Real-Time Operating Systems

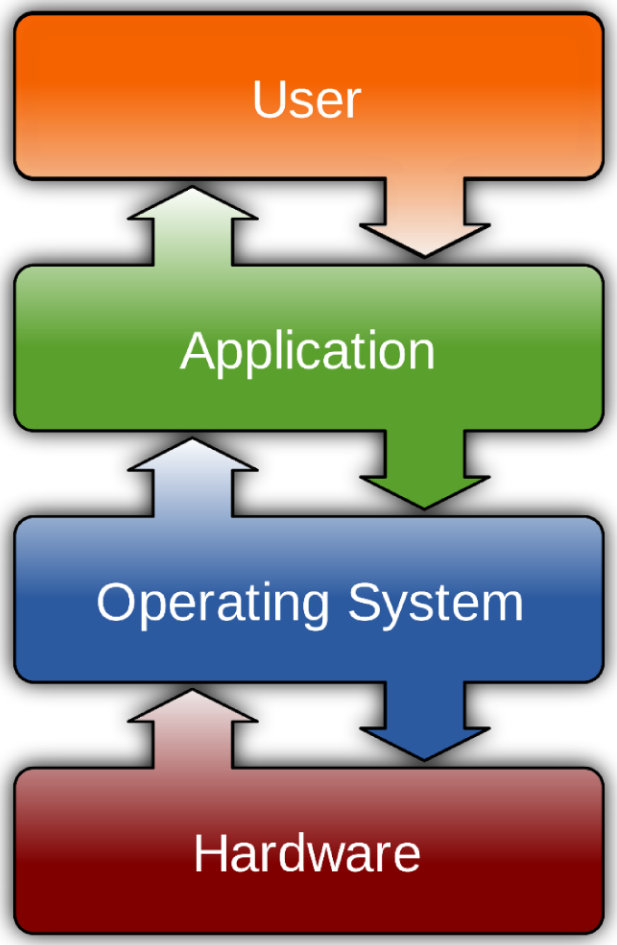
Networks and Embedded Software

Module 6.1

by Wolfgang Neff

Operating Systems (1)

- Allow applications and users to access the hard-ware
- They manage
  - Hardware
  - Resources
- They provide
  - Services

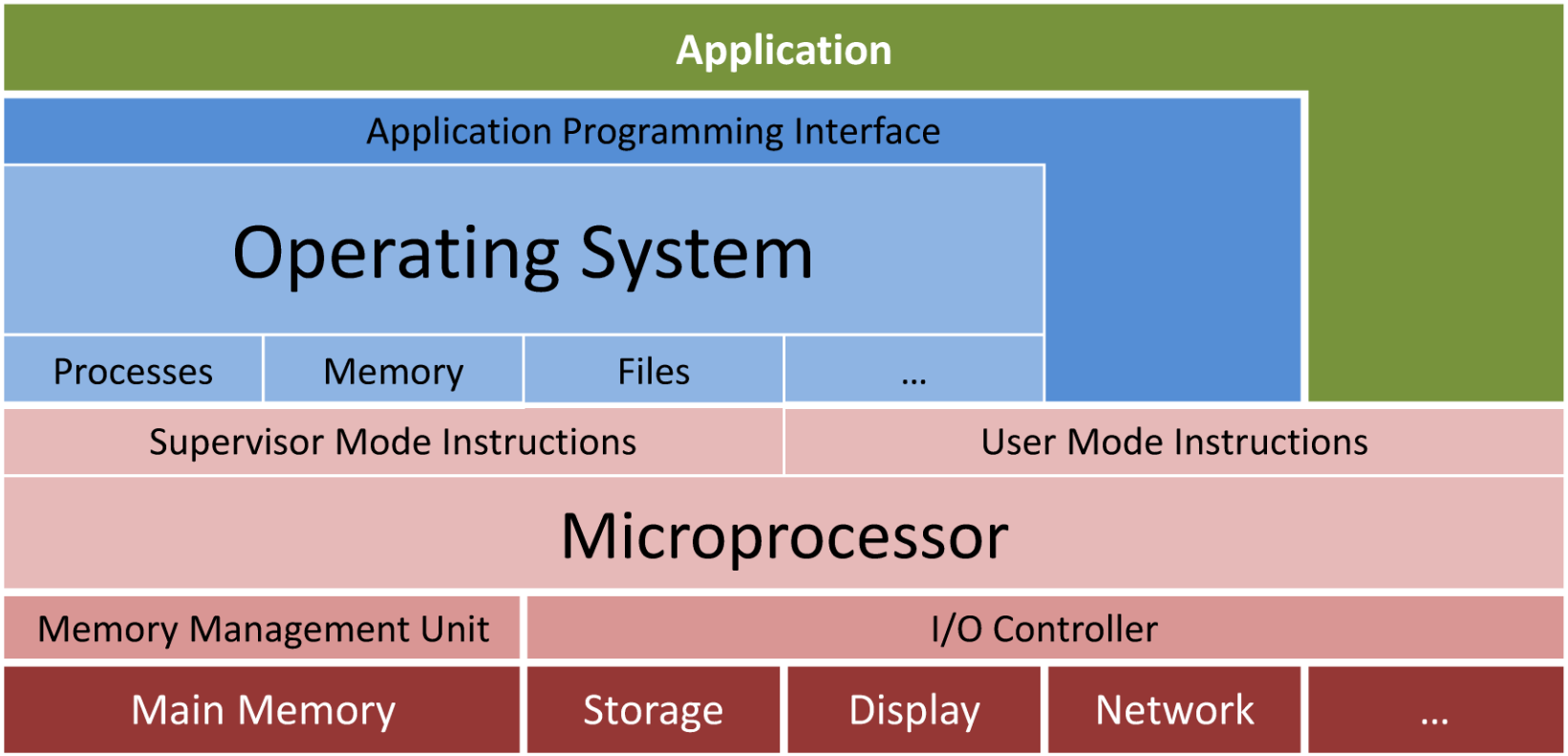


Operating Systems (2)

- Tasks of an Operating System
  - Program execution
    - Management of processes
    - Multitasking
  - Device Control
  - Security
  - Resource management
    - Main memory
    - I/O devices

Operating Systems (3)

- Detailed view



Operating Systems (4)

- Real Time Operating System
  - Process data as they come
  - Response within deadlines
    - Guaranteed maximum response time
  - Used in mission critical contexts
    - Failure endangers the whole mission
  - Classification
    - Hard RTOS: missed deadlines are failures
    - Soft RTOS: missed deadlines degrade quality of service

Bare Metal (1)

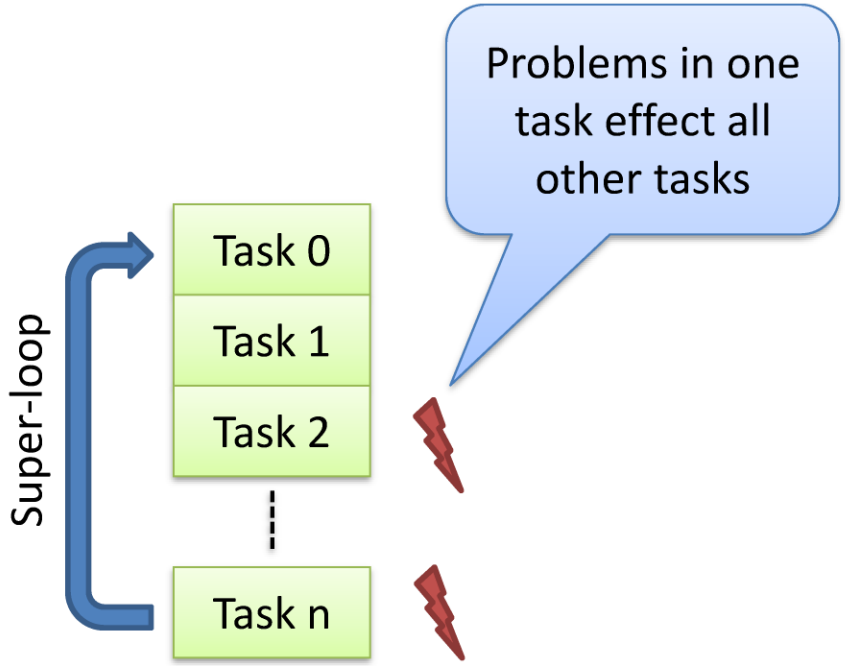
- Systems without operating system
- One non-terminating loop in main
- Two levels
  - Task level
  - Interrupt level

```
ISR(…)
{
}

int main(void)
{
  while (1) {
    task0();
    task1();
    …
    taskN();
  }
}
```

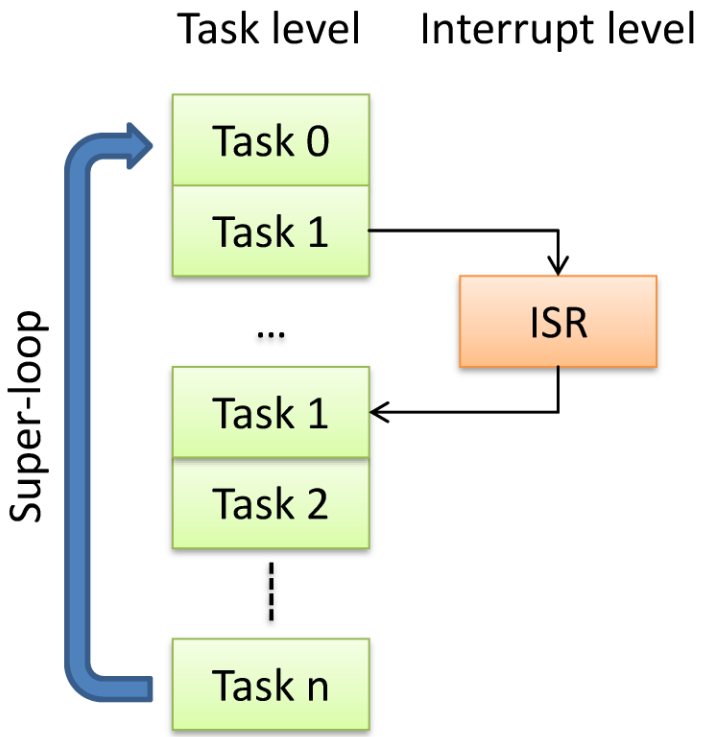
Bare Metal (2)

- Super-loops
  - One loop handles all tasks
  - Advantage
    - Simple structure
  - Drawback
    - Tasks are sequential
    - No priorities



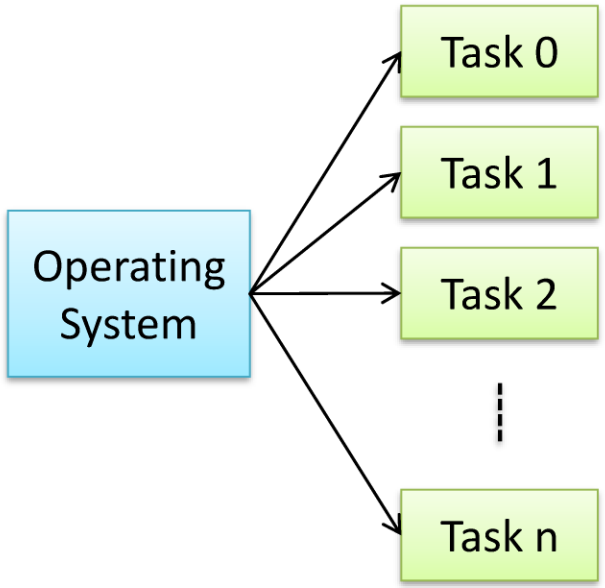
Bare Metal (3)

- Interrupt level
  - Asynchronous tasks handled by interrupts
  - Critical tasks handled by interrupts
  - Advantage
    - Simple structure
  - Drawback
    - ISRs may become too long



RTOS (1)

- Real Time Operating System
  - Create and kills tasks
  - Schedules tasks
  - Provides run time
  - Controls task states
  - Controls resources
  - Manages communication



RTOS (2)

- FreeRTOS example
  - Structure of the main program
    - ```
      #include <FreeRTOS.h>
      #include <task.h>

      int main(void)
      {
         // Setup hardware

        // Create Tasks
         xTaskCreate(vTask,"task", STACK_SIZE, NULL, TASK_PRIORITY, NULL);

         // Start scheduler
         vTaskStartScheduler();
      }
      ```
      ```

RTOS (3)

- FreeRTOS example (continued)
  - Task relevant constants
    - configMINIMAL_STACK_SIZE
    - tskIDLE_PRIORITY
  - Structure of a task
    - ```
      void vTask(void* pvParameters)
      {
        // Task initialization goes here
        while(1) {
          // Task code goes here
        }
      }
      ```
      ```