

操作系统研讨课

Course: B0911011Y

陈明宇、蒋德钧

Fall Term 2021-2022

email: cmy@ict.ac.cn / jiangdejun@ict.ac.cn

office phone: 62601007



Lecture 2 A Simple Kernel (Part I)

2021.09.18



Schedule

- Project 1 due
- Project 2 assignment



Project 1 Due

- P1 due
 - Answer any questions or make any code changes we may ask



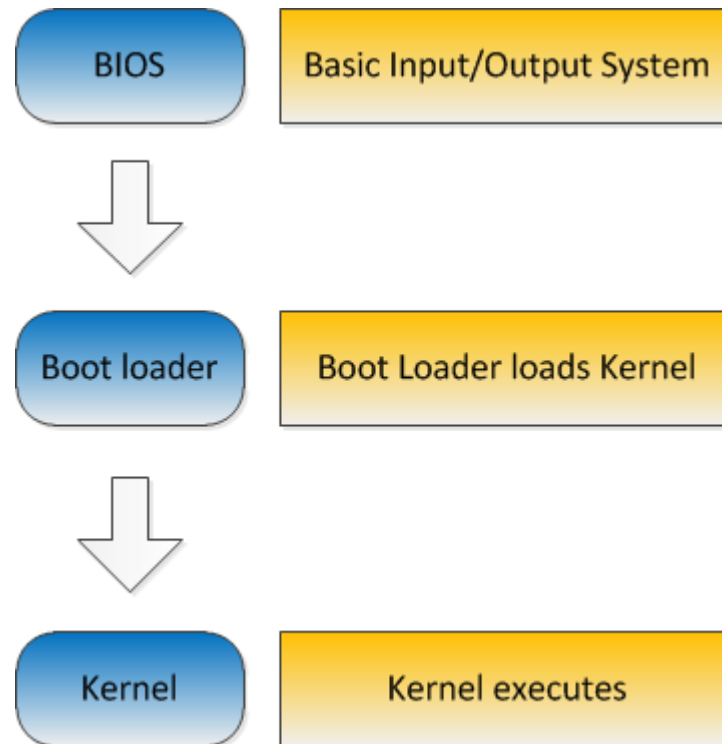
Project 1 Due

- P1 submission
 - Commit the following materials to gitlab
 - Source code
 - README to simply describe your code, e.g. which file is your work or how to run your code
 - Document recording lessons learned from debugging
 - Do not forget to submit before Sep. 20th 23:59



Project 2 – A Simple Kernel

- Booting procedure



Project 2 – A Simple Kernel

- Requirements (part I)
 - Write a simple kernel (non-preemptive)
 - Start a set of kernel tasks
 - Perform context switches between tasks
 - Provide non-preemptive kernel support with context switch
 - Support basic mutex to allow BLOCK state of processes/threads



Project 2 – A Simple Kernel

- A set of multiple tasks
 - Program codes under the *test* directory in start-code
 - Please refer to *test.c* for different groups of tasks
 - Fixed number of tasks for each test group
 - Allocate per-task state statically in main.c
 - STRONGLY suggest to first read the codes of different tasks to understand what they do



Project 2 – A Simple Kernel

- Process Control Block (PCB)
 - A data structure in OS kernel containing the information to manage a particular process
 - Normally, kept in an area of memory protected from normal user access



Project 2 – A Simple Kernel

- Process Control Block (PCB)
 - Process status
 - Status of a process when it is suspended
 - Contents of registers, stack pointers etc.
 - Process scheduling info
 - e.g. priority



Project 2 – A Simple Kernel

- Start a task
 - Task type
 - Kernel task
 - Task entry point
 - Function addresses
 - Please refer to *task_info* structure in start-code



Project 2 – A Simple Kernel

- Start a task
 - Each task is associated with a PCB
 - Initialize PCB
 - Which registers should be set?
 - Where is the task located?
 - How to setup stack? Stack size?
 - Where is the PCB located?



Project 2 – A Simple Kernel

- Start a task
 - Possible memory layout
 - Dynamically allocate memory space for task
 - You can implement an `alloc()` which allocates 4K memory space each time

地址范围	建议用途
0x50000000-0x50200000	BBL 代码及其运行所需的内存
0x50200000-0x50500000	Kernel 的数据段/代码段等
0x50500000-0x60000000	供内核动态分配使用的内存

表 P2-1: 地址空间用途划分



Project 2 – A Simple Kernel

- Start a task
 - Scheduler: single task vs. multi-tasks
 - How to organize tasks being scheduled?
 - Use a queue
 - How to select the next task?
 - FIFO



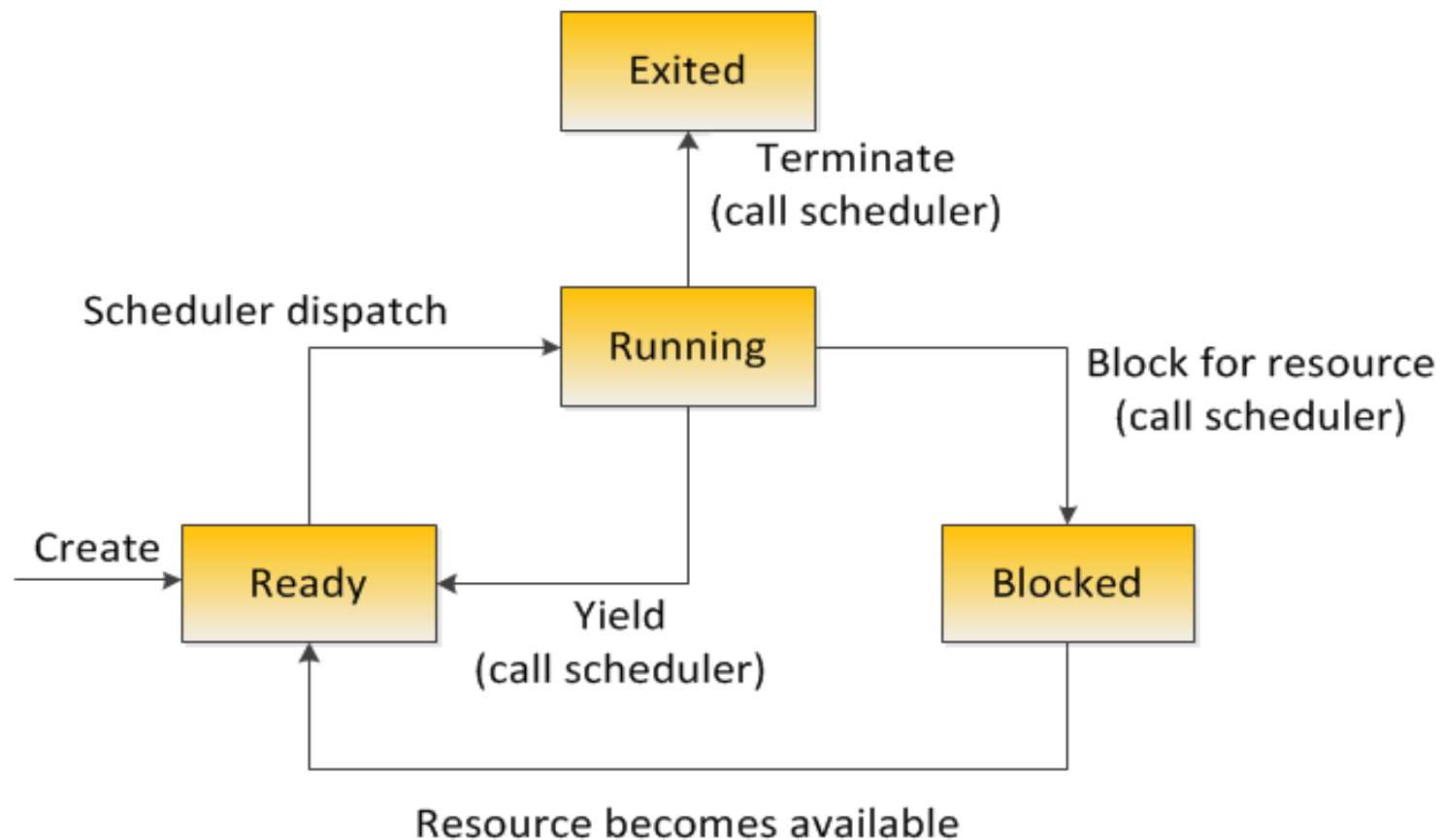
Project 2 – A Simple Kernel

- Start a task
 - Scheduler: first task vs. the following ones
 - do_scheduler()
 - Locate the PCB of the first task
 - Restore PCB



Project 2 – A Simple Kernel

- Scheduler (non-preemptive kernel)



Project 2 – A Simple Kernel

- Yield
 - An action to force a processor to release control of the current running thread
 - Place the current running thread to the end of the running queue
 - In this project, we call `do_scheduler()` to execute yield



Project 2 – A Simple Kernel

- Context switch
 - Save PCB
 - What kind of things to save
 - Registers → Memory
 - Restore PCB
 - Memory → Registers
 - `do_scheduler()`



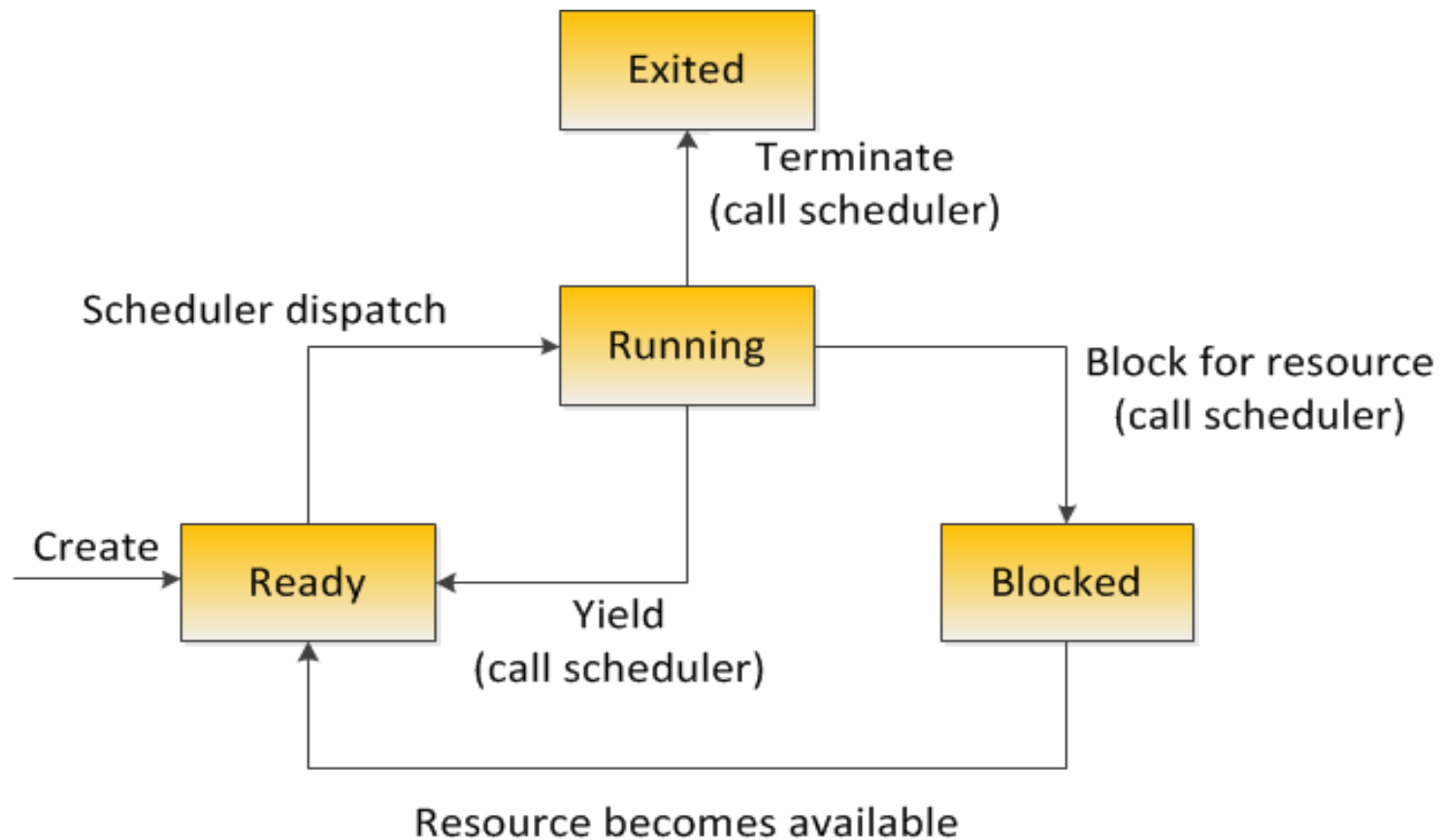
Project 2 – A Simple Kernel

- Create machine image
 - Combine binary of kernel, tasks together into a machine image using createimage in your P1



Project 2 – A Simple Kernel

- Mutex lock



Project 2 – A Simple Kernel

- Mutex lock
 - What if no thread currently holds the lock?
 - Acquire the lock
 - What if the lock is currently held?
 - Wait
 - Implement lock-related functions
 - Manage tasks that do not acquire the lock
 - Same queue vs. different queues?



Project 2 A Simple Kernel

- Step by step
 - Task 0: PLEASE read **Sec. 1 – 4** of the guiding book and start code CAREFULLY
 - Task 1: start a set of kernel threads and support context switch as a non-preemptive kernel
 - Task 2: implement mutex lock to support BLOCK state



Project 2 A Simple Kernel

- Requirement for design review
 - Show the example code of your PCB
 - Provide the workflow or pseudo code for the task initialization
 - When is context switching in this project? Provide the workflow or pseudo code of the context switching?
 - When a task is blocked, how does the kernel handle the blocked task?



Project 2 – A Simple Kernel

- Requirement for S core
 - Start tasks and set PCBs
 - Execute context switch without errors
 - Implement the mutex lock
- No A/C cores for this project part I



Project 2 – A Simple Kernel

- P2 schedule
 - 27th Sep.
 - P2 part I design review
 - 4th Oct.
 - P2 part I due
 - P2 part II assignment
 - 11th Oct
 - P2 part II design review
 - 18th Oct
 - P2 part II due



Project 2 – A Simple Kernel

- Final reminder
 - 重要的事情说三遍
 - Start early
 - Start early
 - Start early

