Introduction to Operating System Concepts

Praktikum Kernel Programming
University of Hamburg
Scientific Computing
Winter semester 2015/2016

Konstantinos Chasapis
Konstantinos.chasapis@informatik.uni-hamburg.de

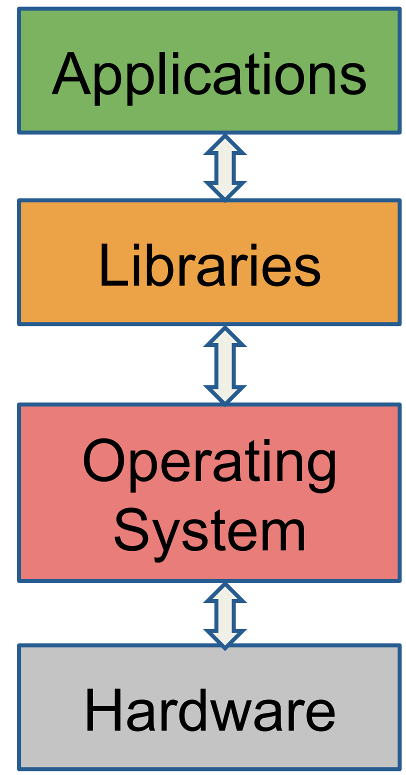
Outline

- What is an Operating System
- History of Operating Systems
- Computer Hardware
- Types of Operating Systems
- Process Management
- Memory Management
- Storage Management

*disclaimer: some of the topics presented here are incomplete.

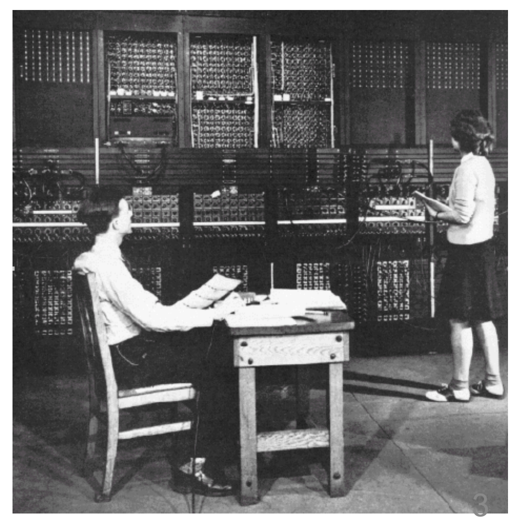
What is an OS

- Hard to define
- Abstracts a set of hardware resources
  - High level interface instead of machine code
    - e.g. file storage on top of block devices
- Resource management
  - Multiplexing (sharing) resources
    - e.g. assign CPU time to applications



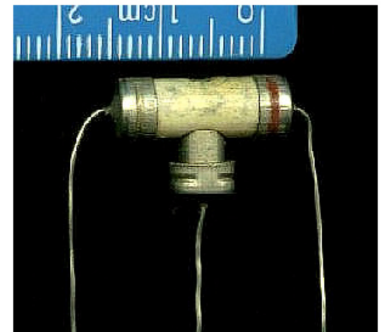
1st Generation

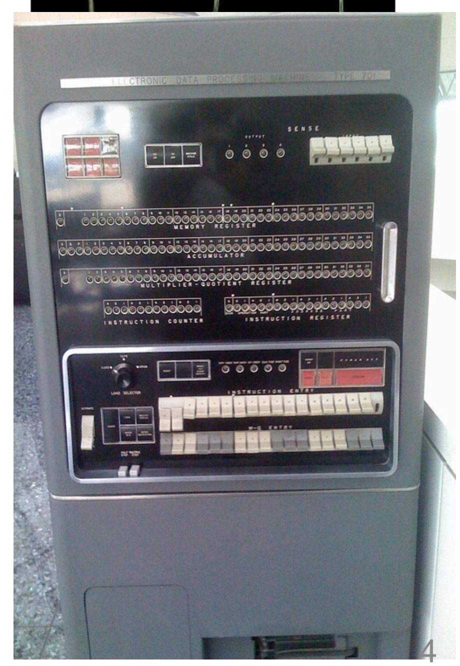
- Vacuum Tubes (1945-55)
  - ~20,000 vacuum tubes were used
  - Programming was done in absolute machine code
  - Assembly language was unknown
  - Each program used the machine exclusively
  - Most famous ENIAC
    - Announced in 1946
    - Solve large class numerical problems



2nd Generation

- Transistors and batch systems (1955-65)
  - Designers / Builders / Operators / Programmers / Maintainers
  - Programmers first wrote the program on paper, then punched it on cards
  - Card readers to read the program source
  - Output stored on tapes and also printed
  - 1st use of compilers (FORTRAN)





3rd Generation

- ICs and Multiprogramming (1965-1980)
  - IBM 360 Mainframe
    - Multiprogramming
    - Several programs in memory at once with separate memory, overlap I/O with computation
  - Timesharing
    - Each user has an online terminal
    - CTSS (Compatible Time Sharing System)
    - MULTICS (MULTiplex Information and Computing System)
    - UNIX, a stripped-down version of MULTICS
    - BSC (Berkeley Software Distribution)

4th Generation

- Personal Computers (1980-today)
  - SYSTEM V, 1st commercial UNIX operating System (1983)
  - LSI (Large Scale Integration)
  - IBM PC (early 1980s)
    - Intel 80286 CPU
    - DOS (Disk Operating System)
    - MS-DOS (Microsoft DOS)
  - LISA
    - First computer with GUI
    - Protected memory, preemptive multitasking

5th Generation

- Smartphones (1990-today)
- Symbian OS
- RIM's Blackberry OS
- Windows Mobile
- Android
- iOS

Modern Operating Systems

Outline

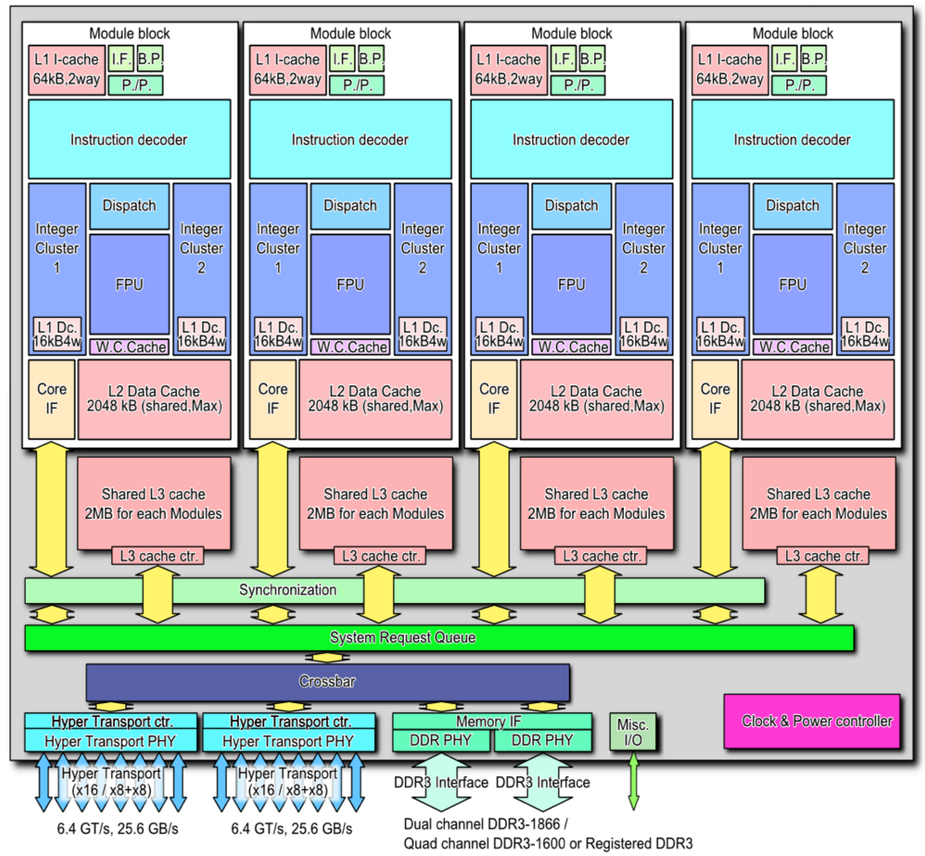
- What is an Operating System
- History of Operating Systems
- Computer Hardware
- Types of Operating Systems
- Process Management
- Memory Management
- Storage Management

Computer Hardware

- Processor
- Main memory
- I/O devices
- Disk
- Busses

Processor (CPU)

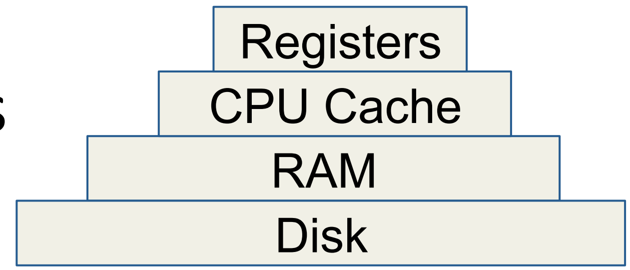
- Specific instruction set
- Basic cycle
- fetch
- decode
- execute
- Multiple cores
- Multiple levels of cache memory

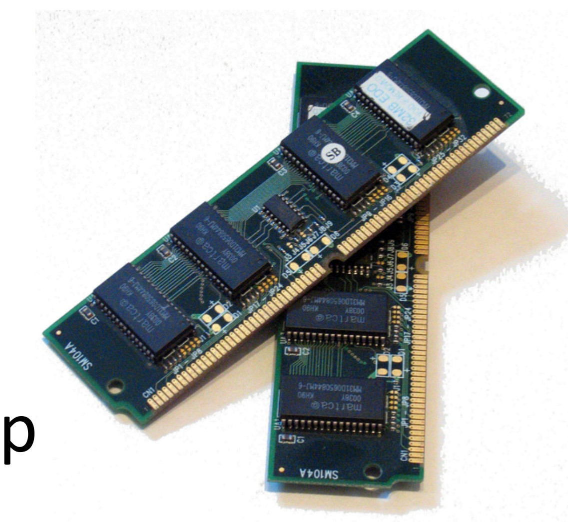


https://commons.wikimedia.org/wiki/File:AMD_Bulldozer_block_diagram_%288_core_CPU%29.PNG

Main Memory

- Random access memory (RAM)
  - Large but slow
  - Volatile, loses data in power loss
    - Static (SRAM)
    - Fast but expensive
    - Used as CPU cache
  - Dynamic (DRAM)
    - Store bits in capacitors
    - Require refresh to retain state
    - Larger than SRAM but slower
  - Non-Volatile, keep state without p
    - Not yet mature technology



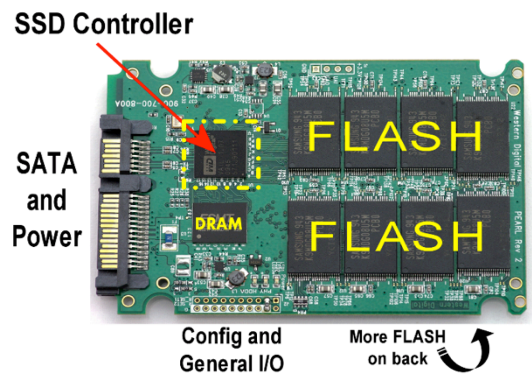


Disks

- Mechanical device
- High capacity
  - 200x size of RAM
- Slow
  - 1000x slower than RAM
- Magnetic recording
- Moving parts
  - Rotating platters
  - Head
- Seek time

I/O Devices

- Usually two parts
  - The actual device
  - Controller
    - Chip(s) that physically controls the device
    - “Talks” to the operating system (OS)
    - Device driver
      - Software that connects OS with the controller

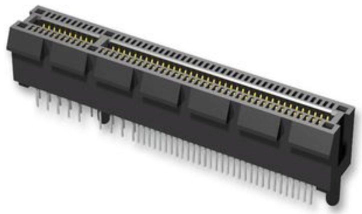


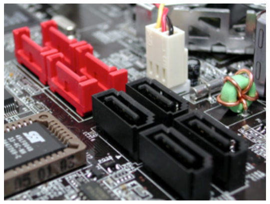
I/O Devices (cont.)

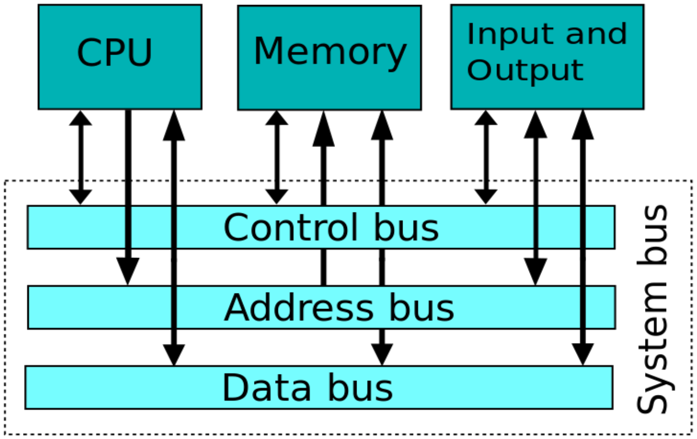
- Forms of communication
  - Busy waiting
    - OS waits until device response
  - Interrupts
    - Device inform the OS that "something" happened
  - Polling
    - The OS regularly checks the device
  - DMA (Direct Memory Access)
    - Special hardware
    - Data movement from memory to controller without going through the CPU

Busses

- Connect computer components
  - Parallel (multiple wires)
    - Carry data words in parallel
  - Serial (lanes)
    - Carry data in serial form in each lane
- Different speeds
- Different functionality
- Examples
  - PCIe (Peripheral Component Interconnect express)
  - SATA (Serial ATA)







Outline

- What is an Operating System
- History of Operating Systems
- Computer Hardware
- Types of Operating Systems
- Process Management
- Memory Management
- Storage Management

Types of OSs

- Multi-user
  - Multiple users access the computer simultaneously
- Single-tasking
  - Only one running program
- Multi-tasking
  - Allows more than one program to run in parallel
- Two types:
  - Pre-emptive, the OS interrupts the running program and assigns the CPU to the next
  - Co-operative, each process give time to the others
- Real-time
  - Aims at executing real-time applications

Types of OSs (cont.)

- Distributed
  - Manages a group of independent computers and makes them appear to be a single computer
- Templated
  - A single virtual machine image as a guest operating system, then saving it as a tool for multiple running virtual machines
- Embedded
  - Designed to be used in embedded computer systems

Monolithic kernel

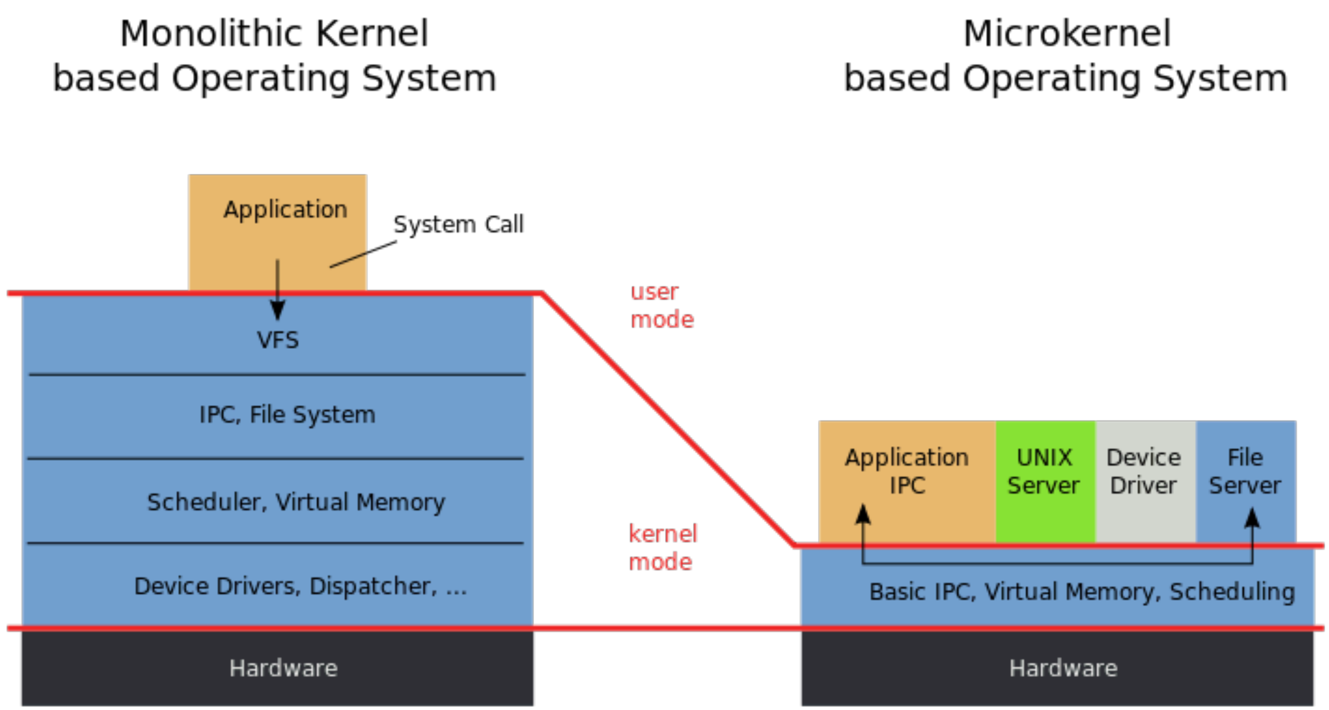
- Single image that runs in a single address space
  – A set of primitive operations are implemented in the operating system level
    - Process management
    - Memory management
    - Device Drivers
  – Trivial (IPC) Inter Process Communication
  – Easy to design
  – Difficult to maintain and extend
  – Examples:
    - MULTICS, SunOS, Linux, BSD

Micro-kernel

- The minimum amount of software that provides the mechanisms needed to implement an OS
  - Also known as μ-kernel
  - Provides
    - Built-in IPC
    - Low level address space management
    - Thread management
  - Easy to extend
  - Performance penalties (requires IPC calls)
  - Examples
    - Symbian, Mac OS, WinNT

Monolithic vs. μ-kernel

Everything that runs in kernel mode defines the OS



Source: http://en.wikipedia.org/wiki/Microkernel#mediaviewer/File:OS-structure.svg

Outline

- What is an Operating System
- History of Operating Systems
- Computer Hardware
- Types of Operating Systems
- Process Management
- Memory Management
- Storage Management

Processes

- An abstraction of a running program
- Every process “thinks” that it runs alone
  - Assume dedicated resources (address space, CPU)
- Has a single control flow (program counter)
  - Which operation is currently executed
- Programmer’s role
  - Defines what the process will do
- Operating system role
  - Process management

Process Management

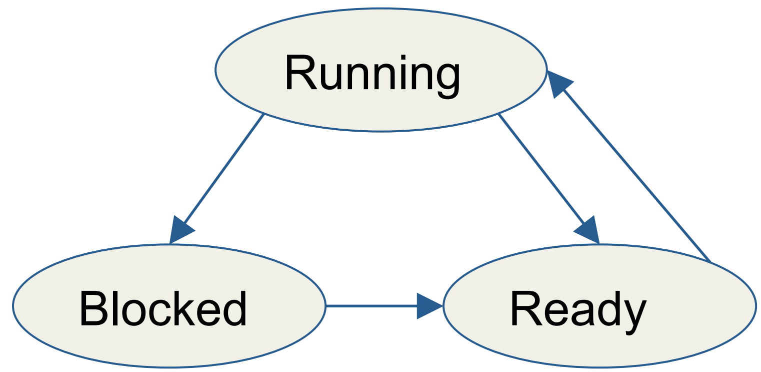
- Process creation
  - System (initialization, system call)
  - User (executes a new program)
- Process termination
  - Normal exit
  - Error (voluntary), Fatal error (involuntary)
  - Killed by another process
- Process Scheduling
  - Assign CPU to the processes
  - Determine which process will execute next
  - Switch between different processes (context switch)

Process Hierarchies

- One process invokes the creation of another
  - Parent process, initiated the creation
  - Child process, the created process
    - Can also create processes
- Daemon process
  - Runs in the background
  - Not controlled by the user
  - Creation
    - Create a new process (child)
    - Kill your parent
    - Continue with the execution

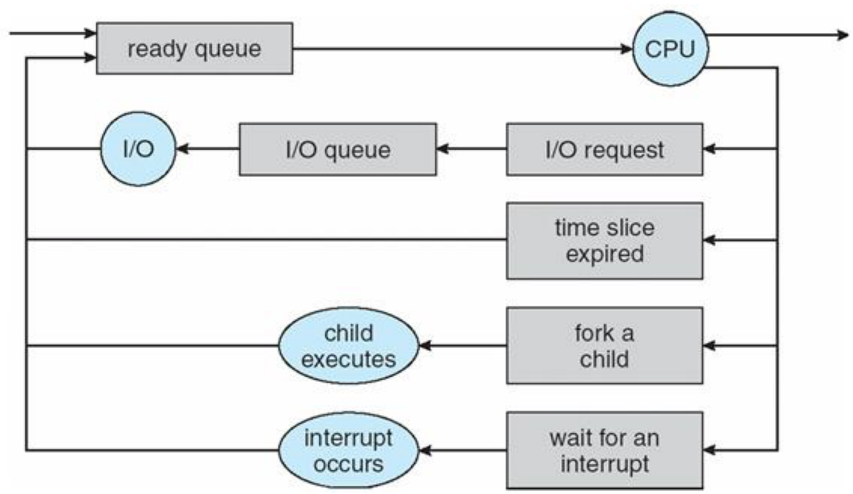
Process States

- Running
  - Currently using the CPU
- Ready
  - Ready to execute
- Blocked
  - Unable to run
  - Waits until “something” external to happen



Process Scheduling

- Hold processes in queues
  - job queue, all processes
  - running processes, able to run
  - device queue, waiting for I/O completion



Process Scheduling

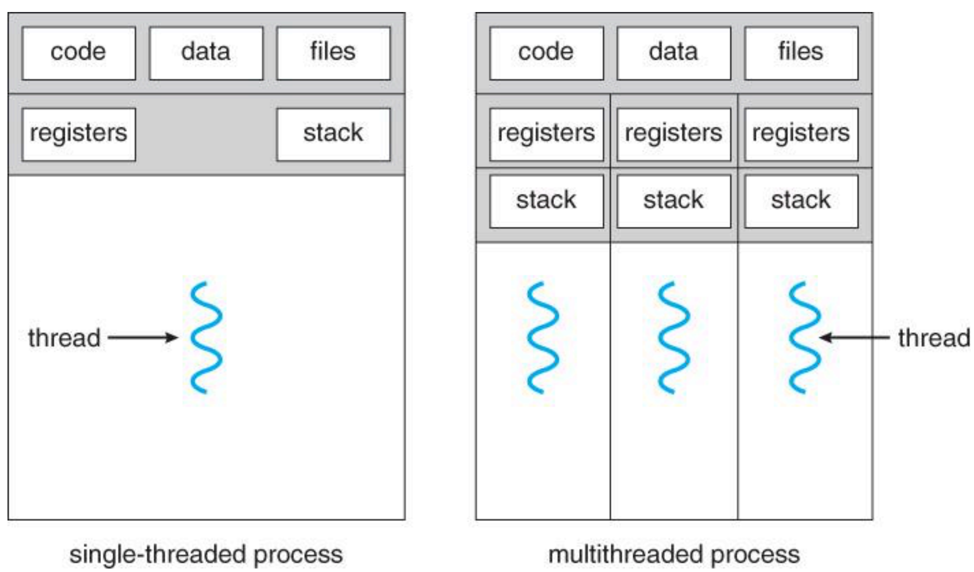
- Scheduler
  - Process that migrates processes between queues
- Policy Considerations
  - Prioritization, which will execute first
  - Fairness, every process gets a fair amount of CPU
  - Starvation, does not get CPU time
  - Maximize CPU utilization

Context Switch

- Switch the CPU to another process/thread
  - Save program "context"
    - Registers
    - Stack
    - Memory
  - Implies overhead

Threads

- Why not having more than one control flow inside a process?
  - Same problem solved by more than one
- Threads are “mini” or lightweight processes



Thread vs. Process

- Per thread
  - Stack
  - Registers
  - Private control flow (Program counter)
- Per process
  - Address space
  - Global variables
  - File descriptors
  - Child processes
  - Signals, signal handling
  - Accounting information

Threads concurrency

- Many threads running in parallel
- Cooperate to solve a single problem
  - Split the problem in sub-problems
- Require coordination/synchronization
  - Control CPU usage
  - Control access to shared resources

Array Multiplication

- Assume that we have to multiply two arrays
- Single threaded solution
  - Start from the beginning of the matrix and calculate every cell one at a time
- Multithreaded solution
  - Split the array in sub-arrays
  - Assign each sub-array to a different thread
  - Run threads in parallel
  - Wait until all threads finish the calculations
  - Output matrix is ready

Synchronous/Asynchronous execution

- Synchronous execution
  - Operation returns when the execution finishes
  - Example
    - I will continue cooking when the water has boiled
- Asynchronous execution
  - Operation returns immediately after execution initialization
  - Check later or notify when the execution finishes
  - Example
    - I put the water to boil and continue with the rest of the recipe
    - After 5 minutes I check if the water has boiled

Web server example

- Web server tasks
  - Accepts requests for a web page
  - Fetches the data from the storage system
  - Replies by sending the data
- Single thread solution
  - Each request is executed serially
  - Only a single request is being processed at a time
  - Every request has to wait for the previous to complete all tasks
  - I/O is slow, lots of time waiting
  - Can not utilize multiple cores

Web server example

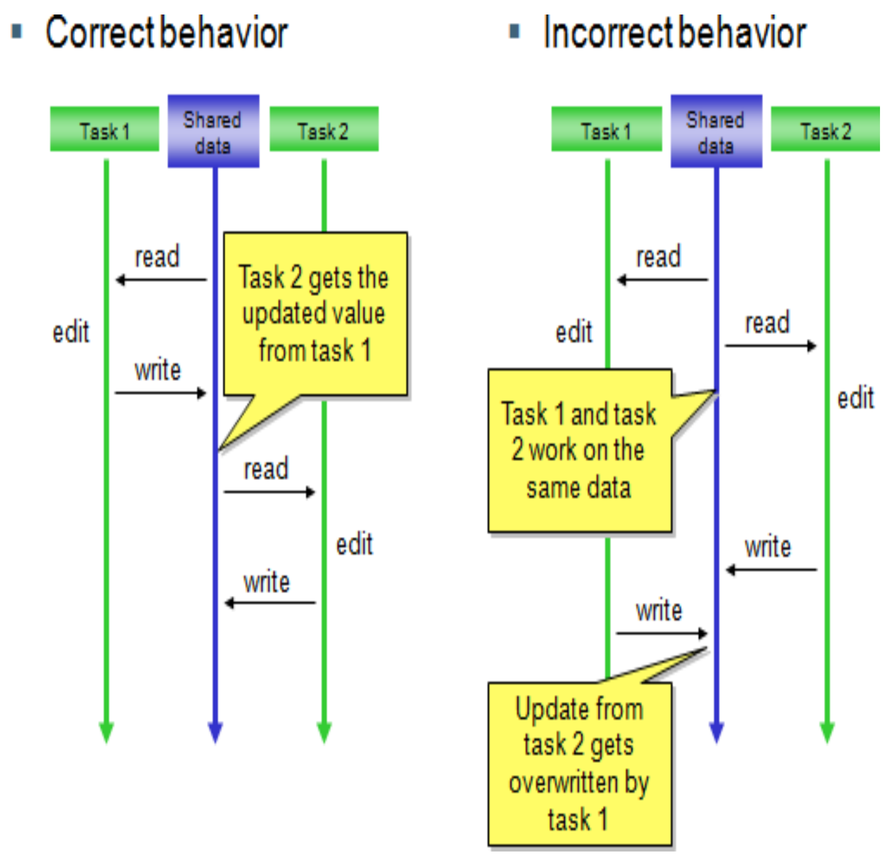
- Multi-threaded
  - Create a thread for each request
  - Multiple requests can be served in parallel
  - Thread execution
    - Accept request
    - I/O request, fetch data from disk
    - This request call can be asynchronous
    - Reply request, send data
  - Multi-core utilization
  - Overlap I/O with computation

Synchronization issues

- Critical region
  - Part of the program that accesses shared resources
    - Global variables
    - Shared memory
    - File descriptors
  - It is safe to be executed by only one process/thread at a time
- Race condition
  - The successful execution depends on the sequence or timing of the other threads/processes

Synchronization issues

- Assume two tasks run in parallel
- Task 1 reads a global variable and updates the value
- Task 2 uses the same variable as Task 1



Mutual Exclusion

- Prevent parallel executions of the "critical region"
- Basic methods for mutual exclusion
  - Sleep and wakeup
    - One task explicitly puts the other one to sleep when it enters the critical region
    - Wakes it up when it exits the critical region
  - Semaphores
    - Special integer variables to count sleeps and wakeups
  - Mutex
    - Simplified semaphore only two states lock,unlock

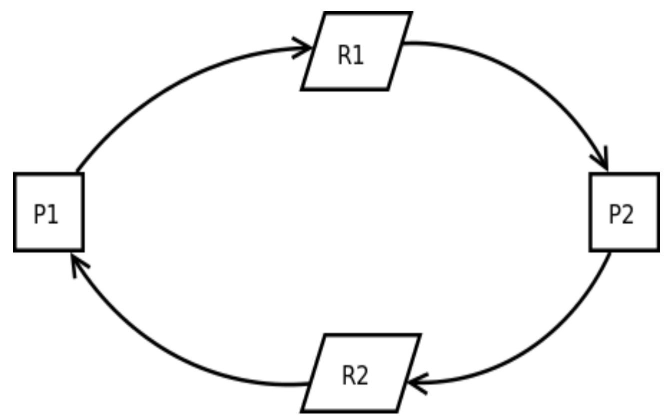
Producer/Consumer

```
#define N 100
int count=0;
procedure producer() {
    while (true) {
        item = produceItem();
        if (count == N)
            sleep();
        insert(item);
        count = count + 1;
        if (count == 1)
           wakeup(consumer);
    }
}
```

```
procedure consumer() {
        while (true) {
            if (count == 0)
                sleep();
            item = remove();
            itemCount = Count - 1;
            if (count == N - 1)
                wakeup(producer);
            consumeItem(item);
        }
    }
```

Deadlock

- One process waits for the other
- Compete for resources
  - P1 needs resources from P2
  - P2 needs resources from P1
- Always allocate with the same order
- Hard to debug

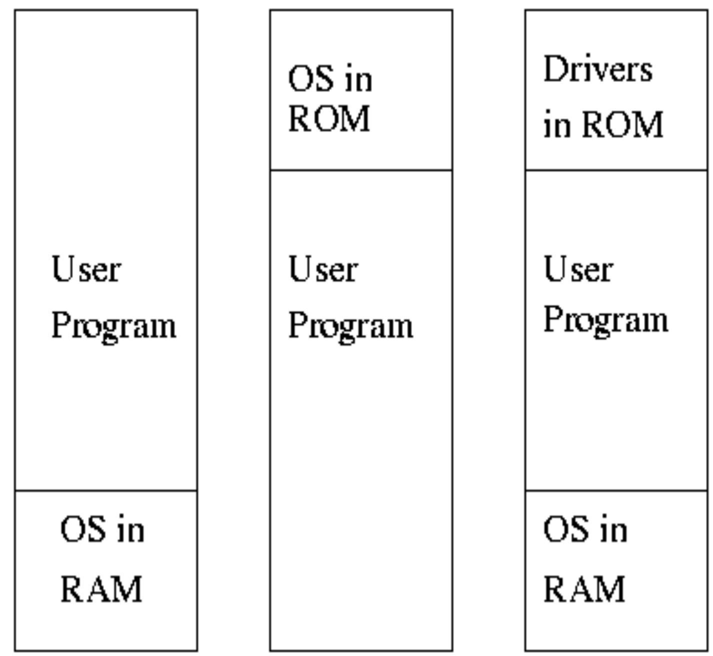


Outline

- What is an Operating System
- History of Operating Systems
- Computer Hardware
- Types of Operating Systems
- Process Management
- Memory Management
- Storage Management

Access Physical Memory

- Use absolute physical addresses
- No memory abstraction
- Run multiple processes
  - OS saves the entire memory
  - OS loads the next process's memory
  - Runs the next process



Memory Abstraction

- Allow multiple processes to exist in memory
- Protection
  - Ask for permission to access an address in memory
- Relocation
  - Allocate physical memory dynamically
  - Relocate the process to a different region
- Sharing
  - Control which data will be shared between processes
- Distinguish physical to logical memory
  - Manage the memory hierarchy

Address Space

- Set of address that a process can use
  - Memory abstraction
  - Map each process's address space into different parts of physical memory
- Use of two registers
  - Base, start address of the program
  - Limit, length of the program
  - Only OS modifies these registers
- Process memory access
  - Logical address + Base register

Swapping

- If main memory is not enough
  - Use disk to temporarily store process data
- Moving from memory to disk and vice versa is called “swapping”
- Slow process
  - Involves disk I/O

Virtual Memory

- Map logical addresses to physical addresses
- Each process has a private address space
- Address space divided to pages
- Pages mapped to physical memory
- OS keeps a page table
  - Translation from virtual to physical pages

Outline

- What is an Operating System
- History of Operating Systems
- Computer Hardware
- Types of Operating Systems
- Process Management
- Memory Management
- Storage Management

Storage Management

- Applications need permanent data storage
  - Persistency
- Hardware provides block level accesses
  - Use fixed-size blocks
  - Every block has its own address
  - Transfers in multiples of blocks
  - Seagate provides object based access (key/value)
- OS provides methods to store and retrieve data from disks

File System

- Move data from memory to disk and vice versa
- Export files abstraction
  - Name data collections with names -> file name
  - Map logical continuous units to arbitrary disk blocks
  - Access them in byte level
- Export file system tree (hierarchy)
  - Group files collection in directories

File System

- Disk space management
  – Keeps track of used space
  – Free space that is not used by the users
- Hold system state
  – System might be in inconsistent state during an unexpected failure
    - Update system variables
  – Journal used to keep previously consistent state
  – Recover from system crash

File System - Data protection

- Each user controls files/directories permission
- Linux permission three groups
  - owner, the file/directory owner
  - group, the group that the user belongs
  - all users, the rest of the users
- Three basic permission types
  - read, can read the file
  - write, can modify the file
  - execute, can execute a file (denotes executables)

Questions?

Konstantinos Chasapis

Konstantinos.chasapis@informatik.uni-hamburg.de