CS350: Assignment 0 – Basic Setup

Tavian Barnes, Emil Tsalapatis

For this course, you will be writing code for a small operating system named CastorOS. CastorOS is an educational OS that runs on real hardware or in a virtual machine [10]. In the course of the assignments we will write code to boot and run CastorOS, and then add system calls to it to implement fundamental OS functionality for loading programs, managing processes and accessing the file system.

You will be running CastorOS using the QEMU emulator [6] that runs the entire OS inside a userspace process. QEMU emulates [3] real hardware, so the OS runs just as it would on bare metal. QEMU is available on Linux and MacOS, and on Windows through WSL [9].

This assignment will get you familiar with downloading, editing, and submitting the CastorOS code, to prepare you for more difficult future assignments. The code as-is will not compile; it is your job to find the error and make a simple change so it will build successfully. Once you've done that, the OS will be able to start booting, but it will not finish booting successfully yet. Getting the boot process to succeed will be your task for Assignment 1.

1 The CS350 Client

You will use the `client.py` command line tool to interact with the CS350 submission server. The tool is a script that captures the changes to your Git repository, submits your work and allows you to check the status of your submission.

1.1 Setting Up the Submission Client

First, download `client.py` from the course website [2]. To authenticate with the submission server, you'll need a username and a magic string. The username is your 8-character UW ID. You will receive the magic string by email from `cs350-noreply@uwaterloo.ca`. Do not share your magic string with anyone else.

Edit `client.py` directly to set the `STUDENT` variable to your username, and the `MAGIC` variable to your magic string. To test whether you have entered your credentials correctly, run:

```
$ python client.py ping
```

```
CS350 Server is active
$ python client.py status
Last Active Submission: None
Grades: None
```

The `client.py ping` command confirms that the submission server is accessible from your machine. If so, the `client.py status` command ensures you have the proper credentials. If you have properly added your credentials to the client, the server will respond with your last active submission and your grades.

1.2 Downloading Castor OS

Next, download the latest version of the CastorOS source:

```
$ python client.py download
```

The command requests CastorOS's source from the submission server and places it in the current directory. Use `tar` to unpack the source into a `castoros` directory that you will develop your solutions in.

```
$ tar zxvf castoros-latest.tar.gz
```

The source tree includes the userspace, kernel, and tools for the OS. The tree looks as follows:

```
castoros
|--AUTHORS
|--bin
|--build
|--include
|--lib
|--LICENSE
|--pxelinux
|--release
|--sbin
|--SConstruct
|--sys
|--tests
```

The source tree holds the different userspace components and the kernel in separate directories. The kernel is entirely within the `sys/` directory and includes device drivers, system calls and core subsystems. Userspace programs are entirely within the `bin/` and `sbin/` directories. These programs use the system API defined within the headers in `include`. The `lib` directory holds userspace shared libraries like `libc` that implement the userspace part of this API and often interface with the kernel. The `build` directory is initially empty and serves as a destination for the compiled kernel binary and disk image.

2 Getting Started

CastorOS uses the SCons [7] build system to compile the OS and create the runnable image. If you are using `linux.student.cs.uwaterloo.ca`, this is already installed for you. Otherwise, you may have to install it yourself. With SCons installed, the entire compilation process happens by running the `scons` command in the `castoros` source directory.

You must configure SCons to use the LLVM [8] 15 toolchain to compile and link the image. OS images must conform to a very precise layout specification. Different compilers or even versions of the same compiler however generate different layouts for the same code and build arguments. LLVM 15 has been confirmed to produce a correct image so we will be using it for all assignments.

Ensure that SCons uses LLVM 15 by creating a file called `Local.sc` in the `castoros` directory which sets the exact compiler (`CC`) and linker (`LINK`) to use. Your `Local.sc` file should look like this:

```
$ cat Local.sc
CC="clang-15"
LINK="clang-15"
```

The above configuration will work on the `student.cs` environment. The Clang command may have a different name under different OSes (for example, `clang15`). Please consult your OS's documentation for more details if running `clang-15` fails.

Note: mac OS uses Mach-O as it's binary format, and does not use the ELF file format used by COS. To build on a mac OS machine you will need to install Clang from your favorite package manager (assuming it includes `lld` support). Importantly the `LINK` parameter will point to your LLVM's `ld.lld` instead of the clang binary.

For example it might look something like this depending where your clang distribution is installed:

```
CC="/opt/local/libexec/llvm-15/bin/clang"
LINK="/opt/local/libexec/llvm-15/bin/ld.lld"
```

2.1 Fixing the Error

Once the build is configured correctly, there should be a single file with a compiler error:

```
$ scons
scons: Reading SConscript files ...
...
1 error generated.
scons: building terminated because of errors.
```

Open that file and follow the instructions near the line with the error to complete the assignment. You should only have to change one line of code.

2.2 Running CastorOS

A successful build will create a disk image in `build/bootdisk.img` and a kernel image in `build/sys/castor`.

```
$ scons
scons: Reading SConscript files ...
...
scons: done building targets.
$ ls -lh build/bootdisk.img build/sys/castor
-rw-r-----  1 user users  128M Sep 11 16:24 build/bootdisk.img
-rwxr-x---  1 user users  189K Sep 11 16:24 build/sys/castor
```

We start up CastorOS by passing the outputs of the compilation process to QEMU. QEMU uses the disk image to emulate a hard disk. The disk image holds a file system with the userspace utilities, configuration files and user data. The kernel image only includes the kernel and is separate because it must be passed directly to QEMU for the machine to boot.

The kernel image conforms to the Multiboot specification [5] used by QEMU to set up the kernel for execution at boot time. This format is widely used by bootloaders [1]. CastorOS can thus be installed on real hardware using any Multiboot compatible bootloader. You do not need to run CastorOS on bare metal for the assignments.

To boot CastorOS in a QEMU virtual machine, run the following command from a terminal:

```
$ qemu-system-x86_64 \
    -smp cpus=1 \
    -kernel build/sys/castor \
    -hda build/bootdisk.img \
    -nic none \
    -nographic
```

The backslashes (\) escape newlines to allow the command to be split into multiple lines. `-smp cpus=1` tells QEMU to emulate a single CPU for now. `-kernel` tells QEMU which kernel to boot. `-hda` sets up an emulated IDE hard drive with the given image. `-nic none` disables the Network Interface Card, and `-nographic` disables the graphical framebuffer [4]. You will interact with the machine through a serial console instead, directly in your terminal.

The output should look something like this:

```
SeaBIOS (version rel-1.16.2-0-gea1b7a073390-prebuilt.qemu.org)
Booting from ROM..Castor Operating System
Invalid magic number: 0x0
flags = 0x24f
mem_lower = 639KB, mem_upper = 129920KB
boot_device = 0x8000ffff
```

```
cmdline = build/sys/castor mods_count = 0, mods_addr = 0x43b000
mmap_addr = 0x9000, mmap_length = 0xa8
 size = 0x14, base_addr = 0x0, length = 0x9fc00, type = 0x1
 <... omitted...>
Initializing GDT... Done!
Initializing TSS... Done!
Initializing IDT... Done!
Initializing Syscall... Done!
Initializing PMAP ... Done!
Initializing XMEM ... Done!
 <... omitted ...>
loader:   Offset            VAddr FileSize  MemSize
loader: 00000000 0000000000200000 00001744 00001744
loader: AllocMap 0000000000200000 00001744
loader: 00001750 0000000000202750 00004049 00004049
loader: AllocMap 0000000000202000 00004799
loader: 000057a0 00000000002077a0 00000200 000007a8
loader: AllocMap 0000000000207000 00000f48
loader: Jumping to userspace
CPU 0
Interrupt 14 Error Code: 0000000000000004
 <Debug Information>
Entered Debugger!
kdbg>
```

3 Submitting Your Solutions

After completing the assignment, use `git commit` to commit your change to the local git repository. **The commit should only modify the `sys/dev/console.c` file. If it modifies any other files the server will automatically reject the submission.** Next, use `client.py patch` to generate a patch from your local source tree:

```
$ python client.py patch
```

This will create a file called "castoros.patch" and in the current working directory. The last step is to submit the patch to the submission system:

```
$ python client.py submit -a asst0
```

The command sends the patch to the submission server and queues it for evaluation. **The submission server takes about an hour to evaluate submissions. Submitting a new patch overwrites any active submissions without evaluating them.**

The status of your submission can be monitored using the `client.py status` command we saw earlier:

```
$ python client.py status
```

References

[1] Bootloader. https://en.wikipedia.org/wiki/Bootloader, August 2024.

[2] CS 350 - Operating Systems. https://student.cs.uwaterloo.ca/~cs350/F24/assignments/, August 2024.

[3] Emulator. https://en.wikipedia.org/wiki/Emulator, August 2024.

[4] Framebuffer. https://en.wikipedia.org/wiki/Framebuffer, August 2024.

[5] Multiboot Specification. https://www.gnu.org/software/grub/manual/multiboot/multiboot.html, August 2024.

[6] QEMU. https://www.qemu.org/, August 2024.

[7] SCons: A software construction tool. http://scons.org, August 2024.

[8] The LLVM Compiler Infrastructure Project. https://llvm.org, August 2024.

[9] What is the Windows Subsystem for Linux? https://learn.microsoft.com/en-us/windows/wsl/about, August 2024.

[10] What is Virtualization? https://aws.amazon.com/what-is/virtualization, August 2024.