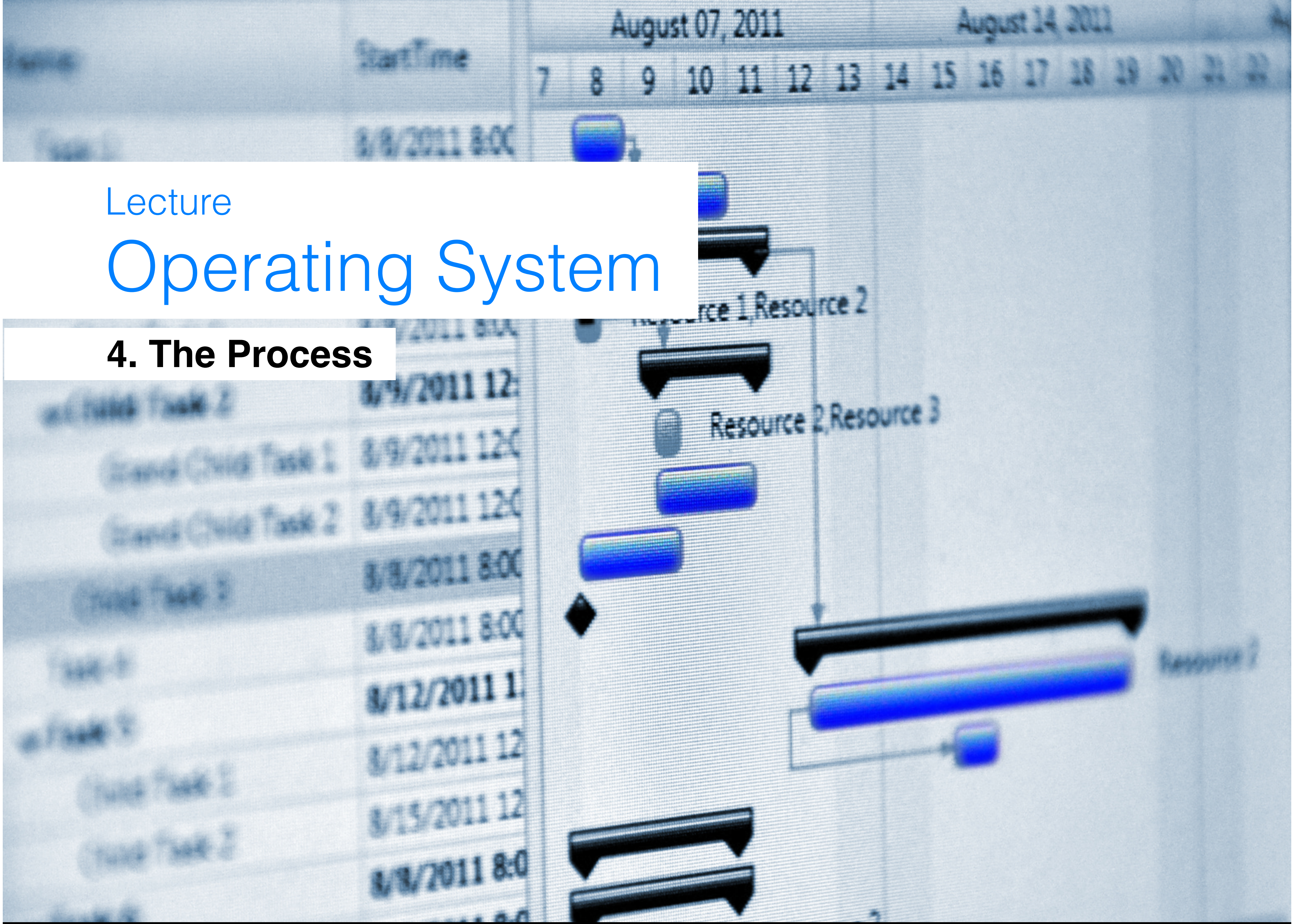


Lecture

# Operating System

## 4. The Process

Professor Dr. Michael Mächtel

# 4. The Process

1. **What is a process?**
2. **Why is limited direct execution a good approach for virtualizing the CPU?**
3. **Execution state and modes of processes**
4. **Policy and Mechanism in general**

# What is a process

- A process is a **running programm**
  - Stream of execution instructions; Running piece of code; …
- A process is different than a program
  - **Program**: Static code and static data
  - **Process**: Dynamic instance of code and data
- What is **process state**?
  - Everything that the running code can affect or be affected by Registers
    - General purpose, floating point, status, program counter, stack pointer
  - Address space
    - Heap, stack, and code
  - Open files

# What is a process (Cont.)

- A process comprises:
  - Memory (adress space)
    - Instructions
    - Data section
  - Registers
    - Programm counter
    - Stack pointer
- Can have multiple process instances of same program
  - Example: many users can run **ls** at the same time

# Process API

- These APIs are available on any modern OS:
  - **Create**
    - Create a new process to run a program
  - **Destroy**
    - Halt a runaway process
  - **Wait**
    - Wait for a process to stop running
  - **Miscellaneous Control**
    - Some kind of method to suspend a process and then resume it
  - **Status**
    - Get some status info about a process

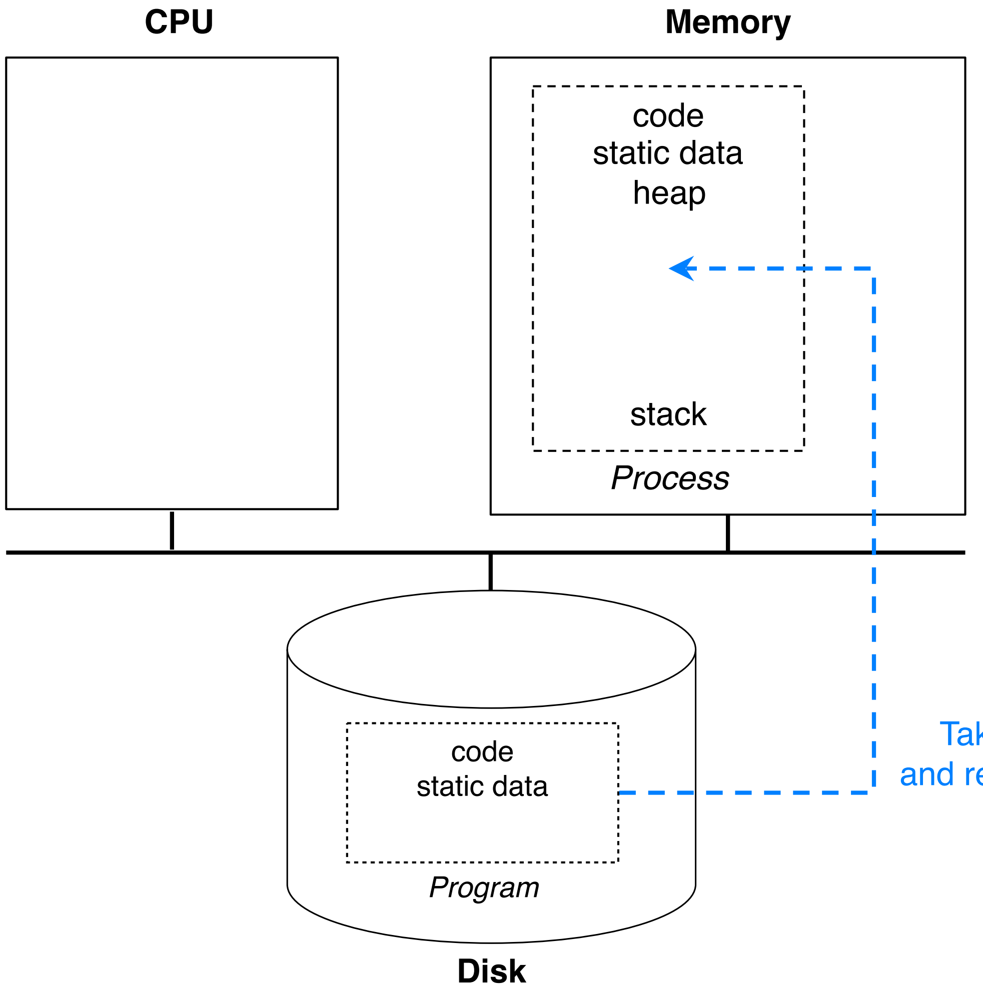
# Process Creation

- **Load** a program code into **memory**, into the address space of the process.
    - Programs initially reside on disk in *executable format*.
    - OS perform the loading process lazily.
        - Loading pieces of code or data only as they are needed during program execution.
- The program's run-time **stack** is allocated.
    - Use the stack for *local variables*, *function parameters*, and *return address*.
    - Initialize the stack with arguments → `argc` and the `argv` array of `main()` function.

# Process Creation (Cont.)

- The program's **heap** is created.
    - Used for explicitly requested dynamically allocated data.
    - Program request such space by calling `malloc()` and free it by calling `free()`.
- The OS do some other initialization tasks.
    - input/output (I/O) setup
        - Each process by default has three open file descriptors.
        - Standard input, output and error
- **Start the program** running at the entry point, namely `main()`.
    - The OS transfers control of the CPU to the newly-created process.

# Loading: From Program To Process



**Loading:**
Takes on-disk program
and reads it into the address
space of process

# Virtualizing the CPU

- Goal: Give each process impression it alone is actively using CPU
- Resources can be shared in **time** and **space**
- Assume single uniprocessor
  - Time-sharing (multi-processors: advanced issue)
- Memory?
  - Space-sharing (later)
- Disk?
  - Space-sharing (later)

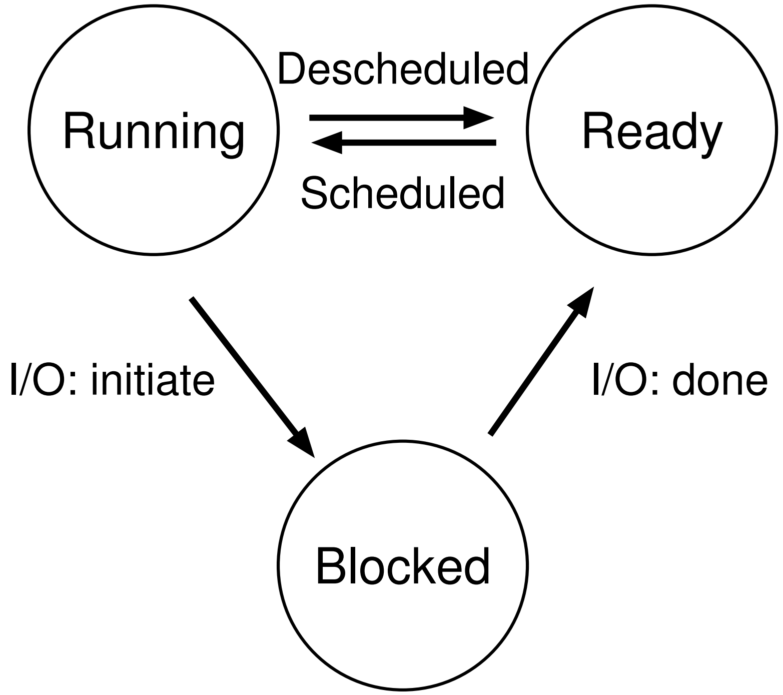
# Provide Good CPU Performance?

- **Direct execution**
  - Allow user process to run directly on hardware
  - OS creates process and transfers control to starting point (i.e., main())
- Problems with direct execution?
  - Process could do something restricted
    - Could read/write other process data (disk or memory)
  - Process could run forever (slow, buggy, or malicious)
    - OS needs to be able to switch between processes
  - Process could do something slow (like I/O)
    - OS wants to use resources efficiently and switch CPU to other process
- Solution: **Limited direct execution** – OS and hardware maintain some control (-> later)

# Process States

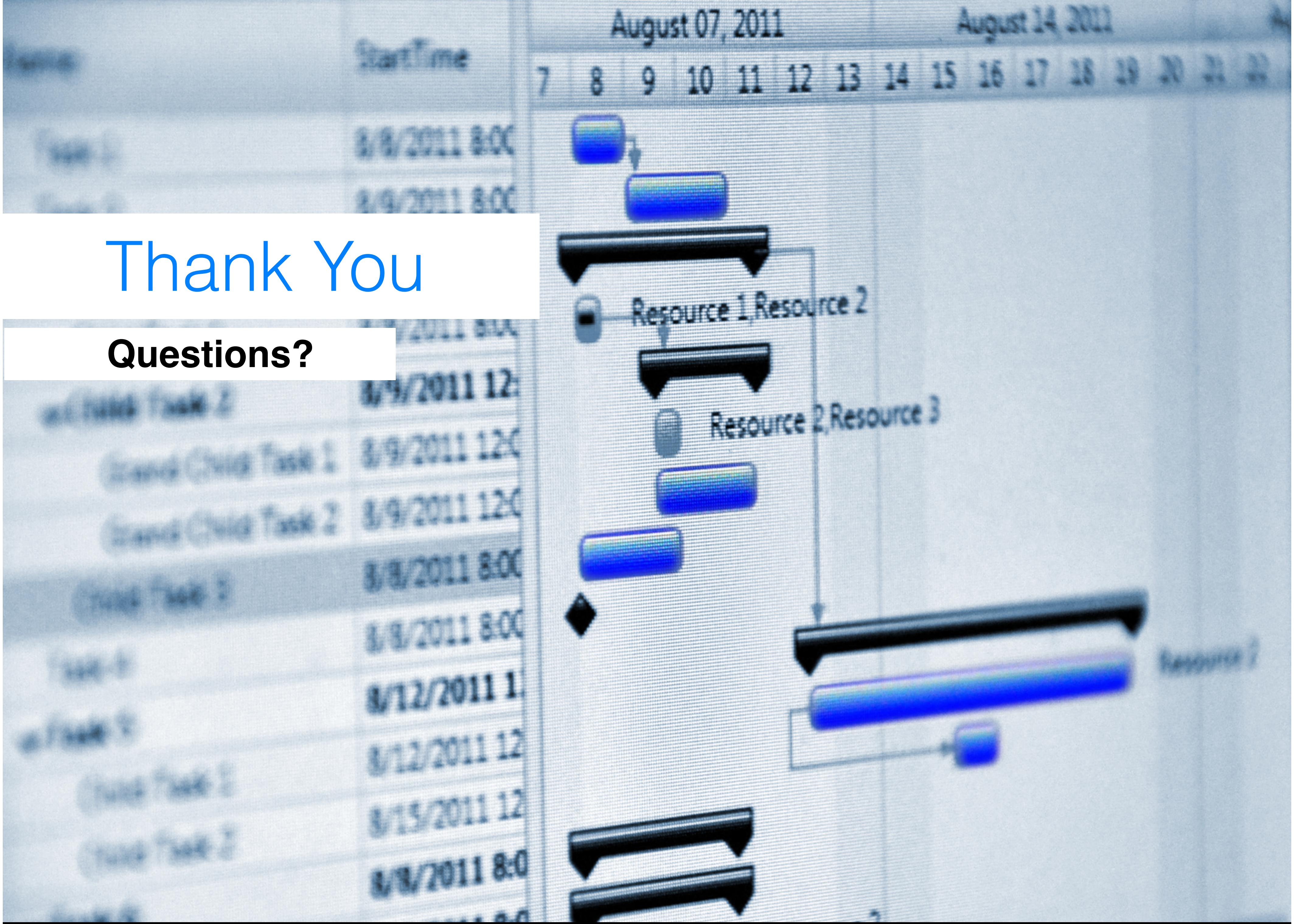
- A process can be one of three states.
  - **Running**
    - A process is running on a processor.
  - **Ready**
    - A process is ready to run but for some reason the OS has chosen not to run it at this given moment.
  - **Blocked**
    - A process has performed some kind of operation.
    - When a process initiates an I/O request to a disk, it becomes blocked and thus some other process can use the processor.

# Process State Transition



# Data structures

- The OS has some key data structures that track various relevant pieces of information.
  - **Process list**
    - Ready processes
    - Blocked processes
    - Current running process
  - **Register context**
- PCB(Process Control Block)
  - A C-structure that contains information about each process.



# Thank You

## Questions?