VU-Kernel

A Simple Kernel Written in C

Jakub Clark
j.n.clark@student.vu.nl

VU University

July 4, 2019

Agenda

1. Introduction
2. Bootstrapping
3. Interrupt Handling
4. Memory Management
5. Storage
6. Chell & Demo
7. Looking Back & Forward

Introduction

About Me

- Computer Science Student
- Intern as a Software Engineer at FireEye
- Most of my programming experience is with Python, Java and a bit of Go

Goals

- Learn more about kernels
- Gain more experience with C
- What goes into designing a kernel/operating system?
- How is memory management done?

Bootstrapping

Bootstrapping

GRUB is the boot-loader, that takes the compiled kernel and loads it into memory.

Information Provided by Multiboot

- Memory Regions
- Total Memory
- Graphics Mode
- Height & Width of the Screen
- And more...

Interrupt Handling

Interrupt Handling

The Interrupt Descriptor Table is responsible for handling incoming interrupt requests.

- Getting input from keyboard
- Page Faults
- And much more...

Interrupt Request comes in -> CPU checks if the IDT has a handler for the incoming IRQ -> calls the IRQ handler

Memory Management

What is Paging?

- The primary way of memory management is with Paging.
- Virtual addresses can make use of the entire 4GB of memory.
- The (available) memory is divided into 4096-byte pages:

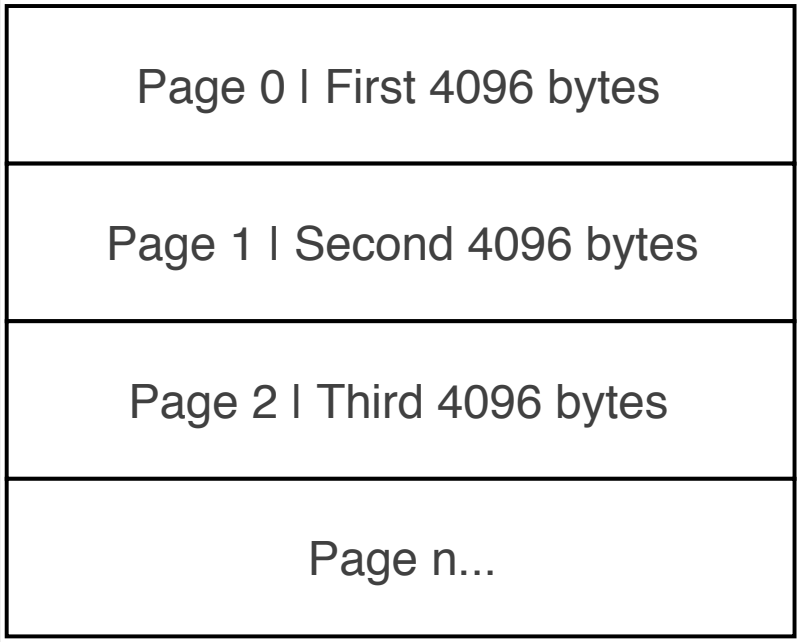

Figure: Memory Divided into Pages

Paging Example

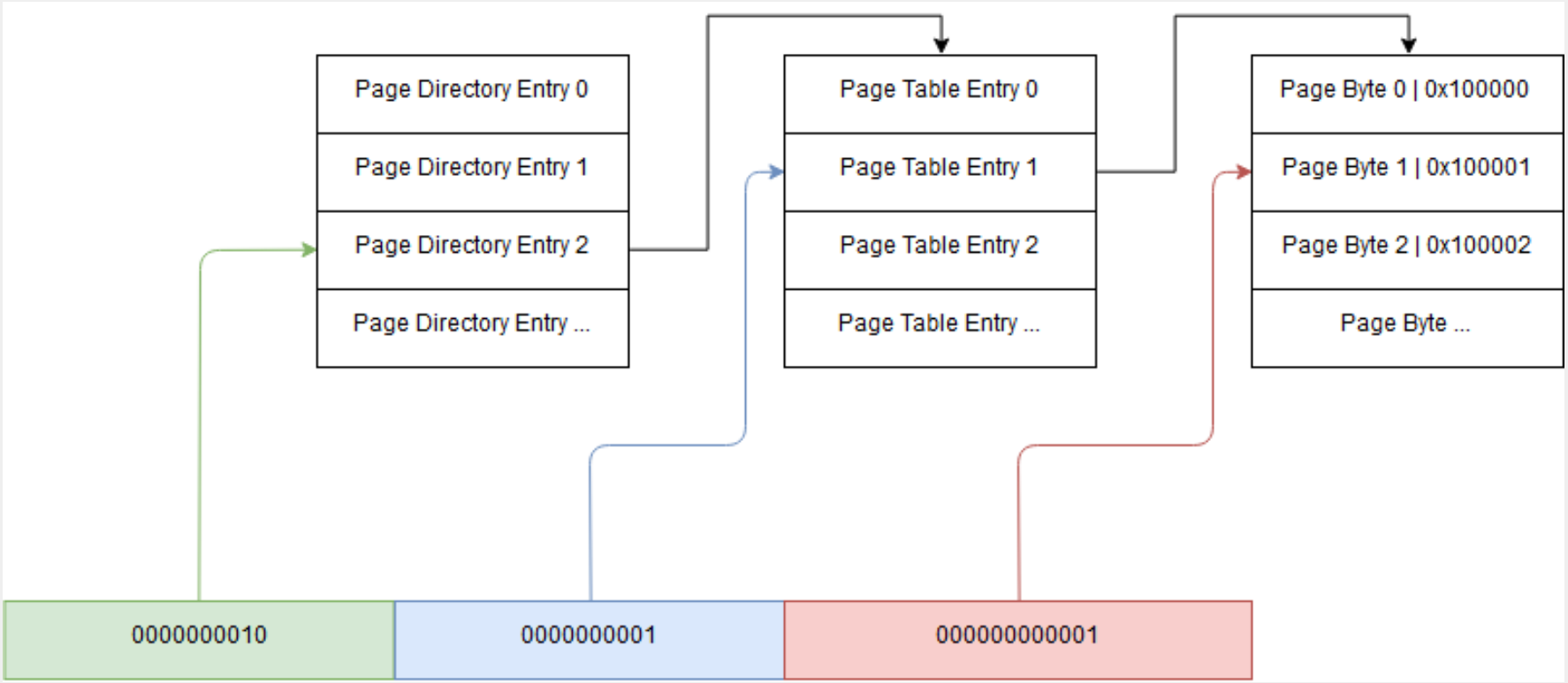

Figure: Paging Example

Paging in C

There is one Page Directory, for the kernel.

```
uint32_t kernel_directory[1024] __attribute__((aligned(4096)));
```

The associated Page Tables are also created, and referenced to, by the Page Directory Entries.

Memory Allocation

Requesting Memory

- Keeping track of physical memory is done by a bitmap.

Bitmap Drawbacks

- Simple solution, but not very efficient
- Better solution: Buddy Memory Allocation
  - Linux uses this solution

Storage

ATA

- ATA Driver, using Port-Mapped IO.
- Read/Write in 512-byte chunks - 1 Sector at a time
- Every byte goes through the CPU.

PIO Drawbacks

- Simple solution, but not very performative
- Better solution: DMA
  - Performs better, as the CPU is not needed to transfer every byte.

VFS/FAT12

- VFS meant to separate the File System Implementation from the interface provided to the programmer.
- Implemented ‘ls‘, ’cd’ and ‘pwd‘

Chell & Demo

Chell

- Shell-like REPL
- Waits for commands, and executes them
- Main way of interacting with the kernel

Demo

Demo Time

Looking Back & Forward

Looking Back

Learned a lot about kernels and memory management

- Kernels are very complex, with a lot of components
- All the components have to work together
- Kernels/OS's hide the (very) ugly
- Standard Libraries hide the ugly

Learned to program in (pure) C...

Looking Back

Learned a lot about kernels and memory management

- Kernels are very complex, with a lot of components
- All the components have to work together
- Kernels/OS's hide the (very) ugly
- Standard Libraries hide the ugly

Learned to program in (pure) C... but don't plan on programming in C.

Looking Forward

While writing VU-Kernel, some interesting topics I read about:

Looking Forward

While writing VU-Kernel, some interesting topics I read about:

- GIL in Python

Looking Forward

While writing VU-Kernel, some interesting topics I read about:

- GIL in Python
- Rust vs C - Memory Management - Ownership & Borrowing

Looking Forward

While writing VU-Kernel, some interesting topics I read about:

- GIL in Python
- Rust vs C - Memory Management - Ownership & Borrowing
- Kernels written in:
  - Go - Bare Metal Gophers by Achilleas Anagnostopoulos
  - Rust - by Philipp Oppermann

Questions?

Any Questions?

Source code can be found at:
`https://github.com/jakubclark/vu-kernel`