**A report describing the operating system project**

* **Do you want to build your operating system completely?**
* **Want to know the steps to create an operating system in detail?**
* **If you want to learn something new and fly in the sky of knowledge to open new horizons**
* **Don't go far, your place is here !!!**
* **To answer these questions, follow the next report that tells you what he did, step-by-step**
* **Contents**
* **The first steps in building the system were as follow.**
* **Creating Boot File.**
* **Creating Kernel File.**
* **linker file.**
* **our.iso.**
* **We encountered many difficulties in setting up the operating system.**
* **References used in the project.**
* **The team's narration and ambitions about the operating system he created.**
* **The first steps in building the system were as follow:**
* **Forming a team consisting of four students**
* **Identify a team leader who organizes tasks among team members**
* **Create google site and advertise the team.**

**Website:** [**https://sites.google.com**](https://sites.google.com/view/ahiyooos/home)

* **C Create accounts on social media to our site.**

**(Facebook - twitter- GitHub -Gmail).**

**Trello: https://trello.com**

* **Getting an information and references around our project (videos, articles, books, photos and websites).**
* **Dividing tasks between the team’s consciousness to know each member the task required of him to do a search for them**
* **Download our emulator software from VMware**
* **After analysis we start the first practical task:**
* [**Step 1: Creating Boot File:**](https://app.site123.com/project/step-1-creating-boot-file?w=2684609)
* **Creating Boot File:**

**1- Download the virtual Machine (VMware) software that allows us to download and manipulate virtual systems.**

**2 - Install the Ubuntu system on the virtual machine because it is easy to use and very simple to deal with.**

**3. Download the nasm emulator to execute and compiler programs written in assembly language on comment in term**

**sudo apt-get install nasm**

**4. Use the Assmpliy language because it deals directly with hardware and does not require a compiler and Write the code for the operation of the system (boot), (there is in down).**

* **Features of Assmpliy Language**

**1- Easy to detect errors**

**2 - Each instruction in the language of Assmpliy kissed**

**instruction in the machine language**

**3. Quick implementation of programs**

**5 - install bot needs to be in the form of exe to deal with the program Iso**

**To convert bots to .bin (exe) in nasm emulator, we write the**

**following command**

**nasm filename.asm –o filename. Bin**

**Note : use Linux**

**6 - Download qemu system from terminal**

**sudo apt-get install qemu to run this file on screen**

* **in the previous step (Boot), We have defined a stack of size 1024 bytes and managed by stack Bottom and stack Top identifiers**

**https://www.mediafire.com/file/cujx56soggb16ji/boot.s/file**

**as --32 boot.s -o boot.o**

**Note : There is another code taken from the site to comply with the kernel and linker**

* [**Step 2: Creating Kernel File:**](https://app.site123.com/project/step-1-creating-boot-file?w=2684609)

**Then in \_start, we are storing a current stack pointer, and calling the main function of a kernel (kernel entry) we show that in this step.**

**Alright, writing a kernel from scratch is to print something on screen.**

**So, we have a VMware ...on qemu system.**

**First you need a multiboot bootloader file that instruct the GRUB to load it.**

**Following fields must be defined.**

**Magic: - A fixed hexadecimal number identified by the bootloader as the header(starting point) of the kernel to be loaded.**

**flags: - If bit 0 in the flags word is set, then all boot modules loaded along with the operating system must be aligned on page (4KB) boundaries.**

**checksum: - which is used by special purpose by bootloader and its value must be the sum of magic no and flags.**

**We don't need other information,but for more details**

[**https://www.gnu.org/software/grub/manual/multiboot/multiboot.pdf**](https://www.gnu.org/software/grub/manual/multiboot/multiboot.pdf)

**code kernel**

[**https://www.mediafire.com/file/fhandqdzehy3hkv/kernel.c/file**](https://www.mediafire.com/file/fhandqdzehy3hkv/kernel.c/file)

**to compiler this file :**

**gcc -m32 -c kernel.c -o kernel.o -std=gnu99 -ffreestanding -O1 -Wall -Wextra**

**gcc -m32 -c utils.c -o utils.o -std=gnu99 -ffreestanding -O1 -Wall -Wextra**

**gcc -m32 -c char.c -o char.o -std=gnu99 -ffreestanding -O1 -Wall -Wextra**

**in entry function in file kernel there are print statement put we need linker and some library so, you can see that in next step**

**to display Hello world in your screen you need to linker between Kernel and boot that in the next section.**

* [**Step 3:linker file:**](https://app.site123.com/project/step-1-creating-boot-file?w=2684609)

**As you know, every process consists of different sections such as data, bss, rodata and text.**

**You can see the each sections by compiling the source code without assembling it.**

**And this sections requires a memory to store them, this memory size is provided by the linker image file.**

**Each memory is aligned with the size of each block.**

**It mostly require to link all the object files together to form a final kernel image.**

**Linker image file provides how much size should be allocated to each of the sections.**

**The information is stored in the final kernel image.**

**If you open the final kernel image(.bin file) in hexeditor, you can see lots of 00 bytes.**

**the linker image file consists of an entry point,(in our case it is \_start defined in file boot.S) and sections with size defined in the BLOCK keyword aligned from how much spaced.**

**(linker.ld):**

https://drive.google.com/open?id=1Ry2l9JaXPnwiwXoh9FrN6whrPVL3KLfp

**Now you need a configuration file that instruct the grub to load menu with associated image file**

**(grub.cfg)**

**https://drive.google.com/open?id=1fWT4gs7UOo-jtmPp0KieA706nC64m4O5**

**so i put all things in this page and also there is file called run.sh to follow run steps .**

https://drive.google.com/open?id=1b9\_lUJ3ep6mgrZsNKVMlpOZ7gVrnK6r8

**Someone is wondering what a function file (Kernel.h). :**

**we have a VGA(Visual Graphics Array), a hardware system that controls the display.**

**Here we are using 16 bit vga buffer, on my machine the VGA address is starts at 0xB8000 and 32 bit starts at 0xA0000.**

**An unsigned 16 bit type terminal buffer pointer that points to VGA address.**

**It has 8\*16 pixel font sizeHere we are using 16-bit vga buffer, on my machine the VGA address is starts at 0xB8000 and 32 bit starts at 0xA0000.**

**An unsigned 16-bit type terminal buffer pointer that points to VGA address.**

**It has 8\*16-pixel font size.**

[**http://www.mediafire.com/file/67aesheu789fm1o/linker.ld/file**](http://www.mediafire.com/file/67aesheu789fm1o/linker.ld/file)

**to compiler this file :**

**ld -m elf\_i386 -T linker.ld kernel.o utils.o char.o boot.o -o our.bin –nostdlib**

* [**Step 3:our.iso :**](https://app.site123.com/project/step-1-creating-boot-file?w=2684609)

<https://www.mediafire.com/file/tviexz4vnfuq6zz/our.iso/file>

**to compiler this file :**

**qemu-system-x86\_64 -cdrom our.iso**

* **We encountered many difficulties in setting up the operating system:**

**1. First, at first, the members were faced with how to start and where to start the project**

**2. We didn't know about the way the OS worked, and what would we do? The solution: We searched all sources and the site on how to build an operating system**

**3. How to apply what we do and how to emulate it**

**Solution: We try to do it in VM, watch video and browse the site on how to do it?**

**4. We did not have enough knowledge to write codes in Arabic language, but we got acquainted with some commands and understood what you mean**

**5.I had no difficulty with c but found a difference in writing for example printf and found it writing kernel print.int**

**6.Initially, the system used to print the sentence Hello to the world, but later we learned how to insert numbers into the system with the help of some friends.**

**PS : You can develop and add programs to your operating system ... using the kernel.**

**And some programs were added, such as: Calculator and Tic-Talk game.**

**Finally, we must find the problems, we must search for them, and we will not be disturbed**

* **References used in the project**

|  |  |
| --- | --- |
| video References | <https://www.youtube.com/channel/UCQdZltW7bh1ta-_nCH7LWYw> |
| [**https://www.youtube.com/watch?v=rr-9w2gITDM&t=249s**](https://www.youtube.com/watch?v=rr-9w2gITDM&t=249s) |
| [**https://www.youtube.com/watch?v=Lke3QOytgcQ&list=PLmlvkUN3-1MNKwINqdCDtTdNDjfBmWcZA**](https://www.youtube.com/watch?v=Lke3QOytgcQ&list=PLmlvkUN3-1MNKwINqdCDtTdNDjfBmWcZA) |
| [**https://www.berlios.de/software/mikeos/**](https://www.berlios.de/software/mikeos/) |
| **https://www.youtube.com/watch?v=rxsBghsrvpI** |

|  |  |
| --- | --- |
| Web References | https://www.codeproject.com/Articles/1225196/Create-Your-Own-Kernel-In-C-2 |
| **https://www.codeproject.com/Articles/1225196/Create-Your-Own-Kernel-In-C-2** |
| **https://github.com/alejandrokf21/Kernel-OS/tree/master/CuervoOS%20-%20Final** |
| [**https://www.berlios.de/software/mikeos/**](https://www.berlios.de/software/mikeos/) |
| **https://github.com/alejandrokf21/Kernel-OS/tree/master/CuervoOS%20-%20Final** |

* **The team's narration and ambitions about the operating system he created:**

**1. Add a feature to control the system with the mouse.**

**2.Add a some base classes for a GUI**

**(graphical userinterface) Framework.**

**Website:** [**https://sites.google.com/view/ahiyooos/home**](https://sites.google.com/view/ahiyooos/home)

**Trello:** [**https://trello.com/b/rMMR00YU/operating-system-project**](https://trello.com/b/rMMR00YU/operating-system-project)

**Supervision of Dr. Hazem El-Baz**