Introduction to the Use of Computers

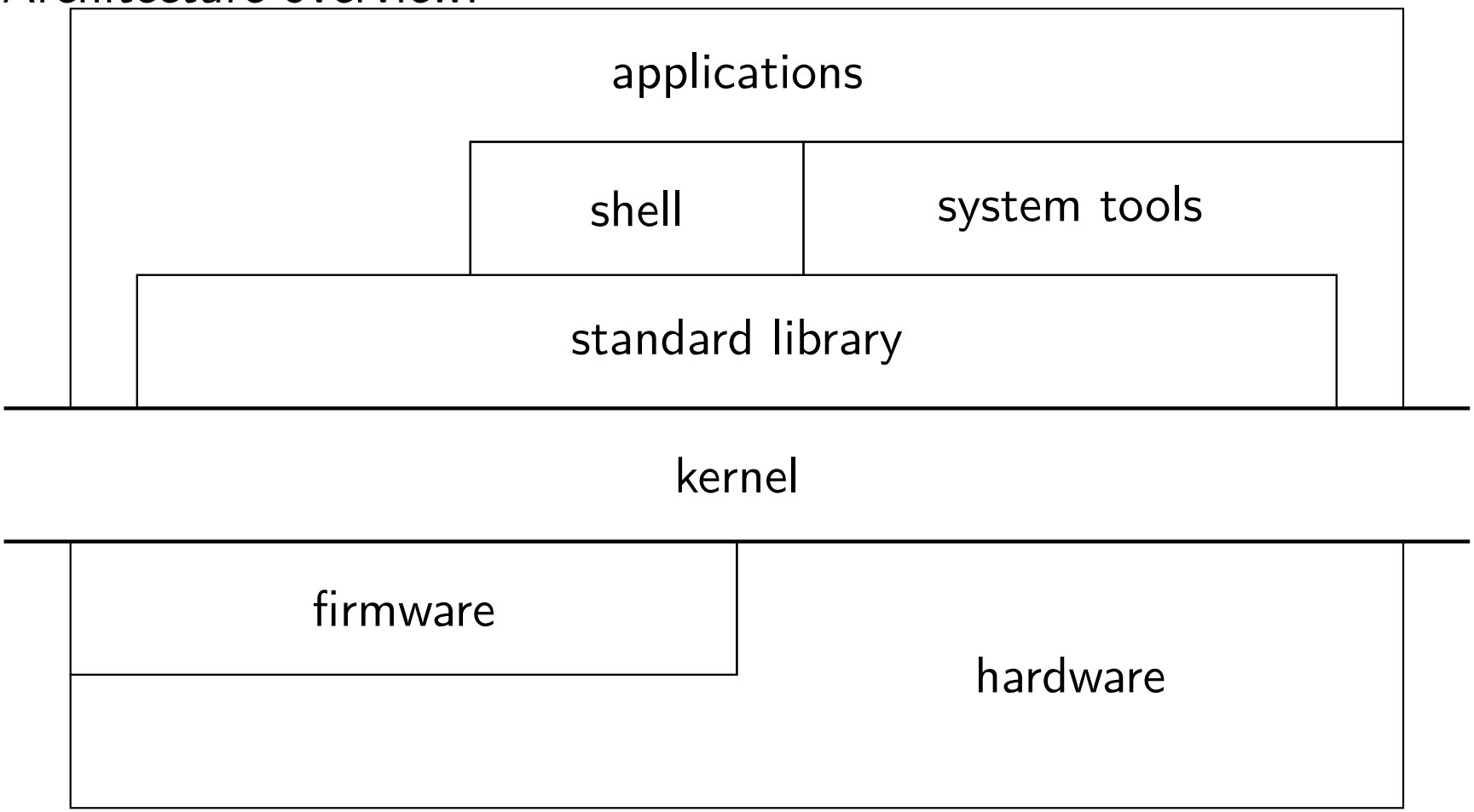
Christophe Rhodes
c.rhodes@gold.ac.uk

Autumn 2012, Fridays: 10:00–12:00: WTA & 15:00–17:00: WHB 300

Operating System

Overview

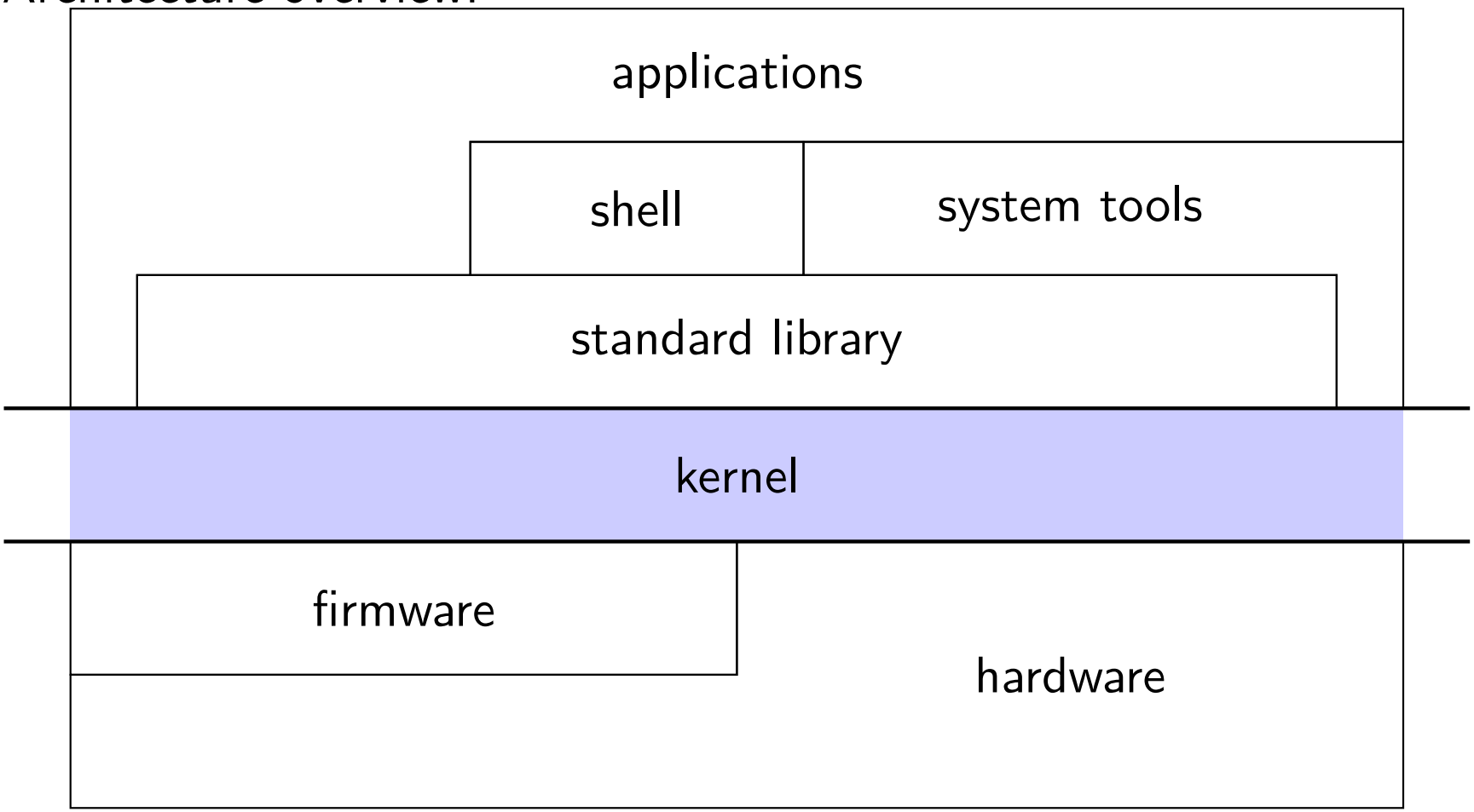
Architecture overview:



Operating System

Overview

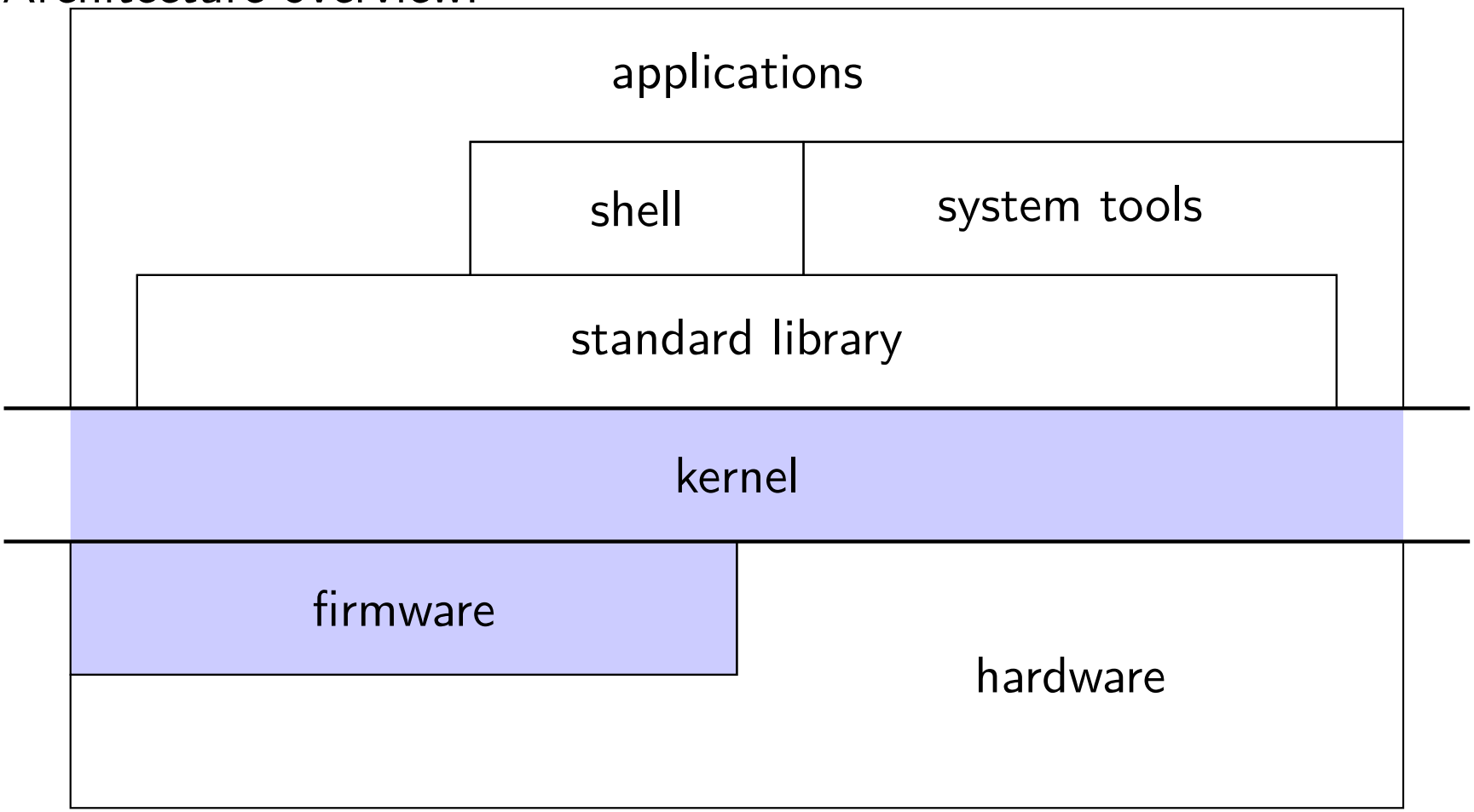
Architecture overview:



Operating System

Overview

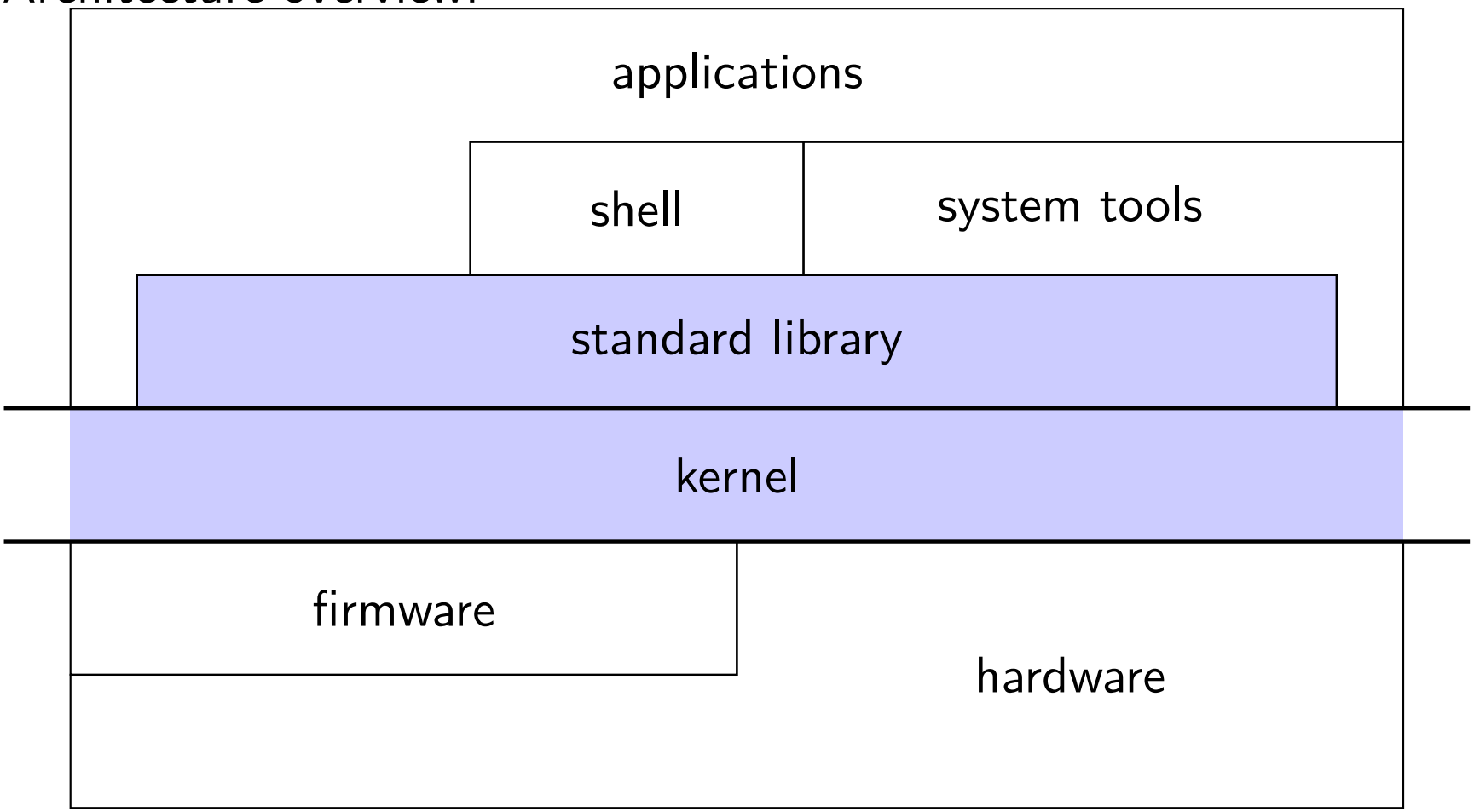
Architecture overview:



Operating System

Overview

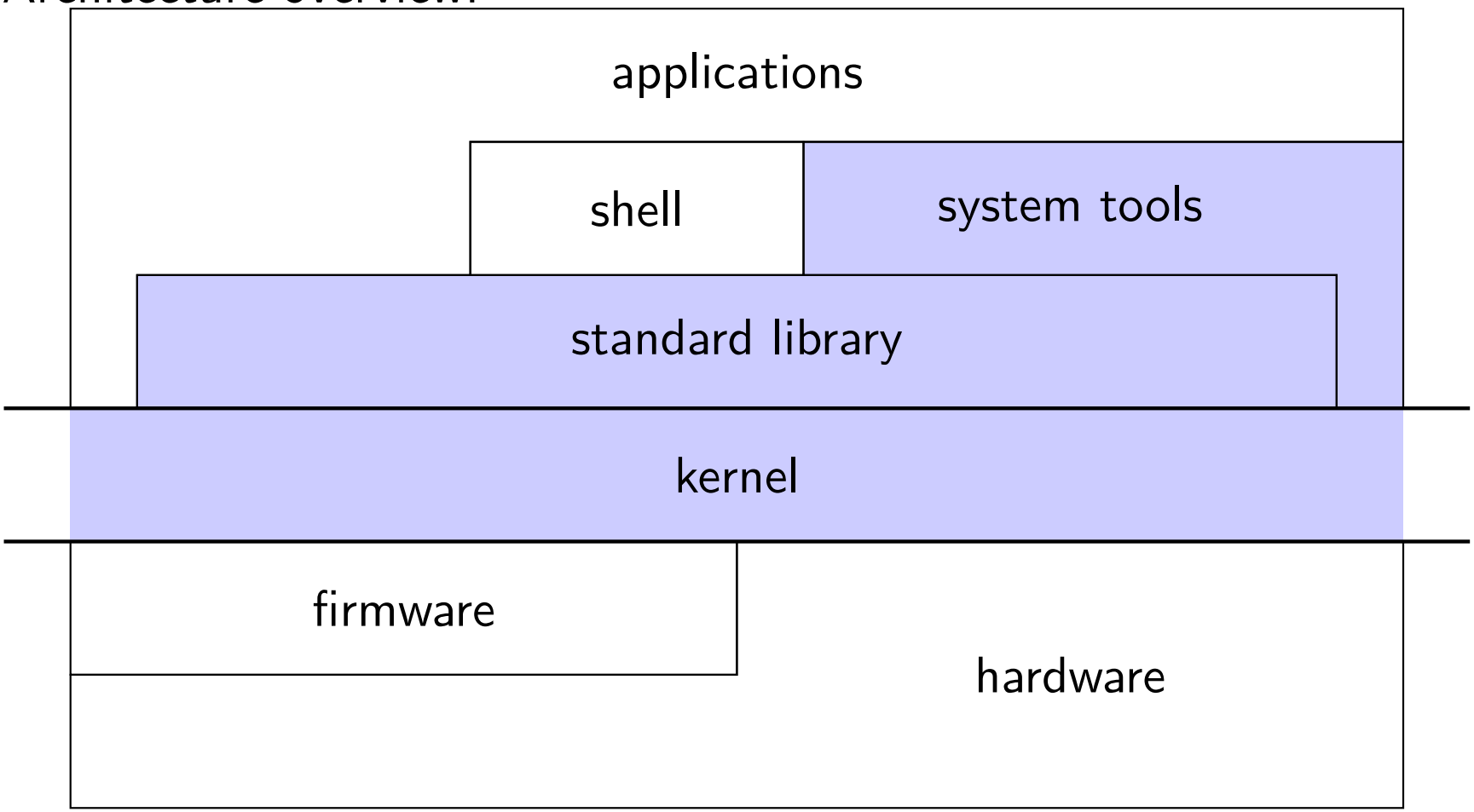
Architecture overview:



Operating System

Overview

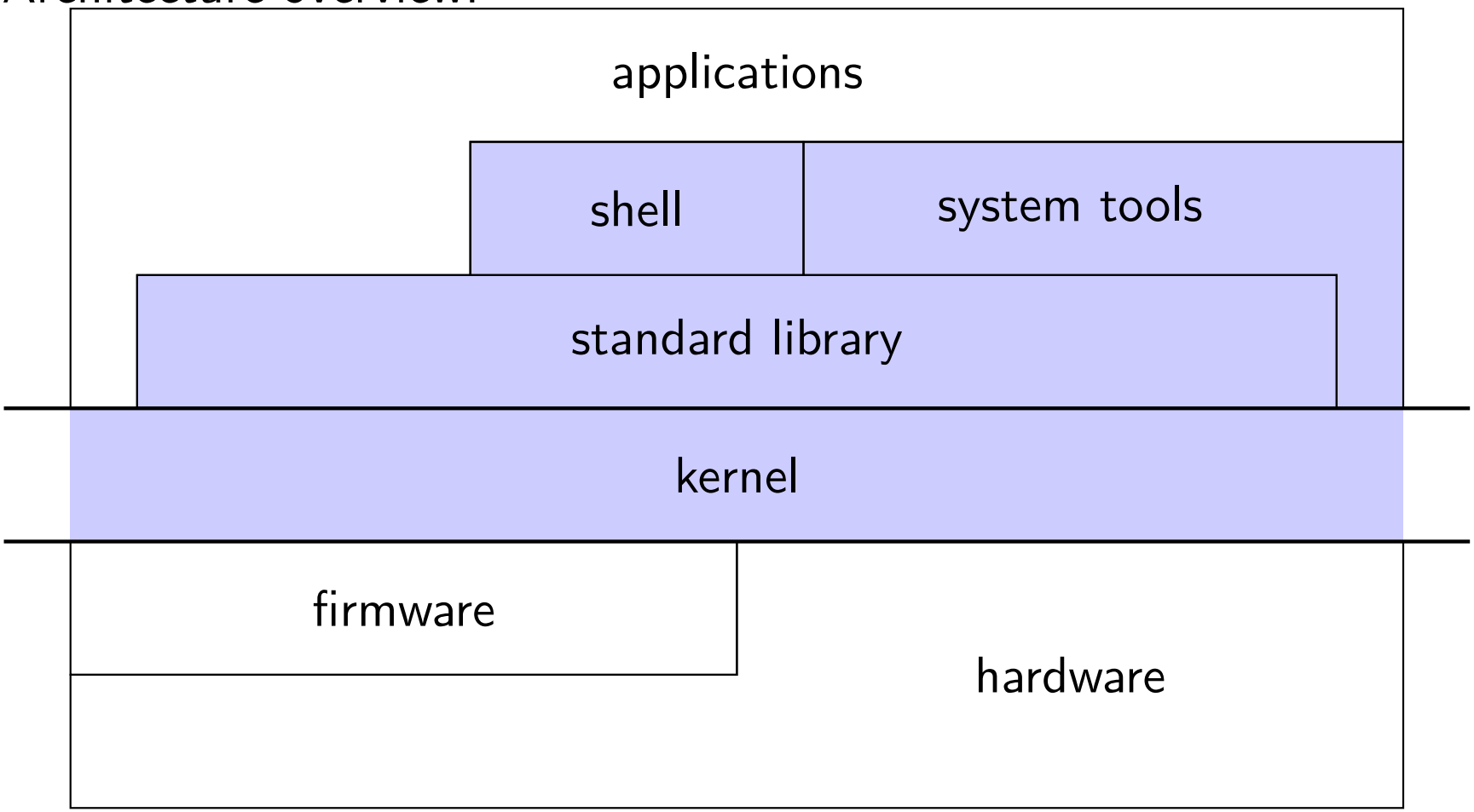
Architecture overview:



Operating System

Overview

Architecture overview:



Operating System

Firmware

Half-way between ‘hard’ware and ‘soft’ware:

- embedded computer program running on bare hardware;
- updatable by software under special circumstances;

Examples:

- Basic Input/Ouptut System (BIOS):
  - PC functionality at startup;
  - variants: OpenFirmware, Extensible Firmware Interface;

Operating System

Firmware

Half-way between ‘hard’ware and ‘soft’ware:

- embedded computer program running on bare hardware;
- updatable by software under special circumstances;

Examples:

- Basic Input/Ouptut System (BIOS):
  - PC functionality at startup;
  - variants: OpenFirmware, Extensible Firmware Interface;
- recording / playback devices:
  - keep up to date with technological and legal changes;

Operating System

Firmware

Half-way between ‘hard’ware and ‘soft’ware:

- embedded computer program running on bare hardware;
- updatable by software under special circumstances;

Examples:

- Basic Input/Ouptut System (BIOS):
  - PC functionality at startup;
  - variants: OpenFirmware, Extensible Firmware Interface;
- recording / playback devices:
  - keep up to date with technological and legal changes;
- wireless network cards:
  - conform to different local regulations on same hardware.

Operating System

Kernel

- provides basic services:
  - open a file;
  - create a directory;
  - connect to a network host.

Operating System

Kernel

- provides basic services:
  - open a file;
  - create a directory;
  - connect to a network host.
- mediates access to hardware:
  - manages bus (and other resource) contention;
  - provides illusion of multiple tasks on a single processor.

Operating System

Kernel

- provides basic services:
  - open a file;
  - create a directory;
  - connect to a network host.
- mediates access to hardware:
  - manages bus (and other resource) contention;
  - provides illusion of multiple tasks on a single processor.

Many special-purpose kernels optimized for particular aspects:

- space (embedded devices: *e.g.* PalmOS);
- reliability (no-access environments: *e.g.* VxWorks);
- real-time response (guidance systems: *e.g.* QNX);
- scalability (mainframes, supercomputers: *e.g.* z/OS).

Operating System

Kernel

Resource Management:

- Device manager:
  - manage bus contention;
  - handle device interrupts;

Operating System

Kernel

Resource Management:

- Device manager:
  - manage bus contention;
  - handle device interrupts;
- Process manager (scheduler):
  - which task should be run next?
  - how long should it be allowed to run?

Operating System

Kernel

Resource Management:

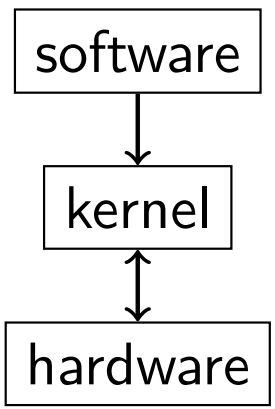
- Device manager:
  - manage bus contention;
  - handle device interrupts;
- Process manager (scheduler):
  - which task should be run next?
  - how long should it be allowed to run?
- Memory manager:
  - maintain association between physical and virtual memory;
  - handle out of memory conditions.

Operating System

Kernel

Three different designs:

- monolithic (*e.g.* Linux);

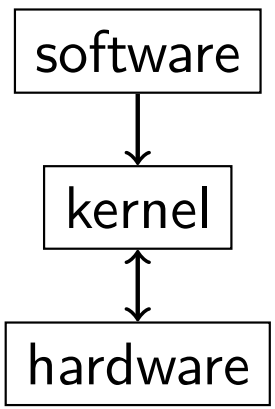


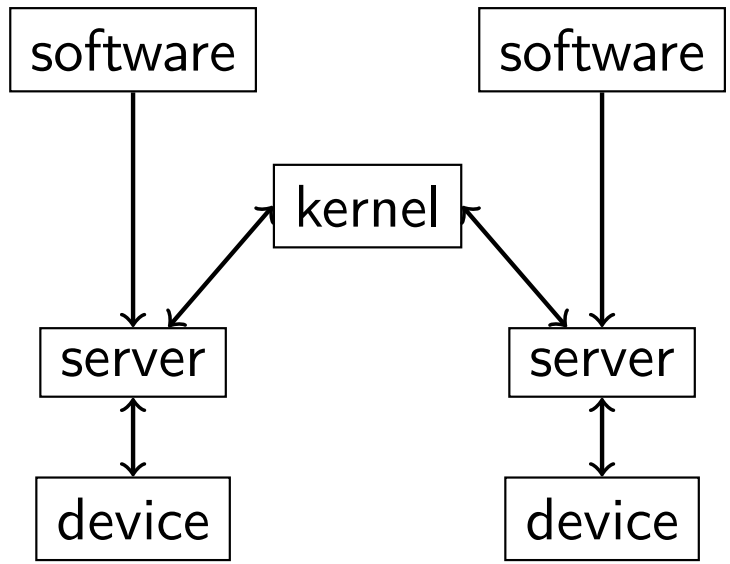
Operating System

Kernel

Three different designs:

- monolithic (*e.g.* Linux);
- microkernel (*e.g.* GNU Hurd);



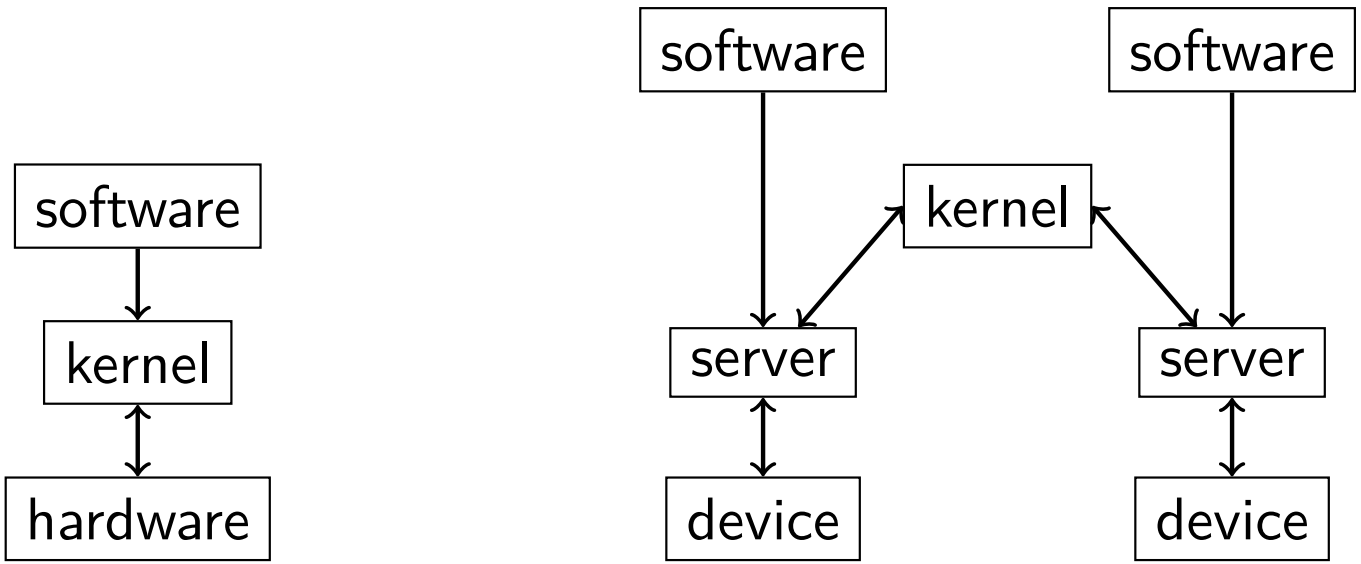


Operating System

Kernel

Three different designs:

- monolithic (*e.g.* Linux);
- microkernel (*e.g.* GNU Hurd);
- hybrid (*e.g.* NT, XNU);

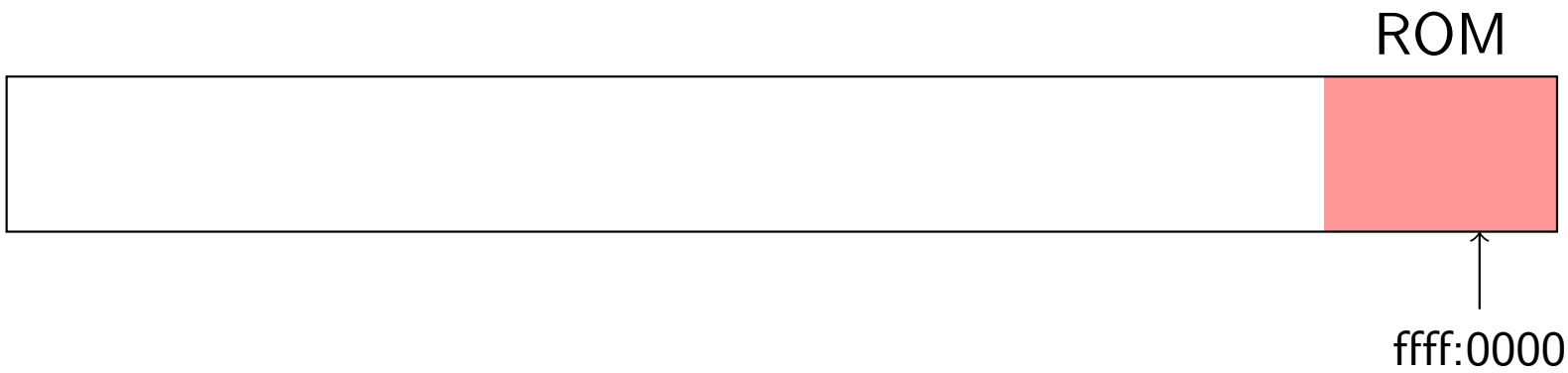


Operating System

Booting

Booting (from *bootstrapping*)

- processors fetch and execute instructions from memory;
- RAM is uninitialized when the computer is turned on...

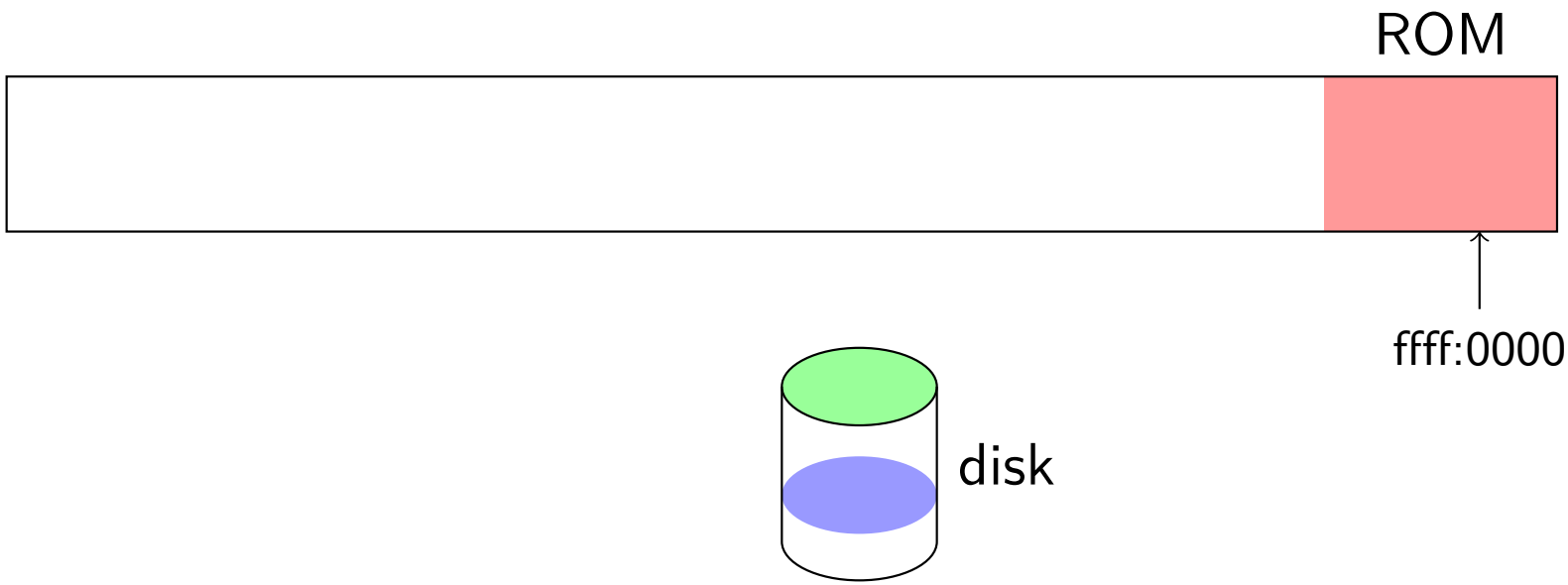


Operating System

Booting

Booting (from *bootstrapping*)

- processors fetch and execute instructions from memory;
- RAM is uninitialized when the computer is turned on...

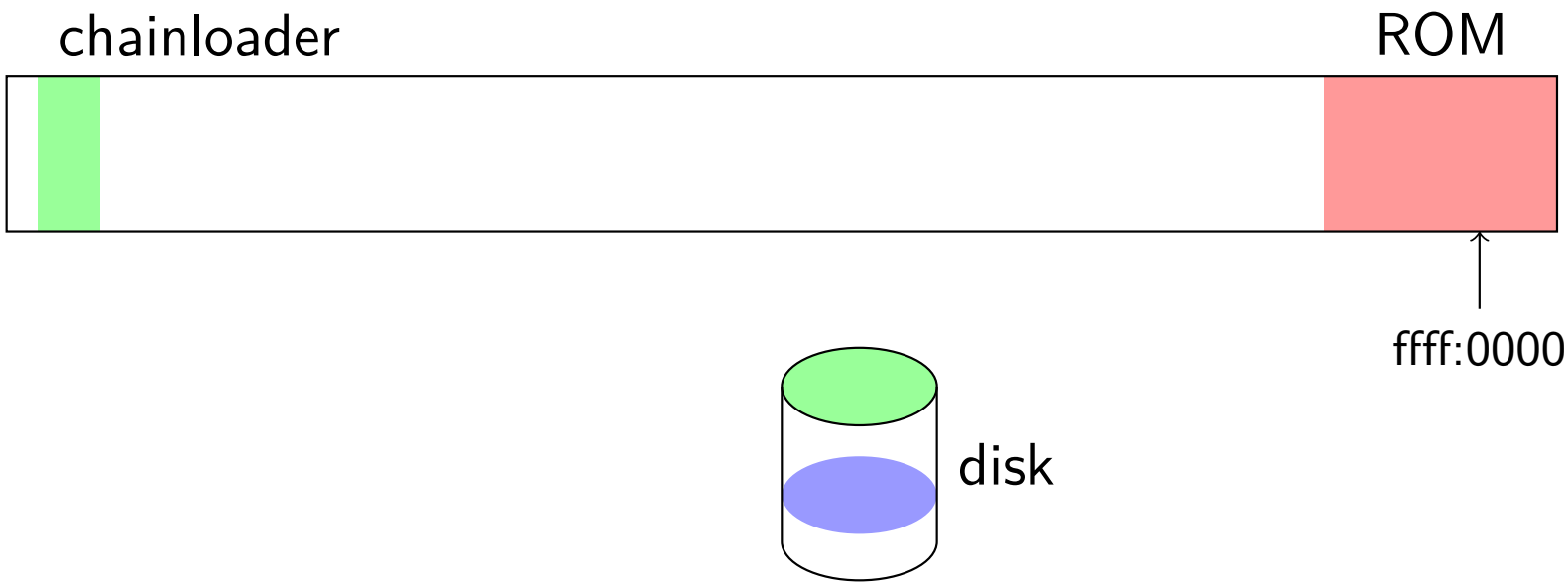


Operating System

Booting

Booting (from *bootstrapping*)

- processors fetch and execute instructions from memory;
- RAM is uninitialized when the computer is turned on...

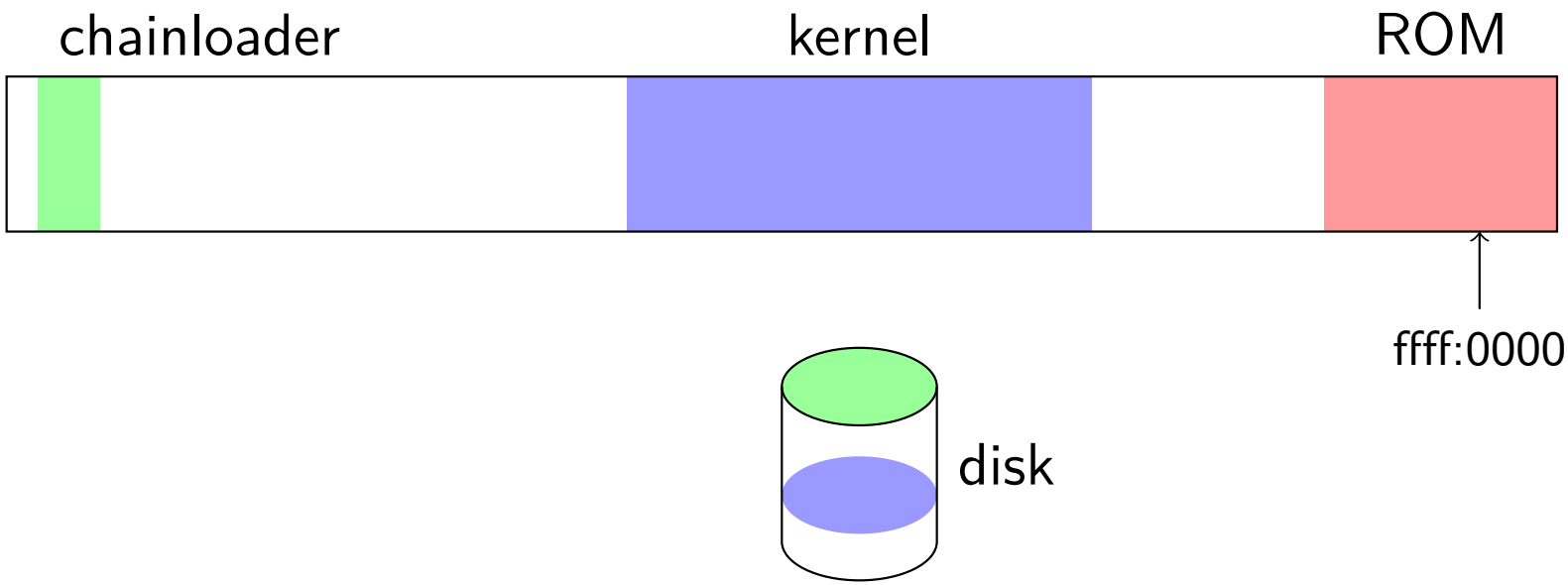


Operating System

Booting

Booting (from *bootstrapping*)

- processors fetch and execute instructions from memory;
- RAM is uninitialized when the computer is turned on...



Operating System

Operating Environment

Components:

- OS kernel;
- System libraries, providing
  - stable programmer interface for kernel services;
  - common functionality for applications;
- one or more ‘shells’ for user interaction;
- system tools for easy access to system functionality.

Loosely: the common features of *all* installations of a particular OS.

Operating System

Standard Library

System library:

- Mac OS X: `/usr/lib/libSystem.dylib`
- Windows: `C:/windows/system32/kernel32.dll`
- Linux: `/lib/libc.so.6`

Many other libraries installed by default on each OS.

- graphics
- networking
- security
- device handling

Operating System

Shell

User-interface to computer's services:

- permits the user to:
  - inspect the running state of the system;
  - perform file and device manipulation;
  - launch new processes.
- examples:
  - Windows Explorer, Finder
  - `cmd.exe`, `bash`

Operating System

Shell

Launching new processes:

- Similar to booting:
  - find program code on disk;
  - load program code into memory;
  - start executing code in memory;

Operating System

Shell

Launching new processes:

- Similar to booting:
  - find program code on disk;
  - load program code into memory;
  - start executing code in memory;
  - return control to the shell.
- new processes:
  - applications
  - system tools

Operating System

Shell

Launching new processes:

- Similar to booting:
  - find program code on disk;
  - load program code into memory;
  - start executing code in memory;
  - return control to the shell.
- new processes:
  - applications
  - system tools
  - (no clear distinction between these)

Operating System

Shell

New processes:

- identify program by name;
- identify *command-line arguments*;
- execute program, passing it its arguments.

Example: `more file.txt`

- program is '`more.exe`' (Windows) or '`more`' (OS X, Linux);
- command-line argument is '`file.txt`'
- execute `more` program, passing it the string '`file.txt`'

Operating System

Shell

New processes:

- identify program by name;
- identify *command-line arguments*;
- execute program, passing it its arguments.

Example: `more file.txt`

- program is '`more.exe`' (Windows) or '`more`' (OS X, Linux);
- command-line argument is '`file.txt`'
- execute `more` program, passing it the string '`file.txt`'
- `more` is a program which displays a file on the terminal with some user interface;
- `more` attempts to open the file named '`file.txt`'
- if successful, `more` displays the file's contents.