hhuOS

Burak Akguel, Christian Gesse, Filip Krakowski, Fabian Ruhland

Institute of Computer Science
Heinrich-Heine-University Düsseldorf

18. Juli 2018

Introduction

Facts about hhuOS

- A small operating system for teaching and learning purposes
- written for x86 32-bit architecture (64-bit maybe later)
- written in C++ and x86-Assembler using g++ and nasm
- Open-Source, published under the GPL v3 license

hhuOS - Features

- Round-robin based preemptive scheduling for threads
- Support for AHCI, USB (partially), PCI and VESA-Graphics
- Different memory managers
- Paging with higher half kernel
- FAT- Filesystem and VFS

Memory & Paging

Overview : Memory & Paging

- Paging is used to abstract physical memory from virtual address spaces
- new pages can be mapped in/out dynamically
- it is possible to create different address spaces
- Kernel is mapped at 3GB $\rightarrow$ Higher-Half-Kernel
- the addresses above 3GB are mapped into all address spaces as Kernelspace
- everything below 3GB can be used for usermode later

Difficulties implementing paging

The bootstrapping process:

- How to allocate memory when no memory manager is available?
- How to map the Kernel-code at 3GB without losing the EIP?
- Solution: activate paging in three steps
  1. First: Create a rough basic mapping for important areas using 4MB paging
  2. Then initialize all important memory managers and the page frame allocator
  3. Set up the first 4KB-Pagedirectory and reload CR3 register

Difficulties implementing paging

Invoking BIOS-calls

- BIOS-calls are necessary to set up VESA-Graphics
- BUT: BIOS-calls run in 16-bit mode without paging
- every BIOS-call would crash immediately if used with Higher-Half Kernel
- Solution:
  1. Switch to a simple 4MB-Pagedirectory that maps the Kernel to low addresses
  2. jump down to low addresses with EIP and switch to 16-bit mode without paging
  3. After returning from BIOS-call restore the old state

Library Features

Library Features

Array copies can be created using a simple assignment

code/Arrays.cpp C++

```cpp
Util::Array<uint32_t> a = {1, 2};

Util::Array<uint32_t> b = a;

a[1] = 3;

printf("a[0]=%d , a[1]=%d\n", a[0], a[1]);

printf("b[0]=%d , b[1]=%d\n", b[0], b[1]);
```

```
a[0]=1 , a[1]=3
b[0]=1 , b[1]=2
```

Kernel Modules

hhuOS supports loading kernel modules at runtime

code/Modules.cpp C++

```cpp
auto moduleLoader = Kernel::getService<ModuleLoader>();

auto file = File::open("/mod/random.ko", "r");

moduleLoader->load(file);
```

compiled modules can be placed on an external storage device

Future Work

Future Work

- Implement more device drivers (sound, graphics card, etc.)
- Ring protection / user mode
- Process system
- Enhanced scheduling (priorities, I/O management)
- New (custom) filesystems
- Multicore support

Fragen