

# hhuOS

Christian Gesse, Fabian Ruhland



07.07.2022

Institute of Computer Science  
Heinrich-Heine-University Düsseldorf

- A small operating system for teaching and learning purposes
- written for x86 32-bit architecture
- written in C++ and x86-Assembler using gcc and nasm
- Open-Source, published under the GPL v3 license
- ~ 25000 lines of code

- Processes & Threads
  - Round-Robin based preemptive scheduling
  - Binary files are executed as processes
  - Each process has its own address space

- Processes & Threads
  - Round-Robin based preemptive scheduling
  - Binary files are executed as processes
  - Each process has its own address space
- Address spaces and memory management
  - Using x86-Paging mechanism (virtual/physical memory)
  - Lazy mapping implementation
  - Kernel address space is always mapped at 3 GiB (not accessible from user space)
  - Different memory managing algorithms are implemented (Free List, Bitmap, Table)

## ■ Unified Library

- Single codebase for user- and kernel-space library
- Functions requiring kernel access are outsourced into an interface (implemented two time - for kernel and user space)
- Kernel access from user space via system calls (software interrupts)

## ■ Unified Library

- Single codebase for user- and kernel-space library
- Functions requiring kernel access are outsourced into an interface (implemented two time - for kernel and user space)
- Kernel access from user space via system calls (software interrupts)

## ■ Further features

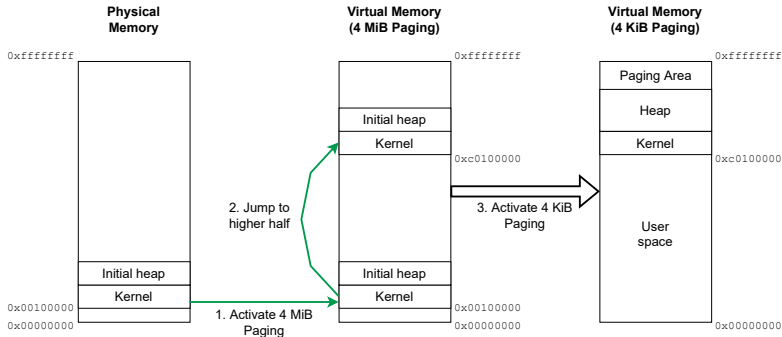
- Virtual Filesystem (can mount physical filesystems)
- Hardware support (Keyboard, PCI, Floppy AHCI, VESA & CGA Graphics)
- Compatible with UEFI and BIOS systems (BIOS calls are supported)
- Own UEFI bootloader (towboot, developed by Niklas Sombert)
- Experimental support for networking, thanks to several bachelors thesis (2 drivers and a UDP/IP Stack)

- Paging: Abstract physical memory from virtual address spaces
- New pages can be mapped in/out dynamically
- Use of different address spaces for process separation
- Kernel is mapped at 3 GiB → Higher-Half-Kernel (always visible)
- Addresses below 3 GiB are used for user space memory

- How to allocate memory when no memory manager is available?
- How to map the Kernel-code at 3 GiB without losing the EIP?
- Solution: Activate paging in several steps



- How to allocate memory when no memory manager is available?
- How to map the Kernel-code at 3 GiB without losing the EIP?
- Solution: Activate paging in several steps



- 1 Invoke Free list manager to find free memory block in heap
- 2 Slice free block (with respect to alignment)
- 3 Return pointer to found block
- 4 If the block was not previously mapped, the first access will generate a pagefault
- 5 Fault handler is invoked during interrupt handling
- 6 Search unused pageframe and map it to the virtual fault address
- 7 Return to program

- *StorageDevice* as interface for block devices
- Only 4 methods: *getSectorCount()*, *getSectorSize()*, *read()*, *write()*
- Implemented for AHCI (hard drives), Floppy, Virtual Drives, and Partitions

- Overlay over physical file systems (e.g. FAT)
- Storage devices can be mounted to any folder
- Files and directories are represented by `Filesystem::Node` (similar to *Inode*)
- Interface `Filesystem::Driver` for physical file systems
- Integrate *FatFs*<sup>1</sup> to support all FAT variants

<sup>1</sup>[http://elm-chan.org/fsw/ff/00index\\_e.html](http://elm-chan.org/fsw/ff/00index_e.html)

- Global scheduler manages processes in a queue
- Timer invokes scheduler every 10ms to switch to next process (and address space)
- Each process has a separate thread scheduler, switching the current thread each time the process becomes active
- Threads can be blocked and unblocked by other threads
- Threads can sleep for a defined time
- It is possible to join other threads (or processes)

- Implement more device drivers (sound, graphics, network, etc.)
- Enhance network stack
- Better scheduling (priorities, I/O management)
- New (custom) filesystems
- Multicore support
- Support modern x86-Features (Physical Address Extensions, Long mode, SSE, etc.)