Republic of Rwanda
Ministry of Education

RTB | RWANDA TVET BOARD

HOBBY KERNEL DEVELOPMENT

CSAHK501

Develop Hobby Kernel using C

Competence

RQF Level:	5
Credits:	12
Sector:	ICT and Multimedia
Trade:	Computer System and Architecture
Module Type:	SPECIFIC
Curriculum:	ICTCSA5001-TVET CERTIFICATE V COMPUTER SYSTEM AND ARCHITECTURE
Copyright:	© Rwanda TVET Board, 2024

Learning Hours
120

2024-25

Issue Date: March, 2024

Purpose statement	This specific module describes the skills, knowledge and attitude required to Develop Hobby Kernel using C. This module is intended to prepare students pursuing TVET Level 5 in Computer System and Architecture. Upon completion of this module, the learner will be able to Prepare working environment, Implement Memory Management, Implement Process Management, implement persistence management, and implement a core kernel.
Learning assumed to be in place	Data Structures and Algorithms using C.

Delivery modality	**Training delivery**		**100%**	**Assessment**		**Total 100%**
	Theoretical content		**20%**	Formative assessment	20%	**50%**
	Practical work:		**80%**		80%	
	Group project and presentation	**30%**				
	Individual project /Work	**50%**				
			Summative Assessment			**50%**

Elements of Competence and Performance Criteria

Elements of competence	Performance criteria
1.Prepare working environment	1.1. Tools and equipment are properly selected based on hobby kernel requirements
	1.2. Software tools are properly installed based on installation guide and system specifications
	1.3. Virtual environment is effectively configured inline with hardware compatibility
2.Implement Memory management	2.1. Memory hierarchy and addressing is properly implemented based on a specific addressing mode
	2.2. Memory allocation is properly implemented based on operating system support

	2.3. Layout of process address space and protection is effectively managed based on memory usage optimization.
	2.4. Virtual memory and swapping techniques are efficiently applied based on system workload
	2.5. Defragmentation Techniques are properly applied based on memory allocation and management best practices
3.Implement Process Management	3.1. Processes and threads are properly implemented based on system workloads.
	3.2. Process scheduling algorithms are properly implemented based on system workloads
	3.3. Parallelism are properly applied based on the intended performance and efficiency
4.Implement persistence management	4.1. Mass storage structure is properly managed based on the storage form factor
	4.2. Input/ Output systems are properly applied based on the interrupt or Direct Memory Access (DMA) techniques
	4.3. File systems are properly implemented based on operating system structure
5.Implement core Kernel	5.1. Assembly language concepts are properly applied based on language standards
	5.2. The core kernel is properly initialized based on the intended use and hardware they want to support.
	5.3. A simple bootloader is properly developed based on target system architecture
	5.4. Interrupt handling is properly implemented based on operating system requirements
	5.5. A function to display console output on the screen in a kernel is properly implemented based on display hardware being used.
	5.6. System call is properly implemented based on operating system kernel architecture.

Knowledge	Skills	Attitude
• Description of hobby kernel. • Identification of the required tools for hobby kernel development. • Identification hardware compatibility. • Description memory addressing and its modes. • Description memory allocation and management unit. • Description layout of process address space. • Identification fragmentation techniques. • Description process state and scheduling algorithms. • Identification Inter-Process Communication primitives. • Identification mass storage structure. • Description assembly Language and x86 architecture.	• Installation of required software tools. • Configuration of virtual environment. • Applying of memory addressing. • Implementation of Cache. • Implementation of memory allocation. • Implementation virtual memory. • Applying fragmentation techniques. • Implementation process control blocks. • Implementation multithreading and synchronization mechaniques. • Development of scheduling algorithms. • Implementation file system • Implementation I/O systems. • Development assembly hello, world. • Applying register commands. • Writing and build C and Assembly kernel objects. • Development a simple boot loader. • Implementation of OS displays. • Implementation system calls.	• Paying attention to detail • Flexibility • Adaptability • Team work • Persistence • Time management and organized • Curiosity • Passion • Creativity • Patience • Integrity • Reliability • Trustworthy • Honesty • Innovation • Confidence

Course content

Learning outcomes	At the end of the module the learner will be able to:
	1. Preparing a working environment 2. Implement Memory Management 3.Implement Process Management 4. Implement persistence management 5. Implement a core kernel
Learning outcome 1: Prepare working environment	**Learning hours: 10**

Indicative content

- **Selection of tools and equipment**
  - ✓ Hobby kernel description
    - Definition
    - Role of kernel in operating system
    - Benefits of hobby kernel development
  - ✓ Requirements for hobby kernel
    - Target architecture
    - Virtualization tools
    - Development tools
  - ✓ Identification of the right tools and equipment to use in development.

- **Installation of Software tools**
  - ✓ QEMU
  - ✓ QtEMU
  - ✓ A HEX Editor
  - ✓ Text editor (VSCode)
  - ✓ NASM
  - ✓ SASM
  - ✓ MinGw For Compiling C Programs
  - ✓ Configuring MinGw For Compiling C Programs

- **Configuring virtual environment**

✓ Identifying hardware compatibility ✓ Configuring installed software to Environment Path ✓ Configuring installed virtualization tools ✓ Running, checking, and testing tools and environment	
Resources required for the learning outcome	
Equipment	Computer, and Projector
Materials	Internet
Tools	HexEditor, Code editor, NASM, SASM, QEMU, QtEMU, Compiler, vmware workstation.
Facilitation techniques or Learning activity	Demonstration, Group discussion, Practical exercises, and Trainer-guided
Formative assessment methods /(CAT)	Written assessment, Oral presentation, and Practical assessment

Learning outcome 2: Implement Memory management	**Learning hours: 25**
Indicative content	
● **Implementation of memory hierarchy and addressing.** ✓ Description + Definition + Hierarchy + Addressing modes + Mechanisms for cache management + Cache replacement policies + Snooping ✓ Applying memory addressing. + Address Translation Scheme + Direct addressing + Indirect addressing + Indexed addressing + Register addressing ✓ Implementation of Cache	

        - Developing algorithms (LRU (Least Recently Used), LFU (Least Frequently Used))
        - Implementing Cache Coherency using MESI or MOESI protocols
        - Testing Cache
    - ✓ Applying Memory Mapping
        - Using paging
        - Using Segmentation
- **Implementation of memory allocation.**
    - ✓ Description of Memory Management Unit
        - Definition
        - Role
        - Structure of Page Table
    - ✓ Implementing Page Table Management
        - Data structures for managing page tables
        - Algorithms for managing page tables
    - ✓ Description of Memory Allocation
        - Definition
        - Memory allocation strategies
    - ✓ Apply Memory Allocation
        - Memory Allocation Algorithms (first-fit, best-fit, or worst-fit)
        - Write test cases
    - ✓ Description of Memory Protection
        - Memory regions.
        - Mechanisms.
    - ✓ Applying Memory Protection
        - Setting access permissions
        - Using hardware-based memory protection
- **Managing layout of process address space and protection**
    - ✓ Process address space.
        - Organization
        - Memory-based attacks
        - Manage Address Space Layout Randomization (ASLR) Settings
    - ✓ Stack and Heap Management
        - Stack algorithm
        - Heap algorithm
    - ✓ Memory usage optimization.
- **Applying Virtual memory and swapping techniques.**
    - ✓ Definition
        - Virtual memory
        - Swapping

- ✓ Applying Page Replacement Algorithms
    - FIFO (First In First Out)
    - LRU (Least Recently Used)
    - Clock algorithm
- ✓ Applying Demand Paging
    - Set required pages into memory
    - Page fault handling
- ✓ Applying Swap Space Management
    - Techniques for managing swap space
    - Reducing disk I/O overhead

- **Applying defragmentation techniques.**
  - ✓ Description
    - Definition
    - Types
    - Techniques
  - ✓ Performing defragmentation
    - Memory Compaction
    - Buddy System Allocation
    - Memory Pooling

Resources required for the indicative content	
Equipment	Computer, and Projector
Materials	Internet
Tools	HexEditor, Code editor, NASM, SASM, QEMU, QtEMU, Compiler
Facilitation techniques or Learning activity	Demonstration, Group discussion, Practical exercises, and Trainer-guided
Formative assessment methods /(CAT)	Written assessment, Oral presentation, and Practical assessment

Learning outcome 3: Implement Process Management	Learning hours: 25

Indicative content

- **Implementation of processes and threads**
    - ✓ Description of key terms
        - Process
        - Thread
        - Multi-thread
        - Process control block
        - State process model
        - No preemptive
        - Preemptive
        - Cooperating process(inter-process)
        - User level threads
        - Kernel level thread
        - Deadlock
        - Livelock
    - ✓ Interpretation of processes
        - states diagram
        - tree
    - ✓ identifying process hierarchy important
        - foreground process
        - visible process
        - service process
        - background process
        - empty process
    - ✓ Applying Process states
        - Fork New process
        - Simulate Running process
        - Simulate Waiting process
        - Simulate Ready process
        - Simulate Terminated process
        - Creation of a thread into a process
    - ✓ Applying thread operations
        - list of operations
        - multithreading
    - ✓ Applying process control block (PCB) to manage process
    - ✓ Applying state management mechanism
        - Context switching

        - Process synchronization(locks, semaphore, monitor)
    - ✓ Applying inter-process mechanism
        - Pipes
        - Shared memory
        - Message queue
        - Signals
    - ✓ Applying algorithm to Handle deadlocks and livelocks

- **Implementation of process scheduling algorithms**
    - ✓ Description of key terms
        - Process scheduling
        - Preemptive
        - Non preemptive
        - CPU Burst
        - I/O Burst
        - CPU Scheduler
        - Dispatcher modules
        - Dispatcher latency
    - ✓ Description of scheduling criteria
        - CPU utilization
        - Throughput
        - Turnaround time
        - Waiting time
        - Response time
    - ✓ Description of scheduling algorithm optimization criteria
        - Max CPU utilization
        - Max throughput
        - Min turnaround time
        - Min waiting time
    - ✓ Applying scheduling algorithms
        - preemptive Scheduling(round robin, shortest remaining time first, priority scheduling)
        - Non-preemptive Scheduling(First come first serve, shortest job first)
    - ✓ Applying scheduling algorithm evaluation
        - Deterministic evaluation
        - Queuing model
        - Little's formula
- **Applying Parallelism**
    - ✓ Description
        - Parallelism Types (task parallelism, data parallelism, instruction-level parallelism, and thread-level parallelism)

        - Parallelism roles
    - ✓ Implementing parallelism techniques
        - Multithreading
        - Multiprocessing
        - SIMD instructions.
- **Applying Lock-Based Concurrency Control**
    - ✓ Description
        - Definition
        - Data items
        - Transactions
        - Locks(Exclusive (X) lock, Shared (S) lock
        - Benefits of Lock-BCC
        - Drawbacks of Lock-BCC
        - Alternatives to Lock-BCC
    - ✓ Implementing Lock-based Concurrency Control Protocol
        - Apply lock due to read and write
        - Apply awaiting until the conflicting lock is released
        - Releasing the lock due to transaction completion

Resources required for the indicative content	
Equipment	Computer and Project
Materials	Internet
Tools	HexEditor, Code editor, NASM, SASM, QEMU, QtEMU, Compiler
Facilitation techniques or Learning activity	Demonstration, Group discussion, Practical exercises, and Trainer-guided
Formative assessment methods /(CAT)	Written assessment, Oral presentation, and Practical assessment

Learning outcome 4: Implement persistence management	**Learning hours: 20**

Indicative content

- **Introduction to persistence management**
    - ✓ Description of key terms
        - System calls
        - Data Buffers

        - Caching
        - Disk flushing
        - Journaling
    - ✓ Identifying Data persistence Mechanism
        - Direct Writes and File System Calls
        - Journaling File Systems
- **Management of Mass storage structure**
    - ✓ Description of mass storage physical structure
        - Types
        - Characteristics
        - Disk attachment
        - Effect of a device’s structure
    - ✓ Applying free space management
        - Techniques
        - advantages
    - ✓ Applying disk scheduling algorithm
        - Disk scheduling overview
        - Importance of Disk Scheduling
        - Select Disk Scheduling Algorithms
    - ✓ Applying RAID (Redundant Arrays of Independent Disks) technique
        - Advantage
        - Levels
        - Key Evaluation Points
- **Applying I/O systems**
    - ✓ Description of I/O system in OS
        - Importance
        - Challenges of device heterogeneity
    - ✓ Identifying I/O devices
        - Types (storage, network, human interface)
        - Characteristics (speed, protocols)
    - ✓ Identifying Device Drivers
        - Function
        - Structure
    - ✓ Identifying I/O models
        - Bloking I/O
        - Non-Blocking I/O
        - I/O completion Ports(IOCPs)

    - ✓ Applying Interrupts handling
        - Interrupt-driven I/O

        - Interrupt service routines(ISRs)
    - ✓ Applying I/O scheduling
        - Scheduling multiple I/O requests
        - Common I/O scheduling algorithms
    - ✓ Apply device management
        - Device allocation and deallocation
        - Error handling (Detecting, reporting, and recovering from I/O errors)

- **Implementing file systems**
    - ✓ Describe file system
        - Definition
        - Types
        - Purpose of file systems
        - Various file system options
        - File system organization (logical view of data)
        - Directory structures (hierarchical organization, navigation)
        - File attributes (name, size, creation/modification time, permissions)
        - File system operations
    - ✓ Identification of existing file systems
    - ✓ Identification of file system selection criteria
        - Performance
        - Security
        - Compatibility
        - Advantages
        - Limitations
    - ✓ Applying basic file operations
        - Create
        - Open/close
        - Read/Write
        - Delete
    - ✓ Applying directory operations
        - Create directory
        - Delete directory
        - List directory contents
        - Change directory
        - Search for a file
    - ✓ Applying File system Management Operations
        - Rename
        - Move/Copy
        - Change permissions
        - Change the ownership

- Search a file ✓ Applying file allocation methods - Contiguous Allocation - Linked Allocation - Indexed Allocation	
Resources required for the indicative content	
Equipment	Computer and Projector
Materials	Internet
Tools	HexEditor, Code editor, NASM, SASM, QEMU, QtEMU, Compiler
Facilitation techniques or Learning activity	Demonstration, Group discussion, Practical exercises, and Trainer-guided
Formative assessment methods /(CAT)	Written assessment, Oral presentation, and Practical assessment

Learning outcome 5: Implement core kernel	**Learning hours: 40**
Indicative content	
● **Applying Assembly Language concepts** ✓ Description - Definition - History - advantages - sections - comment - statement syntax - hello world output - file extension - Compiler - x86 Processor data sizes - x86 Processor Endianness - Classification of Registers (General, Control, Segment)	

        - Variables
    - ✓ Compiling and Linking an Assembly Program in NASM
    - ✓ Applying memory segments
        - data segment
        - code segment
        - stack segment
    - ✓ Applying different categories of register
    - ✓ Applying system calls
    - ✓ Applying addressing mode
        - register addressing
        - immediate addressing
        - memory addressing
    - ✓ Applying variables and constants
    - ✓ Applying arithmetic and logical instruction
    - ✓ Applying conditions
    - ✓ Applying loops(optional)
    - ✓ Applying Numbers
    - ✓ Applying string
    - ✓ Applying array

- **Initialization of the core kernel.**
    - ✓ Description
        - Definition of a kernel
        - History of kernel development
        - Types of kernels
        - Kernel layout
        - Purpose of the kernel
        - Computer tasks to accomplish kernel functions.
        - Device drivers(Types of device drivers,Kernel mode vs User mode)
    - ✓ Write and build the object file from the assembly code using NASM (to fire the main kernel function in the C file).
    - ✓ Write and build the object file from the C code using GCC.
    - ✓ Write a linker script and saving it in an “ld” file extension.
    - ✓ Merge the Assembly-built and C-built objects using the linker script.
        - An executable kernel file will be executed.
        - Test our kernel file.
- **Development of a simple Bootloader with assembly language**
    - ✓ Introduction
        - Definition of a boot loader
        - Languages that can be used.
        - Compilers that can be used.
        - the system boots.
    - ✓ Program architecture.
    - ✓ Development environment

- ✓ Implement basic BIOS Interrupts and screen clearing
- ✓ Implement Mixed code
  - CString
  - CDisplay
  - Types.h
  - BootMain C file
  - StartPoint assembly file
- ✓ Assemble everything
  - Creation of COM file
  - Assembly automation
- ✓ Test and Demonstration
  - How to test boot loader
  - Testing with Virtual Machine VmWare
  - Testing on the real hardware
- ✓ Debug our system with tools
  - TD(Turbo Debugger)
  - CodeView
  - D86
  - Bosch

- **Implementation of Interrupt Handling**
  - ✓ Description
    - Definition
    - Interrupt Handler
    - Interrupt Descriptor Table (IDT)
    - Programmable Interrupt Controller (PIC)
  - ✓ Applying interrupt handlers
  - ✓ Creating an Entry in Interrupt Descriptor Table (IDT)
  - ✓ Handling an Interrupt
  - ✓ Creating a generic Interrupt Handler
  - ✓ Applying Programmable Interrupt Controller (PIC)
- **Implementing a function to display console output on the screen in a kernel**
  - ✓ Printing To Screen (Hello, World OS)
  - ✓ Filling the Screen with Characters(For Fun)!!
  - ✓ Filling The Screen With Colours
  - ✓ Scrolling the screen
  - ✓ Other Bios Display Related Routines

- **Implementing System call**
  - ✓ Introduction of system call
    - Types of System calls in OS
    - Examples of System calls in Windows and Unix.
    - Rules for Passing parameters to the System Call
    - Services Provided by System calls
    - Features of System calls

- System calls advantages ✓ Developing a C-file system call ✓ Modifying necessary files to integrate our system call ✓ Recompile and Reboot ✓ Testing the system call ● **Implement publication of our system** ✓ Build an .iso file from our system. ✓ Test the .iso file - Running it on virtual machine - Running it on physical machine ✓ Generate OS documentation

Resources required for the indicative content	
Equipment	Computer and Projector
Materials	Internet
Tools	HexEditor, Code editor, NASM, SASM, QEMU, QtEMU, Compiler
Facilitation techniques or Learning activity	Demonstration, Group discussion, Practical exercises, and Trainer-guided
Formative assessment methods /(CAT)	Written assessment, Oral presentation, and Practical assessment

Integrated/Summative assessment (For specific module)

XYZ Electronics Ltd located in Gasabo district, it is a growing startup that specializes in creating different computer-based solutions. They have built a computer that is supposed to manipulate .txt files. The company is facing a problem on their self-built computer as it does not run a .txt file because it doesn't have an operating system. As an OS Developer, you are hired to develop an OS that will solve this problem and integrate an OS into this computer system hardware to satisfy their needs.

Task:

1. Develop an operating system (OS) tailored for text editing of .txt files.
2. Generate an ISO image for installation purposes.
3. Prepare clear and concise documentation explaining how to use the OS.

Instruction:

1. Enable the OS to create, edit, and save .txt files on the computer storage.
2. Ensure the OS integrates seamlessly with the company's self-built computer hardware.
3. Include a simple console for user interaction.
4. Implement user-friendly process management within the OS.
5. Ensure the computer does not crash upon reboot even if power cuts off suddenly.
6. Incorporate error handling routines to detect and manage process failures effectively.
7. Display appropriate error messages on the console to notify users of any problems.

The work must be performed within 12 hours .
All equipment, tools and materials are provided.

Resources

Tools	HexEditor, Code editor, NASM, SASM, QEMU, QtEMU, Compiler
Equipment	Computer
Materials/ Consumables	Internet

Assessable outcomes	**Assessment criteria (Based on performance criteria)**	**Indicator**	**Observation**		**Marks allocation**
			Yes	**No**	
Learning outcome 1: **Preparing working environment**. (8%)	Tools and equipment are properly selected based on hobby kernel requirements	Tools and equipment are selected			3
	Software tools are properly installed based on installation guide and system specifications	Software tools are installed			4
	Virtual environment Is effectively configured inline with hardware compatibility	Virtual environment Is configured			3
Learning outcome 2: **Implement**	Memory hierarchy and addressing is properly	Memory addressing is Implemented			5

Memory Management. (21%)	implemented based on a specific addressing mode				
	Memory management unit and allocation is properly implemented based on operating system support	Memory allocation is implemented			5
	Layout of process address space and protection is effectively managed based on memory usage optimization	Layout of process address space is managed			5
	Virtual memory and swapping techniques are efficiently applied based on system workload	Swapping techniques are applied			5
	Techniques for reducing fragmentation is properly applied based on memory allocation and management best practices	Techniques for reducing fragmentation is applied			5
Learning outcome 3: **Implement Process Management** (21%)	Process states and scheduling algorithms are properly implemented based on system workload.	Scheduling algorithms are implemented			5
	Inter-Process Communication(IPC) primitives are properly implemented based on process data exchange mechanisms.	Inter-Process Communication (IPC) primitives are implemented			5
	Deadlock handling algorithms are properly applied based on allocated resources.	Deadlock handling algorithms are applied			5
	Thread states and synchronization is properly applied based on synchronization mechanisms	Thread states is applied			3
		Synchronization is applied			2

	Types of Parallelism are properly applied based on the intended performance and efficiency	Parallelism are applied			2
	Lock-Based Concurrency Control mechanism is properly implemented based on data race prevention measures	Lock-Based Concurrency Control mechanism is implemented			3
Learning outcome 4: **Implement persistence management** (17%)	Input/ Output systems are properly applied based on the interrupt or Direct Memory Access (DMA) techniques	Input/ Output systems are applied			10
	basic file systems operations are properly implemented based on system requirements.	Basic file systems operations are implemented			5
	Persistence management is properly implemented based on implemented file systems	Persistence management is implemented			5
Learning outcome 5: **Implement a hobby kernel** (33%)	Target architecture of a kernel is properly selected based on the intended use and hardware they want to support.	Target architecture of a kernel is selected			5
	A simple boot loader is properly written based on target system architecture	Boot loader is written			10
	Interrupt handling is properly implemented based on operating system requirements	Interrupt handling is implemented			10
	A function to display console output on the screen in a kernel is properly written based on display hardware being used.	A function to display console output on the screen in a kernel is written			5

	System call is properly implemented based on operating system kernel architecture.	System call is implemented			10
Total marks		**120**			
Percentage Weightage		**100%**			
Minimum Passing line % (Aggregate): 70%					

References:

1. Stephen, Bigelow J. "What is a kernel?" TechTarget, August 2022, https://www.techtarget.com/searchdatacenter/definition/kernel

2. Leonid, Rosenboim. "What tools do you use to manage operating systems?" LinkedIn, https://www.linkedin.com/advice/0/what-tools-do-you-use-manage-operating-systems

3. Shichao. "Memory Addressing." Shichao's Notes, https://notes.shichao.io/utlk/ch2/

4. Sabela, Ramos, et al. "Cache Line Aware Algorithm Design for Cache-Coherent Architectures." IEEE Xplore, https://ieeexplore.ieee.org/abstract/document/7378320

5. David, Law, et al. "Memory allocation" Harvard University, May 9th , 2005, https://read.seas.harvard.edu/~kohler/class/05s-osp/notes/notes10.html

6. Stephen, Kellett. "Memory Fragmentation, your worst nightmare." Softwareverify, May 11th, 2021, https://www.softwareverify.com/blog/memory-fragmentation-your-worst-nightmare/

7. "Process Management." GeeksforGeeks, December 6th, 2023, https://www.geeksforgeeks.org/introduction-of-process-management/

8. "Deadlock & Deadlock Handling Methods." GeeksforGeeks, February 14th, 2024, https://www.geeksforgeeks.org/introduction-of-process-management/

9. Kızılarslan, Recep. "Fundamental Components of Parallel Programming in Modern Operating Systems." Medium, September 8th 2023, https://medium.com/@recepkzilarslan/parallel-programming-with-modern-operating-systems-i-introduction-modern-os-and-processors-58582d77567e

10. "Mass Storage Structure." GitHub pages, Southeast University, https://csqjxiao.github.io/PersonalPage/csqjxiao_files/OS2015/OSC12.pdf

11. Ananda, Gunawardena. "Systems Programming in C." Carnegie Mellon University, 2009, https://www.cs.cmu.edu/~guna/15-123S11/Lectures/Lecture24.pdf

12. "Example: Integrated file system C functions." IBM, 2009, https://www.ibm.com/docs/en/i/7.4?topic=support-example-integrated-file-system-c-functions

13. Alexandro, Baldassin. "Persistent Memory: A Survery of Programming Support and Implementations." São Paulo State University (Unesp), and Institute of Geosciences and Exact Sciences, Brazil, January, 2021, https://www.dpss.inesc-id.pt/~dcastro/pdf/CSUR2021.pdf

14. "Introduction to Assembly Language." arm Developer, https://developer.arm.com/documentation/den0013/d/Introduction-to-Assembly-Language

15. Theo. "Creating a Kernel from Scratch." Medium, January 25th 2021, https://theogill.medium.com/creating-a-kernel-from-scratch-1a1aa569780f

16. Rosner, Frank. "Writing My Own Boot Loader." DEV.to, March 10th 2021, https://dev.to/frosnerd/writing-my-own-boot-loader-3mld

17. Rathnayaka, Isuruni. "Develop Your Own x86 Operating System(OS) #4." Medium, August 6th 2021, https://isu-rathnayaka.medium.com/develop-your-own-x86-operating-system-os-4-e8479e150451

18. Sugandhi, Abhresh. "System Calls in Operating System: Overview, Types & Examples." upGrad KnowledgeHut, January 3rd 2024, https://www.knowledgehut.com/blog/web-development/system-calls-in-os

19. tutorial point. "Assembly language programming"Assembly Programming Tutorial (tutorialspoint.com)