

CS3500 LAB 1

Krutarth Patel EE23B137

7 August 2025

Brief

In this assignment we setup xv6-riscv, a toy operating system. The tasks involved were:

1. Setup Docker
2. Write a program to print out 'Hello < Your Roll No >! Welcome to CS3500'
3. Compile the code as a user program and verify correctness

Details

0.1 Setup Docker

I was using Arch Linux so there was no official documentation on how to install docker on my system. However the latest package was available in the AUR repository and it worked just fine.

0.2 Write a program ...

```
1  .section .data
2  # data section, the string to be printed
3  # is defined here
4  msg:    .asciz "Hello EE23B137! Welcome to CS3500"
5
6  .section .text
7  # code starts here
8  .global main
9
10 main:
11     # syscall: write(fd=1, buf=msg, count=14)
12     li a7, 16          # syscall number for write
13     li a0, 1           # file descriptor 1 (stdout)
14     la a1, msg         # address of the message
15     li a2, 33          # length of the message
16     ecall             # make the syscall
17
18     # syscall: exit(status=0)
19     li a7, 2           # syscall number for exit
20     li a0, 0           # exit code 0
21     ecall             # make the syscall
```

- Figuring out the arguments required for the write() and exit() was tricky. I guessed that it was the same as in Linux and it worked.
- The syscall numbers were different for xv6. I found them in `kernel/syscall.h`

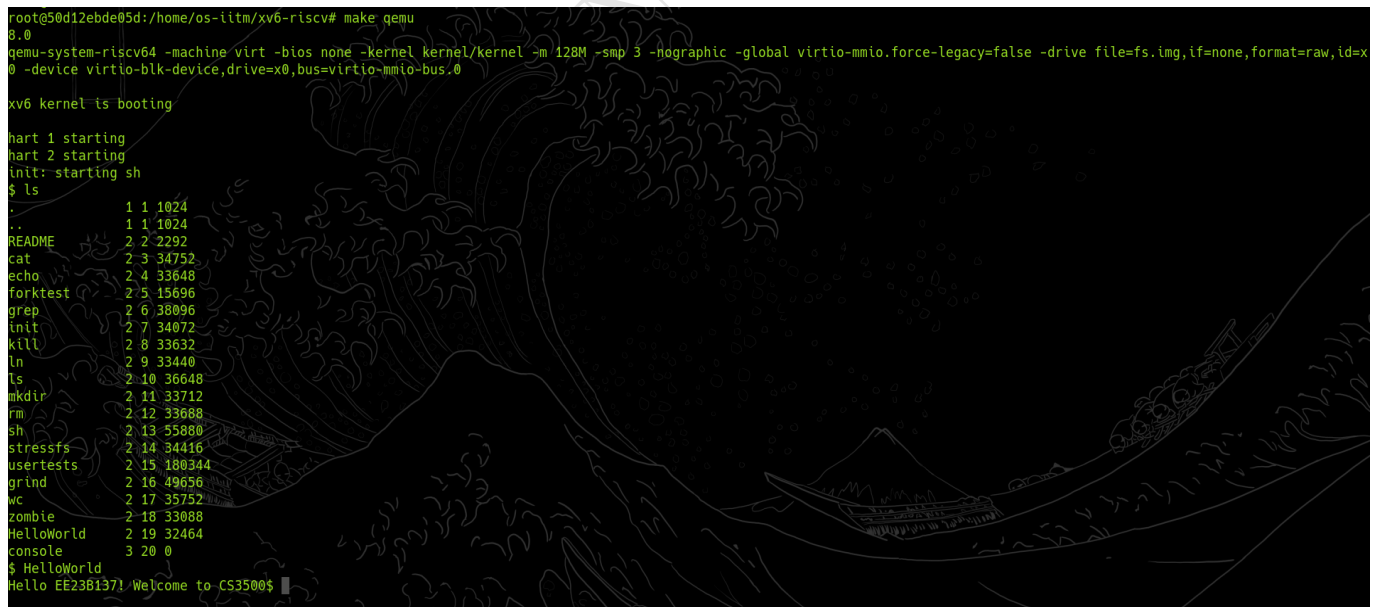
0.3 Compile the code ...

Apply the following patch to Makefile

```
1 diff --git a/Makefile b/Makefile
2 index fab7bc9..347e257 100644
3 --- a/Makefile
4 +++ b/Makefile
5 @@ -109,7 +109,10 @@ $U/usys.S : $U/usys.pl
6
7     $U/usys.o : $U/usys.S
8         $(CC) $(CFLAGS) -c -o $U/usys.o $U/usys.S
9 -
10 +
11 +$U/HelloWorld: $U/HelloWorld.S
12 +     $(CC) $(CFLAGS) -o $U/HelloWorld $U/HelloWorld.S
13 +
14     $U/_forktest: $U/forktest.o $(ULIB)
15         # forktest has less library code linked in - needs to be small
16         # in order to be able to max out the proc table.
17 @@ -142,6 +145,7 @@ UPROGS=\
18     $U/_grind\
19     $U/_wc\
20     $U/_zombie\
21 +     $U/_HelloWorld\
22
23     fs.img: mkfs/mkfs README $(UPROGS)
24         mkfs/mkfs fs.img README $(UPROGS)
```

The whole UPROGS business was difficult to understand. I had to look how other user programs were being compiled and guess my way to this solution.

Proof



```
root@50d12ebde95d:/home/os-lltm/xv6-riscv# make qemu
8.0
qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp 3 -hographic -global virtio-mmio.force-legacy=false -drive file=fs.img,if=none,format=raw,id=x0 -device virtio-blk-device,drive=x0,bus=virtio-mmio-bus.0
xv6 kernel is booting
hart 1 starting
hart 2 starting
init: starting sh
$ ls
.          1 1 1024
..         1 1 1024
README    2 2 2292
cat       2 3 34752
echo      2 4 33648
forktest  2 5 15696
grep      2 6 38096
init      2 7 34072
kill      2 8 33632
ln        2 9 33440
ls        2 10 36648
mkdir     2 11 33712
rm        2 12 33688
sh        2 13 55880
stressfs  2 14 34416
usertests 2 15 180344
grind     2 16 49656
wc        2 17 35752
zombie    2 18 33088
HelloWorld 2 19 32464
console   3 20 0
$ HelloWorld
Hello EE23B137! Welcome to CS3500$
```

How to run

1. Place the HelloWorld.S file inside the user/ directory
2. Apply the above patch to your Makefile
3. Compile and run using `$ make qemu` inside the docker container