Technical University of Republic of Moldova
Chair of Computer Science

REPORT

on OS class

Topic: Intro to operating systems

Done by:	Dvorac Alexei st. gr. FAF-101
Verified by:	Lisnic Andrei

Chişinău – 2012

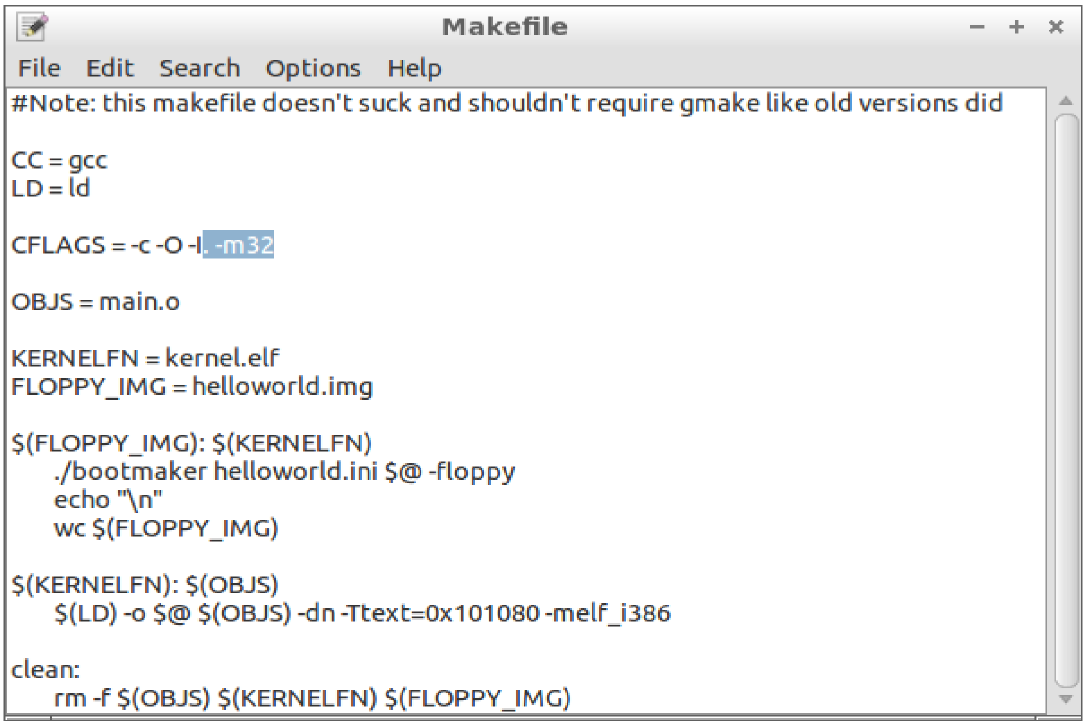
Topic: Intro to operating systems

Goals: Set up build environment.
Launch “Hello World OS” successfully with the message displayed on the screen.

Workflow:

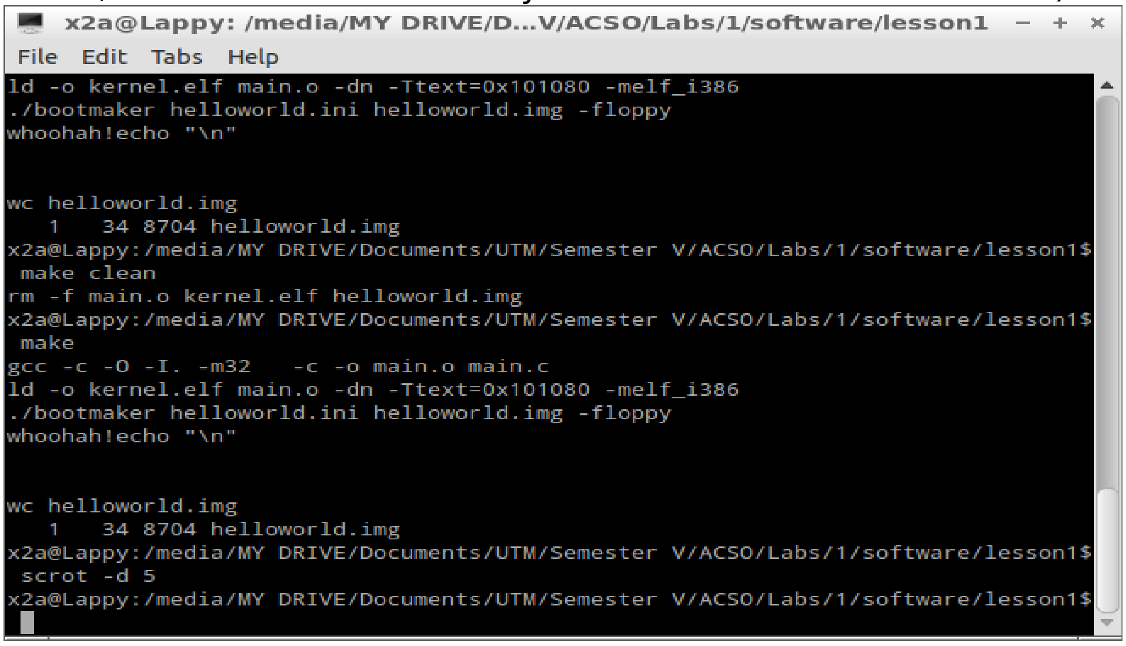
As I am using Linux the process was 2 times easier than for those who use Windows.

I had all the prerequisites installed by default. I just had to deal with the problem that my system was x64. For this it was necessary to add “ -m32” to the CFLAGS value in the Makefile.



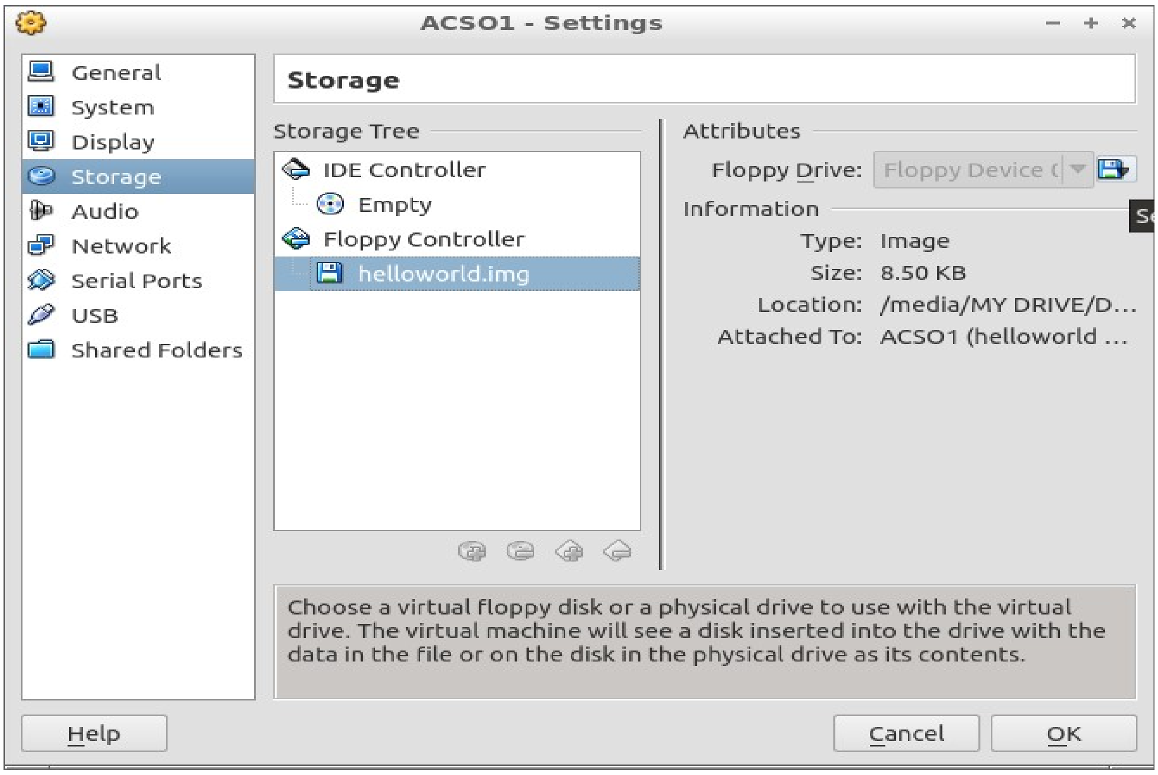
```
#Note: this makefile doesn't suck and shouldn't require gmake like old versions did

CC = gcc
LD = ld

CFLAGS = -c -O -I. -m32

OBJS = main.o

KERNELFN = kernel.elf
FLOPPY_IMG = helloworld.img

$(FLOPPY_IMG): $(KERNELFN)
	./bootmaker helloworld.ini $@ -floppy
	echo "\n"
	wc $(FLOPPY_IMG)

$(KERNELFN): $(OBJS)
	$(LD) -o $@ $(OBJS) -dn -Ttext=0x101080 -melf_i386

clean:
	rm -f $(OBJS) $(KERNELFN) $(FLOPPY_IMG)
```

Then it was necessary to navigate, within the terminal, to the folder with the makefile and compile the program using *make* command(or recompile after some modifications, in this case it is necessary to run *make clean* before *make*).


x2a@Lappy: /media/MY DRIVE/D...V/ACSO/Labs/1/software/lesson1
File Edit Tabs Help

```
ld -o kernel.elf main.o -dn -Ttext=0x101080 -melf_i386
./bootmaker helloworld.ini helloworld.img -floppy
whoohah!echo "\n"


wc helloworld.img
   1   34 8704 helloworld.img
x2a@Lappy:/media/MY DRIVE/Documents/UTM/Semester V/ACSO/Labs/1/software/lesson1$
 make clean
rm -f main.o kernel.elf helloworld.img
x2a@Lappy:/media/MY DRIVE/Documents/UTM/Semester V/ACSO/Labs/1/software/lesson1$
 make
gcc -c -O -I. -m32   -c -o main.o main.c
ld -o kernel.elf main.o -dn -Ttext=0x101080 -melf_i386
./bootmaker helloworld.ini helloworld.img -floppy
whoohah!echo "\n"


wc helloworld.img
   1   34 8704 helloworld.img
x2a@Lappy:/media/MY DRIVE/Documents/UTM/Semester V/ACSO/Labs/1/software/lesson1$
 scrot -d 5
x2a@Lappy:/media/MY DRIVE/Documents/UTM/Semester V/ACSO/Labs/1/software/lesson1$
```

As a result we get a floppy image (helloworld.img) that should be mounted on a virtual machine.


ACSO1 - Settings

Storage

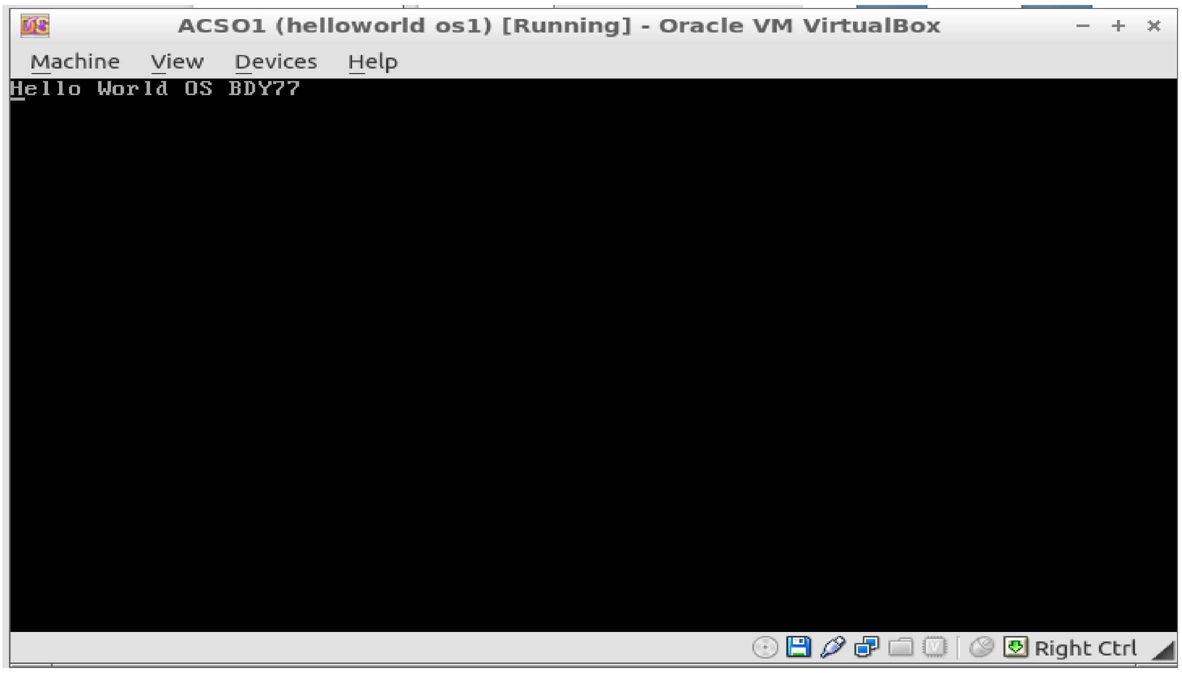
Storage Tree
- IDE Controller
  - Empty
- Floppy Controller
  - helloworld.img

Attributes
Floppy Drive: Floppy Device (

Information
Type: Image
Size: 8.50 KB
Location: /media/MY DRIVE/D...
Attached To: ACSO1 (helloworld ...

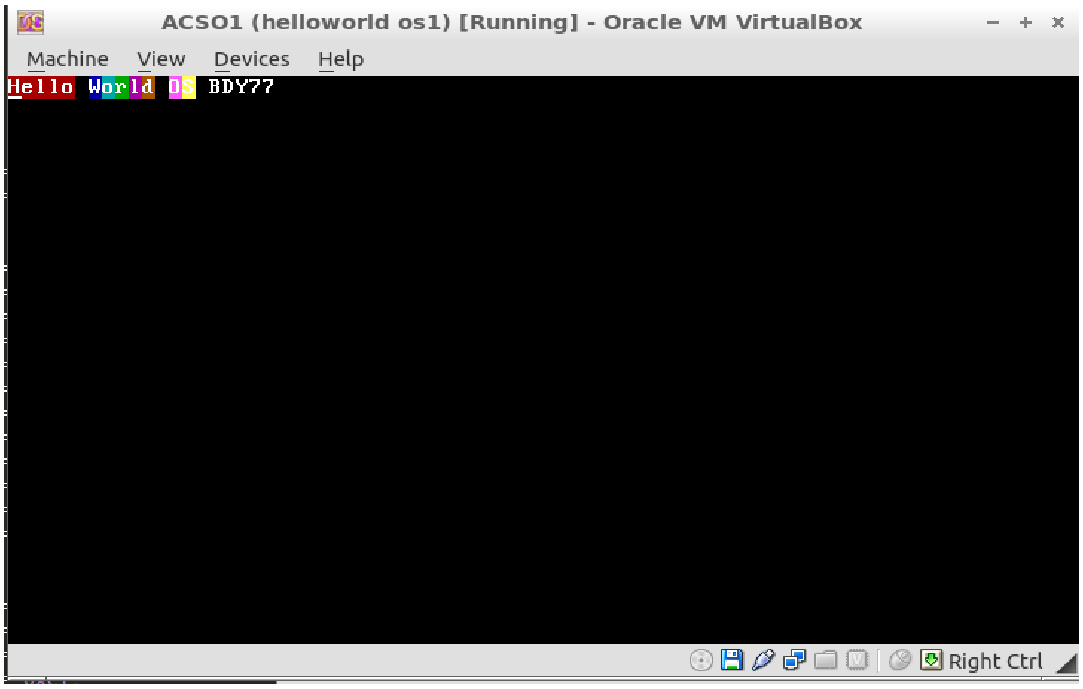
Choose a virtual floppy disk or a physical drive to use with the virtual drive. The virtual machine will see a disk inserted into the drive with the data in the file or on the disk in the physical drive as its contents.

When running the machine the following result can be seen


ACSO1 (helloworld os1) [Running] - Oracle VM VirtualBox

```
Hello World OS BDY77
```

If we add several modifications to the main.c file then we can change the output text, the color of the font and the color of text highliting.



```
Hello World OS BDY77
```

Questions:

<u>- What is the role of an operating system?</u>

The OS provides a way for the user to work with the data presented in a human readable way, not in infinitely long strings of ones and zeros.
It handles the execution of the programs called/run by the user. It should manage the resource usage and allocation to different processes.
OS, also, does manage: input/output, file system, authorisation.

<u>- What is a boot process?</u>

Boot process is the initial set of operations that a computer system performs when electrical power is switched on. The process begins when a computer that has been turned off is re-energized, and ends when the computer is ready to perform its normal operations.
On modern computers this process includes a power-on self-test, locating and initializing peripheral devices, finding and loading an OS.

<u>- How to make a drive bootable?</u>

A bootable device is defined as one that can be read from, and where the last two bytes of the first sector contain the little-endian word AA55h, found as byte sequence 55h, AAh on disk, or where it is otherwise established that the code inside the sector is executable on x86 PCs.

- Why we didn’t use printf in hello world example?

printf could not be used, as the library it is contained by is available within an OS and was not included in our boot.asm file. However, a function that would execute this could've been implemented.

- Provide an elaborate description of what boot.asm and loader.asm does.

The *boot.asm* program was designed and written to be an easier method of using and installing the popular loader.asm code. It is a simple program which gets linked with the loader object module to become a DOS executable file that installs the loader on either the A: or B: floppy disk drive.

It allows a programmer to quickly create a bootable floppy disk which boots to his or her own code.

When you have built and installed the code, you will have created an executable file boot.exe (from boot.asm) which installs the boot loader on a floppy disk. Using this, you will create a floppy disk which contains a bootloader (loader.asm) and an operating system stub (beroset.asm) which you may then replace with your own code.

The *loader.asm* program is a demonstration boot loader which is capable of loading a "home-grown" operating system from either a floppy or hard drive. I have also included a demo target program beroset.asm which just displays a message and halts, but it shows that the program was really loaded into memory. The thing that makes this boot loader different from most is that it loads an ordinary DOS file into memory and is not dependent on the target program being in a particular location on the hard drive or floppy.

The purpose of this program is twofold:

1. to demonstrate some methods for manipulating DOS files without DOS
2. to show a somewhat practical bootloader

Conclusion: This laboratory work was an intro into the operating systems development and allowed me to see the very basic steps needed to make a bootable image.