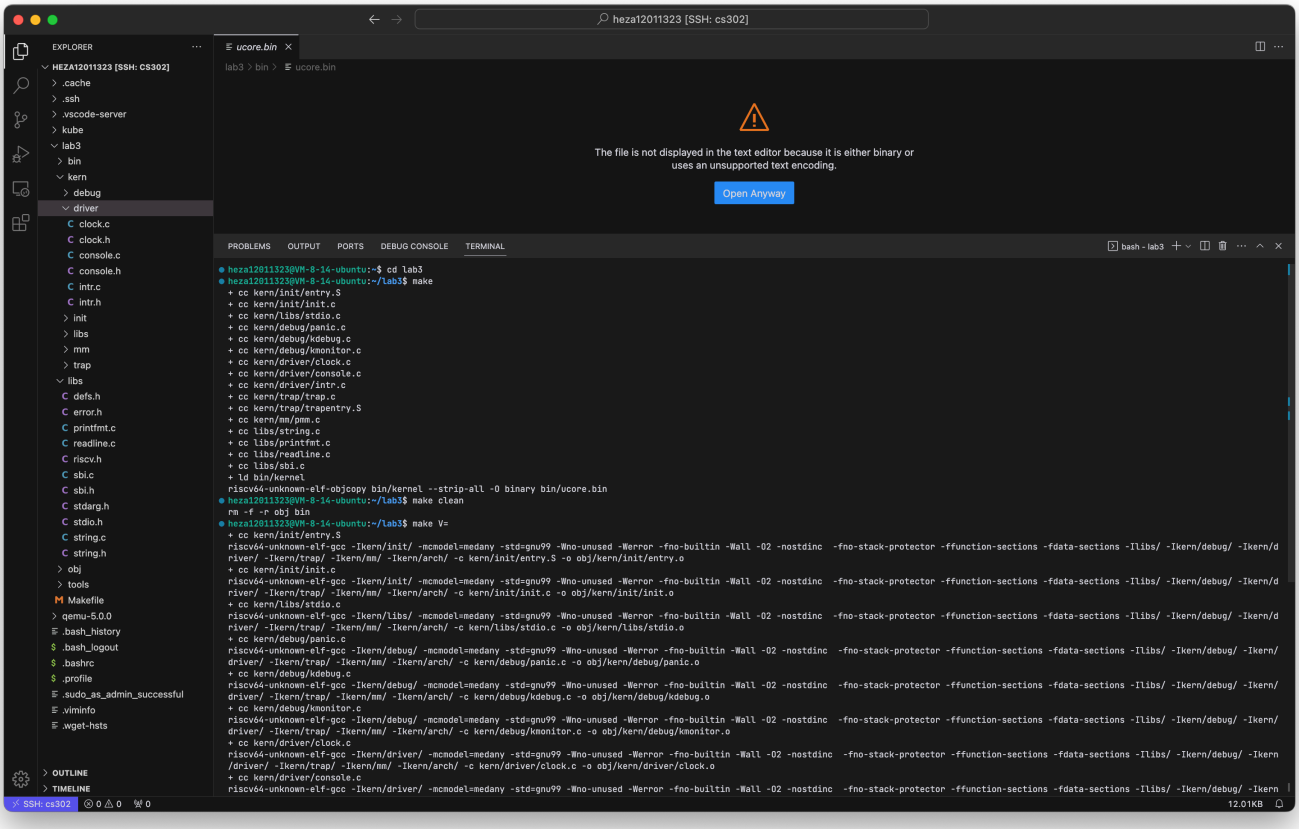
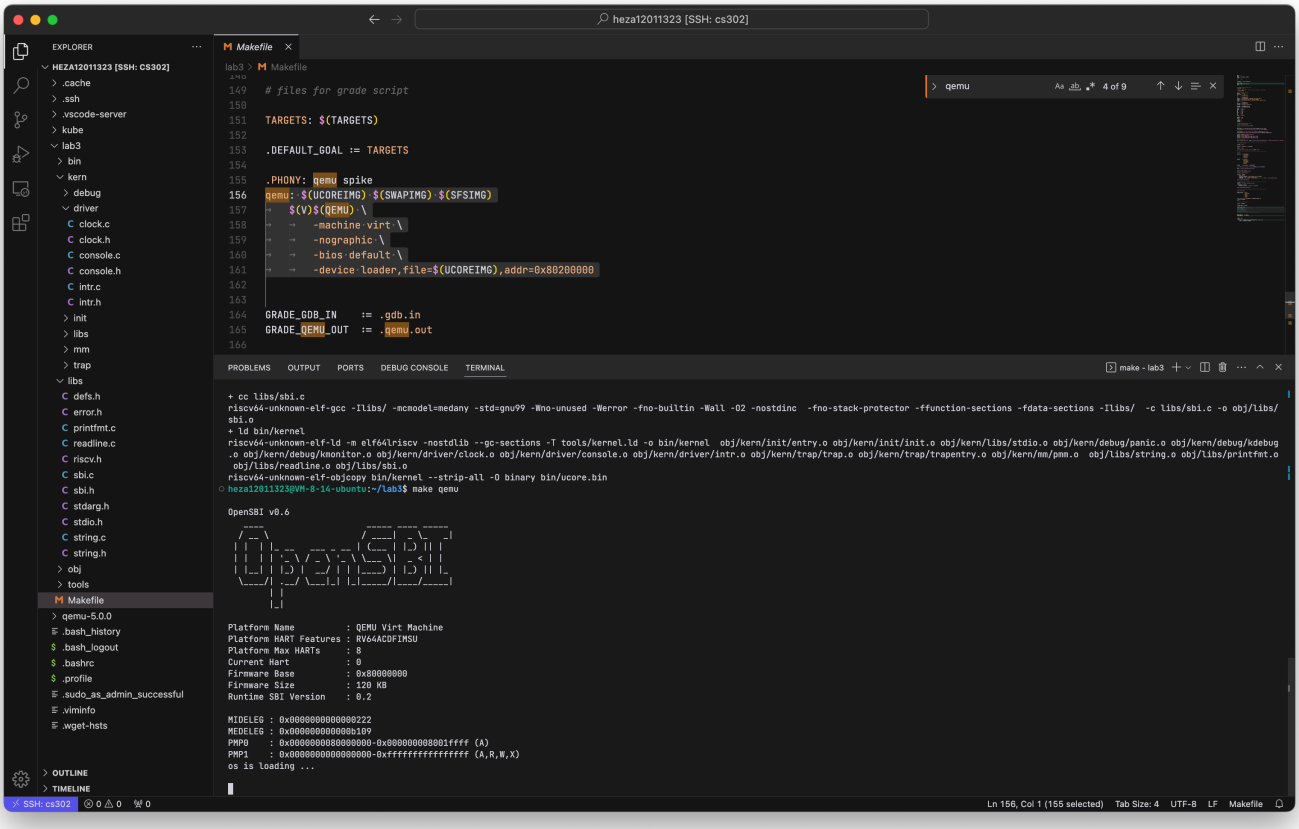
CS302 Lab3 Report

> 何泽安 12011323
>
> 2023.3.1

1. First upload the related code (UCore) into server and extract it. Then directly use the `make` command to generate the virtual disk using the well config Makefile -- related binary files are generated under the `bin` folder.



Then we use the `make qemu` target to start the VM.



In summary, only three commands are needed:

```
cd /path/to/src
make
make qemu
```

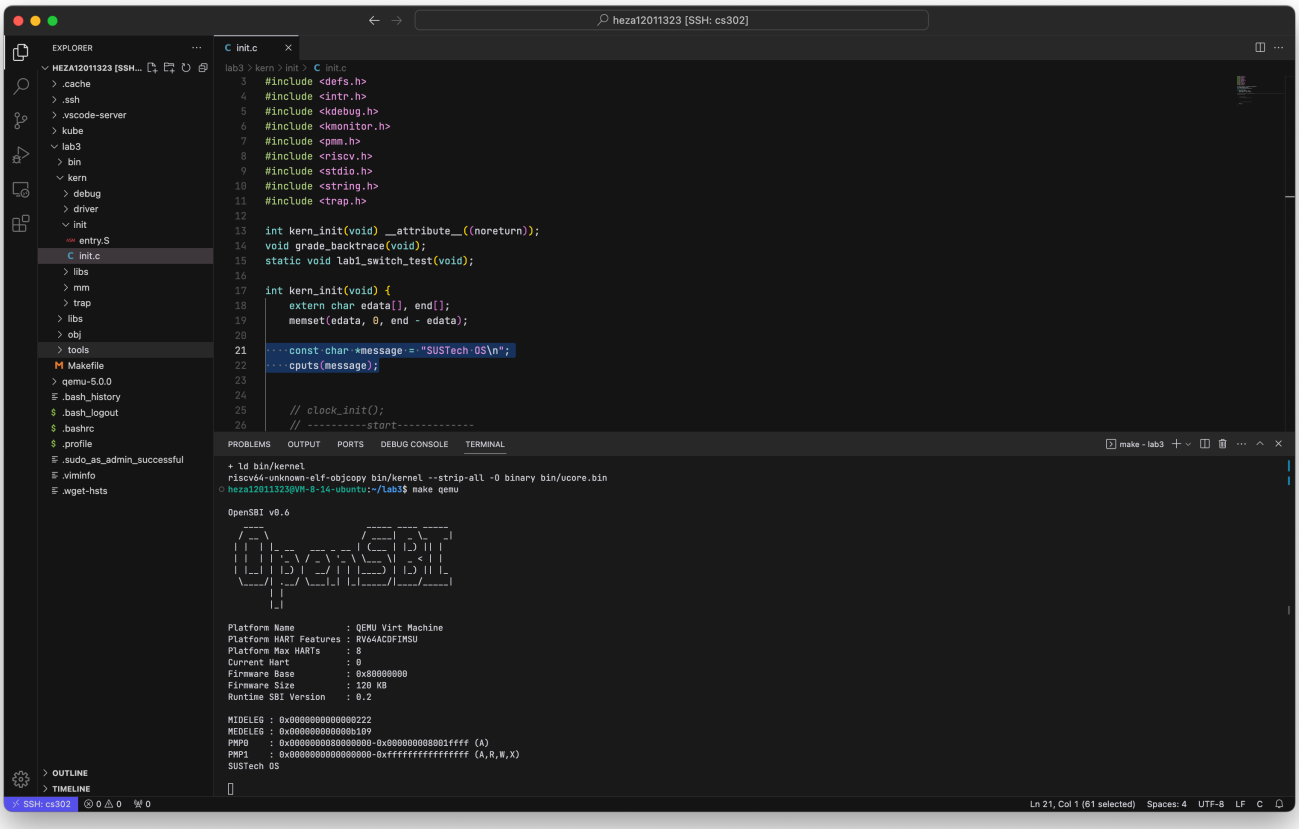
2. A **ELF** file (Executable and Linking Format) has a rather complex structure, including an ELF header, some rudendent debug info, etc, it contains the memory layout in the ELF header, which needs to be parsed by something like a OS, but without that, say, in OpenSBI firmware, the ELF file cannot be directly executed.

   While **BIN** files are much simpler, with raw binary machine codes, it is the final way that the memory looks before the CPU starts executing it, however, since it cannot represent info in the ELF way, it may be much larger than ELF files. Also, bin files can be directly executed by firmwares.

> ELF is a cut-up/compressed version of that, which the CPU/MCU thus can't run directly. [StackOverflow]
>
> Bin file is purely, binary files with bits and bytes which go and locate at the particular address. [StackOverflow]

3. A *linker* helps linking input files (`.o`) into the output file (`elf`). During this step, the linker will face many sections in the input/output files, the *linker script* hereby helps descripting the way to map sections from input files to output files (merging the sections of the target files, relocate the starting/ending addresses of each section), it also rules the memory layout of these sections.

4. Find the file under `kern/init/init.c`, edit the c-string `message`, save, clean the last compiled files, recompile (`make`) and run vm (`make qemu`)



5. We need to first find the header file `libs/stdio.h` and declare the function `int double_puts(const char **str*)`, then implement the function in `kern/libs/stdio.c`. After that we are able to call this function in `init.c`.

heza12011323 [SSH: cs302]

lab3 > kern > init > init.c

```c
#include <trap.h>

int kern_init(void) __attribute__((noreturn));
void grade_backtrace(void);
static void lab1_switch_test(void);

int kern_init(void) {
    extern char edata[], end[];
    memset(edata, 0, end - edata);

    const char *message = "SUSTech OS\n";
    cputs(message);

    const char *iloveos = "ILOVEOS";
    double_puts(iloveos);

    // clock_init();
    // ----------start-------------



    // -----------end--------------
```

lab3 > kern > libs > stdio.c

```c
    char c;
    while ((c = *str++) != '\0') {
        cputch(c, &cnt);
    }
    cputch('\n', &cnt);
    return cnt;
}

// add
int double_puts(const char *str) {
    int cnt = 0;
    char c;
    while ((c = *str++) != '\0') {
        cputch(c, &cnt);
        cputch(c, &cnt);
    }
    cputch('\n', &cnt);
    return cnt;
}

/* getchar - reads a single non-zero character from sta
int getchar(void) {
    int c;
    while ((c = cons_getc()) == 0) /* do nothing */;
```

lab3 > libs > stdio.h

```c
#ifndef __LIBS_STDIO_H__
#define __LIBS_STDIO_H__

#include <defs.h>
#include <stdarg.h>

/* kern/libs/stdio.c */
int cprintf(const char *fmt, ...);
int vcprintf(const char *fmt, va_list ap);
void cputchar(int c);
int cputs(const char *str);
int double_puts(const char *str);  // add
int getchar(void);

/* kern/libs/readline.c */
char *readline(const char *prompt);

/* libs/printfmt.c */
void printfmt(void (*putch)(int, void *), void *putdat,     t
void vprintfmt(void (*putch)(int, void *), void *putdat     st
int snprintf(char *str, size_t size, const char *fmt, .
int vsnprintf(char *str, size_t size, const char *fmt,      st

#endif /* !__LIBS_STDIO_H__ */
```

PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

```
riscv64-unknown-elf-objcopy bin/kernel --strip-all -O binary bin/ucore.bin
heza12011323@VM-8-14-ubuntu:~/lab3$ make qemu

OpenSBI v0.6
   ____                    _____ ____ _____
  / __ \                  / ____|  _ \_   _|
 | |  | |_ __   ___ _ __ | (___ | |_) || |
 | |  | | '_ \ / _ \ '_ \ \___ \|  _ < | |
 | |__| | | | |  __/ | | |____) | |_) || |_
  \____/|_| |_|\___|_| |_|_____/|____/_____|
        | |
        |_|

Platform Name          : QEMU Virt Machine
Platform HART Features : RV64ACDFIMSU
Platform Max HARTs     : 8
Current Hart           : 0
Firmware Base          : 0x80000000
Firmware Size          : 120 KB
Runtime SBI Version    : 0.2

MIDELEG : 0x0000000000000222
MEDELEG : 0x0000000000000b109
PMP0    : 0x0000000080000000-0x000000008001ffff (A)
PMP1    : 0x0000000000000000-0xffffffffffffffff (A,R,W,X)
SUSTech OS

ILLLOOVVEEOOSS
```

SSH: cs302 ⊗ 0 ⚠ 0 0 Ln 22, Col 20 Spaces: 4 UTF-8 LF C