NTNU | Norwegian University of Science and Technology

Operating Systems

Lecture overview and Q&A Session 8 – 7.3.2022

Michael Engel

Lectures 13 and 14

Real-time scheduling (not relevant for the exam!)

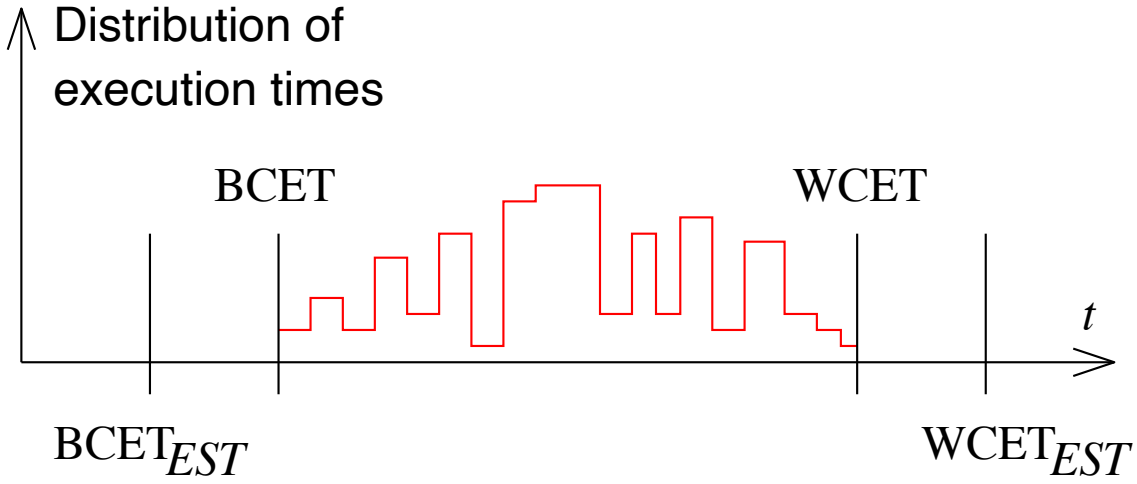
- Real-time problems, deadlines and scheduling
- Rate-monotonic (RM) scheduling
- Earliest deadline first (EDF) scheduling

I/O management and disk scheduling

- I/O device interfacing: buses, interrupts, DMA, addresses
- Polling vs. interrupt-driven I/O
- Device drivers and I/O system layers
- Unix device abstractions and device classes
- Buffering and I/O scheduling

Real-time problems, deadlines and scheduling

- Deadline classification:
  - **soft**: the obtained result (the reaction of the system) is useful even if it was obtained after the deadline has passed
  - **firm**: the result is useless after the deadline has passed
  - **hard**: if the deadline passes without a system reaction, damage can occur
- Analysis: Worst-Case Execution Time (WCET)



The estimated WCET$_{EST}$ has to be ***guaranteed*** larger or equal to the real WCET.

However, the difference between the two should be as small as possible ("*tight bounds*")

Rate-monotonic (RM) scheduling

A1. All tasks are *preemptible* at any time
A2. Only *compute time* is a relevant resource
A3. All tasks are *independent*
A4. All tasks are *periodic*
A5. The *relative deadline* of a task is equal to its period

- ***Necessary*** condition for schedulability: the ***utilization U*** of the system is less than or equal to 1:

$$U = \sum_{i=1}^{m} \frac{C_i}{T_i} \leq 1$$

Assumption: Uniprocessor

U: system load
m: number of tasks

- ***no deadline violations if the following condition holds:***

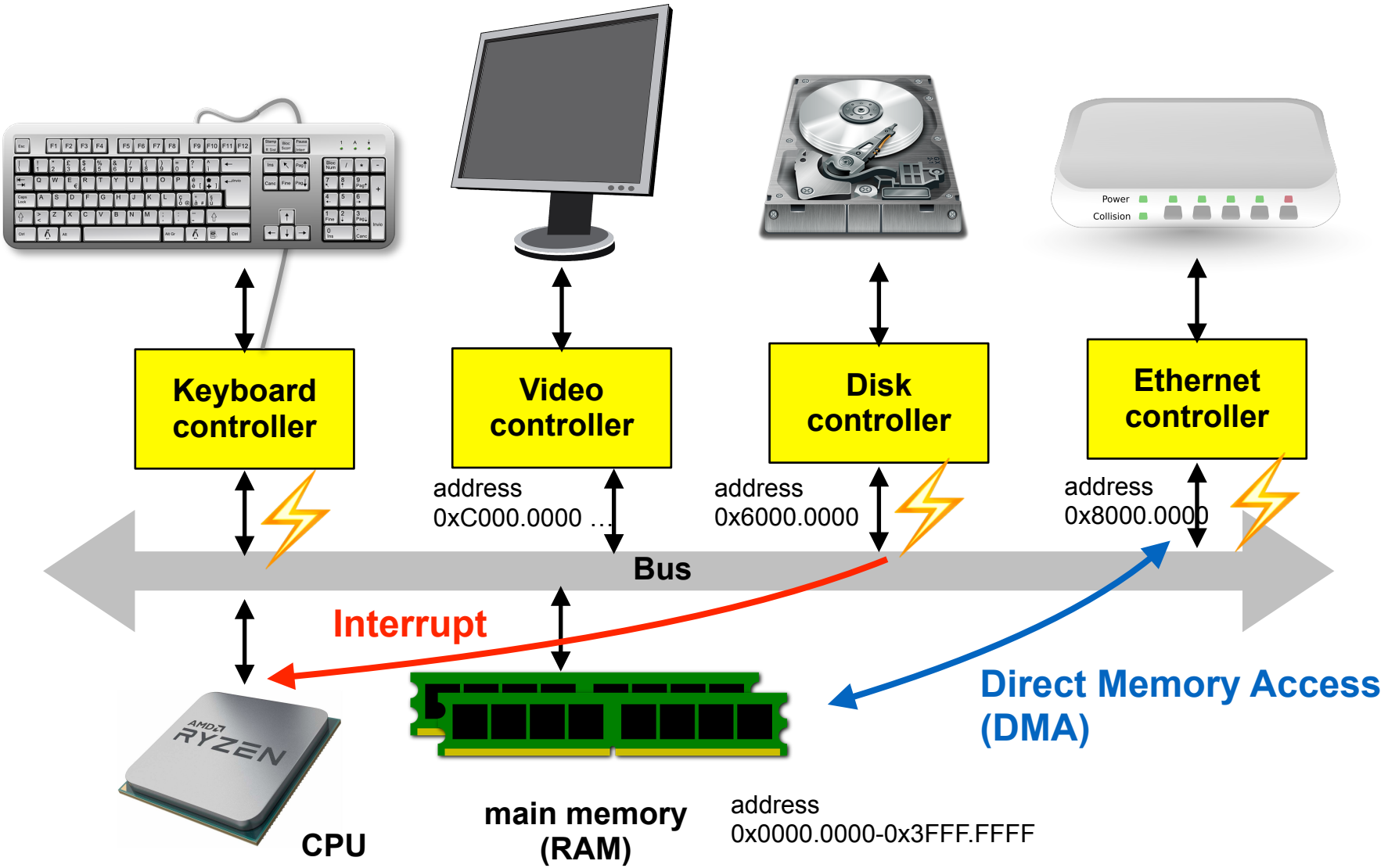
$$U \leq m \cdot \left(2^{1/m} - 1\right)$$

for m → ∞: U <= 0.7

Earliest deadline first (EDF) scheduling

- Tasks which are ready are sorted in order of their ***absolute*** deadlines (specified as relative times)
- If the first task in the list has an earlier deadline than the currently running task, the running task is ***preempted*** immediately!
- If a schedule exists which is able to keep all deadlines, then EDF also keeps all deadlines → **EDF is optimal**
  - …for independent tasks with dynamic priorities!
- Especially for **periodic** tasks the following holds:
  If $U \leq 1$, then EDF always finds a valid schedule
  (without missing deadlines!)

I/O device interfacing



Polling vs. interrupt-driven I/O

Polling: active waiting in a loop

```
/* Copy character into kernel buffer p */
copy_from_user (buffer, p, count);
/* Loop over all characters */
for (i=0; i<count; i++) {
  /* Wait “actively” until printer is ready */
  while (*printer_status_reg != READY);
  /* Print one character */
  *printer_data_reg = p[i];
}
return_to_user ();
```

Interrupts: asynchronous notification

Interrupt handler communicates with I/O device

```
if (count > 0) {
 *printer_data_reg = p[i];
  count--;
  i++;
} else {
  unblock_user ();
}
acknowledge_interrupt ();
return_from_interrupt ();
```

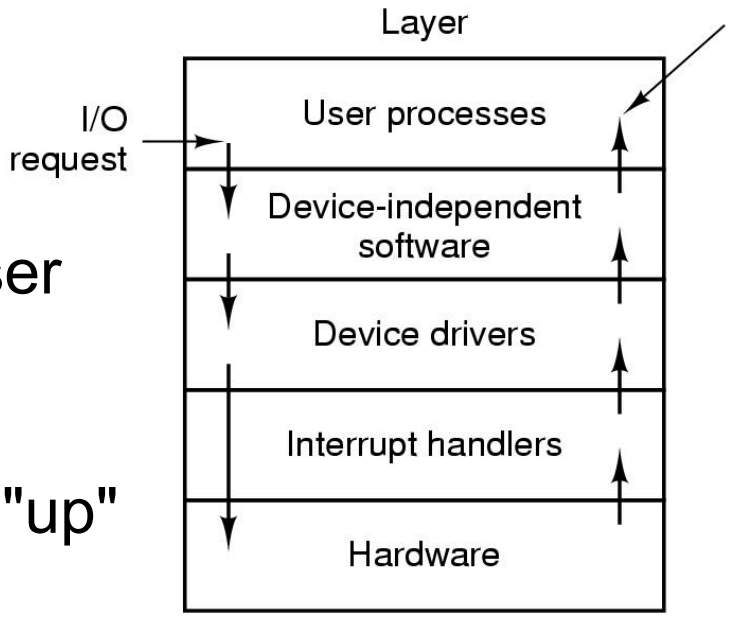
Device drivers and I/O system layers

Drivers

- Part of the OS that serves as a unified interface for *classes* of devices
- Allow applications to be written independent of specific devices

I/O system layers

- I/O requests travel "down" from user mode via kernel (device driver) to the hardware
- Responses (e.g. read data) travel "up" in the other direction



Unix device abstractions

Unix devices

- Peripheral devices are realized as ***special files***
  - Devices can be accessed using read and write operations in the same way as regular files
  - Opening special files creates a connection to the respective device provided by the **device driver**
  - Direct access to the driver by the user

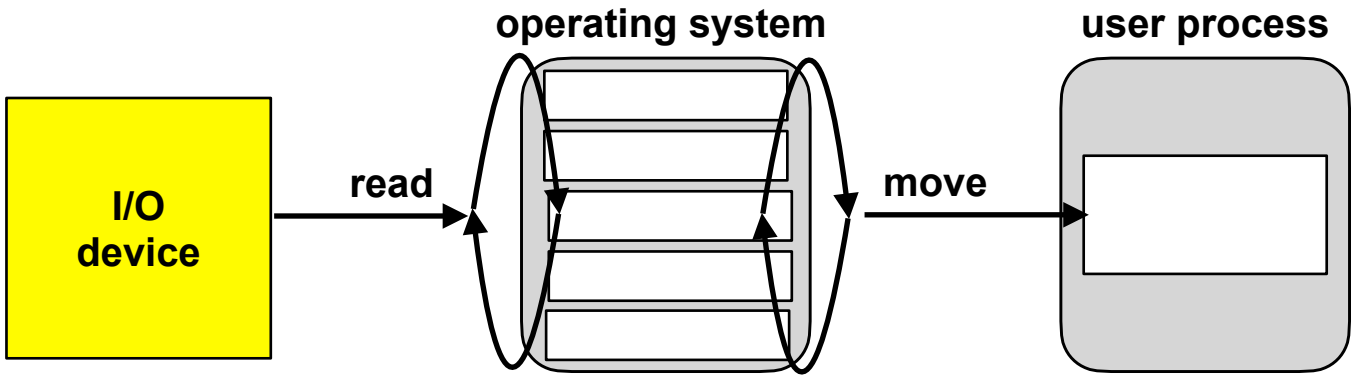
Classes

- **Block oriented special files** (block devices)
  - Disk drives, tape drives, floppy disks, CD-ROMs
- **Character oriented special files** (character devices)
  - Serial interfaces, printers, audio channels etc.

Buffering and I/O scheduling

Buffering

- Single/double/***ring buffering***



I/O scheduling

- Consider ***mechanical properties*** of devices (e.g. hard disks)
  - seek and rotational delay
  - **Optimization criterium:** positioning time
  - Examples: FIFO, SSTF, Elevator

Overview Theoretical Exercise 6

All about real-time scheduling and I/O

Why?

- Real-time is important if you want to work with ***embedded systems***
- I/O performance is highly relevant for all computer use cases
  - scheduling is not only for processes, but also for I/O