

## Course

# BVS1 - Operating Systems and Distributed Systems 1

---

Version: 3 | Last Change: 01.04.2022 09:35 | Draft: 0 | Status: vom verantwortlichen Dozent freigegeben

### ^ General information

<b>Long name</b>	Operating Systems and Distributed Systems 1
<b>Approving CModule</b>	<a href="#">BVS1_BaET</a> , <a href="#">BVS1_BaTIN</a>
<b>Responsible</b>	Prof. Dr. Cartsten Vogt Professor Fakultät IME
<b>Level</b>	Bachelor
<b>Semester in the year</b>	winter semester
<b>Duration</b>	Semester
<b>Hours in self-study</b>	60
<b>ECTS</b>	5
<b>Professors</b>	Prof. Dr. Cartsten Vogt Professor Fakultät IME
<b>Requirements</b>	procedural programming architecture of a digital computer (basic knowledge) Internet protocols (basic knowledge)
<b>Language</b>	German
<b>Separate final exam</b>	Yes

## Final exam

### Details

Students shall prove that they can 1.) explain and apply fundamental terms, concepts, and techniques, 2.) apply programming and more abstract concepts to solve application problems in the field of concurrent and distributed programming and 3.) assess the correctness of statements and program code. Typical types of assignments are 1.) multiple choice questions, fill-in-the-blank texts, assessment of statements, 2.) write program code or develop a solution in a more abstract form to solve given problems of limited size and 3.) finding errors in texts and program code.

### Minimum standard

At least 50% of the total number of points.

## Exam Type

Students shall prove that they can 1.) explain and apply fundamental terms, concepts, and techniques, 2.) apply programming and more abstract concepts to solve application problems in the field of concurrent and distributed programming and 3.) assess the correctness of statements and program code. Typical types of assignments are 1.) multiple choice questions, fill-in-the-blank texts, assessment of statements, 2.) write program code or develop a solution in a more abstract form to solve given problems of limited size and 3.) finding errors in texts and program code.

## ^ Lecture / Exercises

### Learning goals

---

#### Knowledge

fundamentals of operating systems and distributed systems  
position and tasks of an operating system in a computer  
resources to be managed  
concurrency in hard- and software  
components and properties of distributed systems  
software structures  
operating system kernel  
hierarchical structures  
virtual machines  
client-server systems  
peer-to-peer systems

---

the UNIX/Linux operating system  
history and standards  
layered structure  
kernel with programming interface  
shell with user interface  
fundamental user commands  
structure of the file system  
programming in C

---

concurrency  
processes and threads  
fundamental properties  
processes in UNIX  
threads in Java  
synchronization  
fundamental conditions  
mutual exclusion  
sequencing  
mechanisms  
interrupt masking  
spinlocks  
signals  
semaphores

monitors  
deadlocks

---

communication  
fundamental terms  
storage-based vs. message-based communication  
mailboxes and ports  
synchronous vs. asynchronous communication  
local communication  
shared memory  
message queues  
pipes  
communication in distributed systems  
protocols  
sockets

---

## Skills

using the interfaces of an operating system:

user interface (console)  
programming interface (API)

---

controlling concurrent operations in an operating system  
from the user interface  
by API functions

---

synchronizing concurrent operations by synchronization mechanisms

---

using various communication mechanisms  
local mechanisms  
mechanisms in computer networks

## Expenditure classroom teaching

Type	Attendance (h/Wk.)
Lecture	2
Exercises (whole course)	1
Exercises (shared course)	1
Tutorial (voluntary)	0

## Separate exam

none

## ^ Practical training

### Learning goals

---

#### Knowledge

commands of the character-based Linux/UNIX command interface  
usage at the console  
usage in shell scripts  
esp. to control concurrent processes

---

C functions of the UNIX/Linux programming interface  
to access files and devices  
to start and control processes  
to synchronize processes  
to transfer data between processes (locally and in a network) - depending on available time

---

#### Skills

application of the aspects listed above to real-world scenarios in small teams

### Expenditure classroom teaching

Type	Attendance (h/Wk.)
Practical training	1
Tutorial (voluntary)	0

### Separate exam

#### Exam Type

working on practical scenarion (e.g. in a lab)

#### Details

Students work in small teams. Each team completes multiple "rounds" with assigned appointments in the lab. In each round, programming assignments are solved.

For the preparation of a laboratory appointment a "preparation sheet" has to be solved. The acquired knowledge will be tested at the beginning of the appointment (short written entrance test, interview with the supervisor). In case of failure, a follow-up appointment must be taken; in case of multiple failures, the student will be excluded from the lab. In case of success, a "laboratory work sheet" with further tasks will be worked on under supervision (and, if necessary, with assistance).

#### Minimum standard

Successful participation in all laboratory appointments, i.e. in particular independent solution (or with some assistance if necessary) of the programming assignments.

