward pointer can be used, when a resource has been moved. The old address now has a pointer to the new address of the resource. When some process wants to access the resource, it only has to follow the pointers until it finds the moved resource.

Exercise 10

Centralized Approach In the centralized approach to achieve mutual exclusion there is one coordinator that gives access to a resource. He gets a request from a process, then sends a OK to the process and revives a release form the process when he is done. If a process has to wait, he just does not get a OK. It is not possible for the process to know whether he has to wait, or the coordinator has failed.

Distributed Approach In the distributed approach to achieve mutual exclusion a node sends a Request to everyone and waits for the ok of everyone (or the majority in Maekawa Algorithm). It is very error prone, since the failure of one process can prohibit all the others from doing anything.

Ring-Token Approach In the ring token approach to achieve mutual exclusion a token is passed in the ring. Only the Process having the token can access the resource. If a process does not need the resource or is done with it, it passes the token on to the next process.

Maekawa Algorithm

Bully Algorithm The bully Algorithm is used to handle the failure of a coordinator in a centralized approach. When one node notices that the coordinator has failed, the node starts an election: It sends every other node with higher ID an election message upon which the later node also starts an election and sends an ok message to the former node. The node that does not get any OK messages (thus the node with the highest ID that is working) is the new coordinator and sends a coordinator message to all the other nodes.

Ring Algorithm The Ring algorithm is also for replacing a failed coordinator. The election works as follows: The noticing node starts the election, writes its id into an array and passes that array in a ring. Every other node writes its ID into the array. The staring node receives the array once again and selects the Process with the highest ID to be the coordinator.

Reliable Multicast Algorithm With a reliable multicast algorithm, either all working nodes or none of the working nodes in a group get the message. The mutlicast is not reliable if some get the message while others don't. This is achieved by the reciver helping the sender and sending the message also to all the other nodes.