📁 Project Documentation

🛠️ PoC 64-bit Operating System — ELF + WinPE Support (MIT Licensed)

Languages: x86_64 Assembly & C
Test Platform: QEMU
Bootloader: GRUB2 (external)
Filesystem: Custom EXT4 driver + mkfs
Binary Support: Custom ELF and WinPE Parsers
License: MIT

🧾 Part 1: Legal Documentation

1.1 Project Licensing and Identity

- **License**: MIT
- **Project Status**: Proof of Concept (PoC)
- **Development**: Entire kernel, binary loaders, EXT4 FS, and tools are original.
- **Third-Party Bootloader**: GRUB2 used externally for multiboot2-compliant loading.

MIT License Declaration:

```
MIT License

Copyright (c) 2024-2025 COS

Permission is hereby granted, free of charge, to any person
obtaining a copy of this software and associated documentation
files (the "Software"), to deal in the Software without
restriction, including without limitation the rights to use, copy,
modify, merge, publish, distribute, sublicense, and/or sell copies
of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:
```

```
The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.
```

1.2 Third-Party Usage and Scope

GRUB2 only loads the kernel; the kernel and drivers are not GPL-contaminated. EXT4 support (driver + mkfs) is original and not derived from Linux kernel code. QEMU is used for testing only, not redistributed or integrated. No reuse of Windows or Linux code in ELF/PE parsers.

Component	Source/Origin	License Type of Use Notes
GRUB2 Bootloader	GNU GRUB	GPLv3
QEMU Emulator	QEMU Project	GPLv2
EXT4 Specification	Kernel Docs	Public Domain Functional reference
PE/ELF Specs	MS / SysV ABI	Public Docs
ASM Boot Chain	Public Docs (Intel Manual, OSDev) and Open Source Projects Explanations	Public Domain Documentation of Several Aspects of the Boot Sequence

It should be noted that a C reimplementation has been considered for the final release due to the similarities present within the source code and other projects as the boot sequence may only be made following steps that are always the same.

1.3 IP Risk & Compliance Mitigation

Binary formats:

- ELF and PE are supported through specs and observation, not code reuse.
- Windows-specific behavior is avoided unless publicly documented.

Filesystem:

- EXT4 support based on structure, not code lifting.

Bootloader:

- GRUB2 is treated as an external loader; avoid bundling to preserve MIT purity.

📡 PART 2: TECHNOLOGY WATCH (2025)

Ongoing awareness of tools, formats, kernel development, and licensing updates.

2.1 Monitored Topics

Date	Topic	Source	Summary
2025-05-24	Recursive Mapping	https://os.phil-opp.com/page-tables/#recursive-mapping	Clever trick wher the page tables m themselves — a lo that simplifies pa table manipulatio This was added during the creatio of the memory mapping algorith as we realised tha we needed a way access every virtu physical addresse the nodes formin the page table.
2025-04-14	GCC Stack Canaries	https://ctf-wiki.mahaloz.re/pwn/linux/mitigation/canary/	Explains how GC inserts secret valu into the stack to

Date	Topic	Source	Summary
			catch and halt bu overflows. During compilation, we encountered an is where GCC would add __chk_fail() around some of th functions. An incomplete implementation needed to be provided.
2025-02-13	RTC Initialisation for EXT4 Metadata	https://web.archive.org/web/20150514082645/http://www.nondot.org/sabre/os/files/MiscHW/RealtimeClockFAQ.txt	Details how to wo with the Real-Tim Clock. We decided implement prope support for the RT to provide correct timestamps for th EXT4 filesystem.
2025-01-11	Understanding Virtual Paging	https://zolutal.github.io/understanding-paging/	Explains how the system translates virtual addresses physical ones.
2024-11-30	Memory Management	https://wiki.osdev.org/Memory_Map_(x86)	Lays out the phys memory map on x It became relevan once the MemoryManagem module could fina be addressed properly.
2024-11-17	Disk ATA PIO Mode (48bit)	https://wiki.osdev.org/ATA_PIO_Mode	A guide to issuing low-level disk commands via AT PIO using 48-bit addressing. Our t had to write a dri for ATA disks to b able to communic

Date	Topic	Source	Summary
			with Qemu's virtu hardware.
2024-11-17	ATA Registers	https://atola.com/products/tf1/manual/registers.html	Inventory of I/O ports and register needed to communicate directly with ATA drives.
2024-11-10	Higher Half Setup	https://medium.com/@connorstack/how-does-a-higher-half-kernel-work-107194e46a64	Explains why and how kernels are mapped high in virtual memory – separating them from userspace cleanly. This was implemented as a preventative measure long bef the Memory Manager was actually started.
2024-10-06	OS Initialisation	https://www.gnu.org/software/grub/manual/multiboot/multiboot.html	The Multiboot specification — th handshake betwe bootloaders and kernels, ensuring smooth entry into land. This structu needed to be properly written ensure that GRUB would return pro hardware information, including the free and reserved memory block addresses.
2024-09-15	EXT4 Specification	https://www.kernel.org/doc/html/v6.9/filesystems/ext4/index.html?highlight=ext4	Technical exposit of EXT4's internal mechanisms, as

Date	Topic	Source	Summary
			implemented in t Linux kernel.
2024-09-15	EXT4 Disk Layout	https://blogs.oracle.com/linux/post/understanding-ext4-disk-layout-part-1	Walks through ho EXT4 organizes d physically on disk was used during t initial implementation o the cos_mkfs function as well a the read/write syscalls.
2024-09-08	Translation Lookaside Buffer	https://en.wikipedia.org/wiki/Translation_lookaside_buffer	The TLB is a cach for memory mappings that speeds up addres resolution. TLB fl sequences were added long before the Memory Manager was completed as a preventative measure once we would reach virtu address mapping.
2024-09-02	Allowed Freestanding Headers	https://wiki.osdev.org/C_Library#Freestanding_and_Hosted	Lists the standard library headers permissible wher one's compiling without an OS — freestanding environment. Thi link was especiall useful as we were sure whether or n we could use the va_args functiona usually provided complete OSes.
2024-08-31			Deep dive into a legacy interrupt

Date	Topic	Source	Summary
	8285A PIC Documentation	https://pdos.csail.mit.edu/6.828/2012/readings/hardware/8259A.pdf	controller. It was especially relevan considering the fa that we had troub initialising it at fi
2024-08-31	8295A PIC Initialisation Procedure	http://www.brokenthorn.com/Resources/OSDev16.html	Walkthrough for initializing the PI sets the stage for interrupt-driven execution. This st was required as enabling long mo without it would invariably end up triggering Triple Faults due to the hardware timer o IRQ0 being improperly remapped.
2024-08-31	AMD64 Red Zone	https://forum.osdev.org/viewtopic.php?t=21720	The AMD64 Red Z is a quiet 128-byte region below the stack pointer that shouldn’t be touc in leaf functions - per the AMD64 AI We recall that dur the initialisation process, we had a of an issue setting the PIC and resea into the matter le to the discovery o several compile fl we should have integrated into th project on the ker side at the very le
2024-08-31	Better Debugger	https://sourceforge.net/projects/bochs/files/bochs/2.8/	Bochs includes a detailed software debugger tailored

Date	Topic	Source	Summary
			emulating and inspecting low-lev x86 internals. It proved useful for debugging hardw related issues we encountered.
2024-07-04 and 2024-08-31	IDT Documentation	https://wiki.osdev.org/Interrupt_Descriptor_Table	Covers how to wi up the table that routes hardware software interrup to handlers. It wa required for initialisation of th hardware.
2024-07-04 and 2024-08-31	GDT Documentation	https://wiki.osdev.org/Global_Descriptor_Table	Guide to building Global Descriptor Table. It was required for initialisation of th hardware.
2024-06-24	ASM Instructions	https://www.felixcloutier.com/x86/	A rich catalog of x instruction sets — invaluable for decoding binary behavior and crafting assembly
2024-06-24	OS Initialisation	https://cdrdv2.intel.com/v1/dl/getContent/671447	Intel's roadmap fr power-on to OS handoff — BIOS, firmware, and ea hardware setup.

2.2 Future considered enhancements/features

Date	Topic	Source	Summary
2025-05-11	Zero Page	https://en.wikipedia.org/wiki/Zero_page	At some point, we asked ourselves how complete OSes handled NULL

Date	Topic	Source	Summary
			pointer dereferences as Linux defined it as (void *)0. We ended up discovering that this Virtual Address acted as a trap instead and should be reserved by the kernel. It still hasn't been implemented within COS.
2025-05-04	Program Launching	https://electronics.stackexchange.com/questions/440421/int-functions-with-no-return-statement	Once program launching became a concern once more, we asked ourselves how complete OSes would handle main() functions with no return, as the CPU would end up reading garbage if it was called as a function pointer. It is then that we realised that launching a full program

Date	Topic	Source	Summary
			wouldn’t be as simple as using function pointers to call the main() function and instead that we would need a scheduler to gracefully handle context switching.
2025-04-08	Memory Management Optimisation/ Feature	https://www.kernel.org/doc/gorman/html/understand/understand011.html	Compared to our custom Memory Manager, this model would pre-partition the RAM into caches managing slabs (blocks of contiguous memory). With these caches, select commonly used objects can be reused without the need to reallocate them first, therefore speeding up operations. For instance, allocating a structure and then asking the kernel to

Date	Topic	Source	Summary
			free it would instead keep the structure in memory [with its internal information being reinitialised] only if it is commonly accessed.
2025-02-13	ELF Specification	https://refspecs.linuxfoundation.org/elf/elf.pdf	Blueprint for the ELF binary format — the standard for Unix-like systems' executables and object files. This will be required for the next part of the project and thus hadn't been implemented yet.
2024-09-15	Hash Tree Algorithm	https://blogs.oracle.com/linux/post/understanding-ext4-disk-layout-part-2	This blog post discusses the Hash Trees used by the EXT4 specification for storing directory entries. By default, they are linearly written in directory data

Date	Topic	Source	Summary
			blocks. This newer way of indexing files and folders would allow for the search complexity to be logarithmic instead.
2024-09-08	Memory Management Optimisation/ Feature	https://en.wikipedia.org/wiki/Copy-on-write	Technique that delays memory duplication until modification — efficient for process creation and memory savings. Once scheduling tasks will be implemented, shared libraries (which are read-only resources) could be accessed by several programs in a memory efficient way.
2024-09-08	Memory Management Optimisation	https://www.scaler.com/topics/operating-system/page-replacement-algorithm/	A tour of strategies for deciding which memory pages get evicted — central to managing

Date	Topic	Source	Summary
			RAM pressure. This Memory Management technique has been considered as an evolution of the project where multitasking is available.
2024-08-31	APIC Documentation	https://web.archive.org/web/20140308064246/http://www.osdever.net/tutorials/pdf/apic.pdf	Intro to the Advanced Programmable Interrupt Controller — key for multiprocessor and fine-grained interrupt control. This is the newest hardware used as part of Intel CPUs, replacing the PIC. We would like to update the code with proper APIC support later down the line.
2024-06-21	WinPE File Format Specification	https://learn.microsoft.com/en-us/windows/win32/debug/pe-format	Microsoft's breakdown of the PE binary format. During the planning of project, we started work on a WinPE binary parser.

2.3 Excluded Enhancements/Features

Date	Topipc	Source	Summary
2024-09-06	Level 5 Paging	https://www.intel.com/content/www/us/en/content-details/671442/5-level-paging-and-5-level-ept-white-paper.html	Discusses the expansion of virtual address space with 5-level paging — useful when dealing with colossal memory ranges (more than 256TiB). Due to the fact that the COS will never need that much memory to function properly, this technology was discarded early on during the planning phase of the project.

🤝 PART 3: EXPERT INTERACTIONS & CONTACT LOG

3.1 Expert Directory

Name / Alias	Specialty	Affiliation / Project	Contact	Status
Dr. Moy	Compilation	Independent	Matthieu.Moy@univ-lyon1.fr	Active

3.2 Conferences & Community Engagement

Contacts/discussion boards which have been considered but haven't yet been reached out for: - OSDev Forums (Help related to OS Development and low level) - CrossOver Team (Main contributor to the WINE project, a translation layer for Linux)

📎 APPENDICES

Yearly Development Timeline

Start date	Target date	Status
2025-05-31	2025-06-05	Mickael: Adding legal documentation. Corentin + Mickael: Continuing work on MemoryManager. Thomas + Adam: Continuing remaining work on FileSystem.
2025-05-13	2025-05-30	Mickael + Corentin: Fixing Memory Mapping and adding proper Kernel Mapping (to be tested thoroughly). Adam: Continuing work on touch. Thomas: Continuing work on mv.
2025-04-29	2025-05-13	Mickael + Corentin: Adding Memory Allocation and starting work on Memory Mapping. Thomas: Continuing work on mv. Adam: Continuing work on touch.
2025-04-14	2025-04-29	Mickael: Adding ls command PoC. Corentin + Mickael: Continuing Memory

Start date	Target date	Status
		Management Revised algorithm (Page Frame Manager and Reclaiming Allocated Unused Kernel Pages) Adam: Starting work on the touch command. Thomas: Starting work on the mv command.
2025-03-31	2025-04-14	Adam + Thomas: Adding full support for EXT4 read capabilities. Adam + Thomas: Adding full read/ write wrappers. Mickael: Fixing the sectorcount bug + Merging of every branches into one. Corentin: Starting the revision of the MemoryManagement algorithm (Page Frame Manager and Free Memory Claiming)
2025-03-15	2025-03-27	Mickael:Adding support for stack smashing canaries being added by GCC. Fixing the memset bug by adding a properly named memset function. Adam + Thomas: Continuing work on the read/write wrappers.
2025-02-28	2025-02-15	

Start date	Target date	Status
		Adam: Fixing a part of the write wrapper + Working on read wrapper. Thomas: Working on write wrapper. Corentin: Adding more printf arguments and updating the command line to support arguments. Mickael: Fixing the EXT4 Support.
2025-02-11	2025-02-24	Adam + Thomas: Starting work on the read and write wrappers respectfully. Adam + Corentin: Fixed Block Descriptor entries and Block Descriptor Bitmaps. Mickael: Added Inode Headers and Inode Bitmaps. Initialisation of default inodes Added support for RTC to the COS in order for timestamps to work. Thomas: Started research on Linux ELF parsing.
2025-01-31	2025-02-13	Thomas: Fixed the read/write ASM functions. Mickael: Fixed the read/write ASM functions + Added
2025-01-01	2025-01-11	

Start date	Target date	Status
		incomplete Virtual Addressing support.
2024-12-14	2024-12-15	Corentin: Work on MemoryManagement. Adam: Work on Ext4 filesystem. Thomas: Debugging of the Read/Write function phase. Mickael: Started working on Documenting and sourcing the project under the Documentation module.
2024-12-07	2024-12-08	Corentin: Work on MemoryManagement. Adam: Work on Ext4 filesystem. Thomas: Debugging of the Read/Write function phase. Mickael: Started working on Documenting and sourcing the project under the Documentation module.
2024-11-30	2024-12-01	Corentin: Started work on MemoryManagement by creating necessary structures. Adam: Continued work on Ext4 filesystem. Thomas: Debugging of the Read/Write function phase. Mickael: Started work

Start date	Target date	Status
		on RAM block ordering in memory.
2024-11-23	2024-11-24	Corentin: Tried to fix the mirroring to push every module to the EPITECH mirror. Adam: Continued work on Ext4 filesystem. Thomas: Added Read/Write ASM functions to the core module. Testing phase. Mickael: Tried to fix the mirroring to push every module to the EPITECH mirror + Helped Thomas on debugging.
2024-11-16	2024-11-17	Corentin: Work on Unit Testing support and some code refactoring work. Adam: Continued work on Ext4 Filesystem. Thomas: Continued work on Read/Write ASM functions. Mickael: Finished working on Higher Half Kernel + RAM space retrieval thanks to GRUB.
2024-11-09	2024-11-10	Corentin: Work on Unit Testing support and initial push to the EPITECH mirror. Adam: Continued work on Ext4 Filesystem blocks.

Start date	Target date	Status
2024-11-02	2024-11-03	Thomas: Continued work on Read/Write ASM functions. Mickael: Helped Adam on Ext4 filesystem blocks + continued work on Higher Half Kernel. Corentin: Worked on adding the workflow to the project. Adam: Continued work on Ext4 Filesystem blocks. Thomas: Continued work on Read/Write ASM functions. Mickael: Sending the Kernal to Higher Half Memory.