

- PA4-2 实验报告
 - 实验代码、重点问题以及关键结果
 - 1 完成串口的模拟
 - 2 通过硬盘加载程序
 - 3 完成键盘的模拟
 - 4 实现VGA的MMIO
 - 运行结果
 - 思考题

PA4-2 实验报告

211180074 彭安澜

2024 年 6 月 30 日

实验代码、重点问题以及关键结果

本次实验中主要完成了以下内容的代码：

1 完成串口的模拟

1. 在 `include/config.h` 中定义宏 `HAS_DEVICE_SERIAL` 并 `make clean`;
2. 实现 `in` 和 `out` 指令;
 - 此处注意不需要实现 `in` 和 `out` 以立即数 `imm8` 为操作数的两条指令，各自只要实现 `_v` 和 `_b` 的两条指令即可
 - 另外注意端口号从 `0-65535`，总共为 `16` 位，而每个端口对应的数据为 `8` 位，因此在读端口号以及向端口读或写数据时需要注意数据宽度。
3. 实现 `serial_printc()` 函数;
 - 本质就是 `printf` 函数将字符打印在终端来模拟串口的输出，但要换成 `out_byte` 来实现。
4. 运行 `hello-inline` 测试用例，对比实现串口前后的输出内容的区别。

- 。模拟串口前，会看到红色的nemu trap output提示输出：

The screenshot shows a terminal window with a dark sidebar on the left containing navigation options like 'All Vaults', 'Hosts', 'SFTP', 'Port Forwarding', 'Snippets', 'ICS', 'History', and 'Show more...'. The main terminal area displays the following text:

```
nemu trap output: [src/main.c,82,init_cond] {kernel} Hello, NEMU world!
nemu trap output: [src/elf/elf.c,29,loader] {kernel} ELF loading from ram disk.
nemu: HIT GOOD TRAP at eip = 0x080490d8
NEMU2 terminated
./nemu/nemu --autorun --testcase test-float --kernel
NEMU load and execute img: ./kernel/kernel.img elf: ./testcase/bin/test-float
nemu trap output: [src/main.c,82,init_cond] {kernel} Hello, NEMU world!
nemu trap output: [src/elf/elf.c,29,loader] {kernel} ELF loading from ram disk.
nemu: HIT BAD TRAP at eip = 0x080490bd
NEMU2 terminated
make-[1]: Leaving directory '/home/pa211180074/pa_nju'
./nemu/nemu --autorun --testcase hello-inline --kernel
NEMU load and execute img: ./kernel/kernel.img elf: ./testcase/bin/hello-inline
nemu trap output: [src/main.c,82,init_cond] {kernel} Hello, NEMU world!
nemu trap output: [src/elf/elf.c,29,loader] {kernel} ELF loading from ram disk.
nemu trap output: Hello, world!
nemu: HIT GOOD TRAP at eip = 0x08049023
NEMU2 terminated
pa211180074@l8b2c1c2e89e:~/pa_nju$ make submit_pa-4-1
-----make-----
1719637648831
Sat Jun 29 13:07:28 CST 2024
scripts/submit pa-4-1 211180074
(1/5) Confirm Submission

Terms to submit:
  1. I take full responsibility to the code being submitted
     * I have protected my own code and have not released it to any unauthorized third-party
     * I have properly configured the project with respect to the stage, special attention to:
         > include/config.h
         > kernel/Makefile
  2. I have completed my assignment by myself, and have not attempted to hack the tracking or submission processes
     * I will cooperate in any further investigation if required
     * I am aware that there will be heavy punishments if I cheated

Agree the above terms, and confirm to submit PA-4-1 for 211180074 [yes/no]
(default yes, input your choice and press ENTER):yes
(2/5) Check for Integrity ... Pass
(3/5) Re-Build Project ... Pass
(4/5) Execute Tests ... Complete
(5/5) Pack Project ... Pass
Submission sequence completed, you may proceed to:
  1. Submit your report to CSLabCMS if required
  2. Attend our survey if you would like to
pa211180074@l8b2c1c2e89e:~/pa_nju$
```

- 。模拟串口后，会看到提示消失：

The screenshot shows a terminal window with a dark sidebar on the left containing navigation options like 'All Vaults', 'Hosts', 'SFTP', 'Port Forwarding', 'Snippets', 'ICS', 'History', and 'Show more...'. The main terminal area displays the following text:

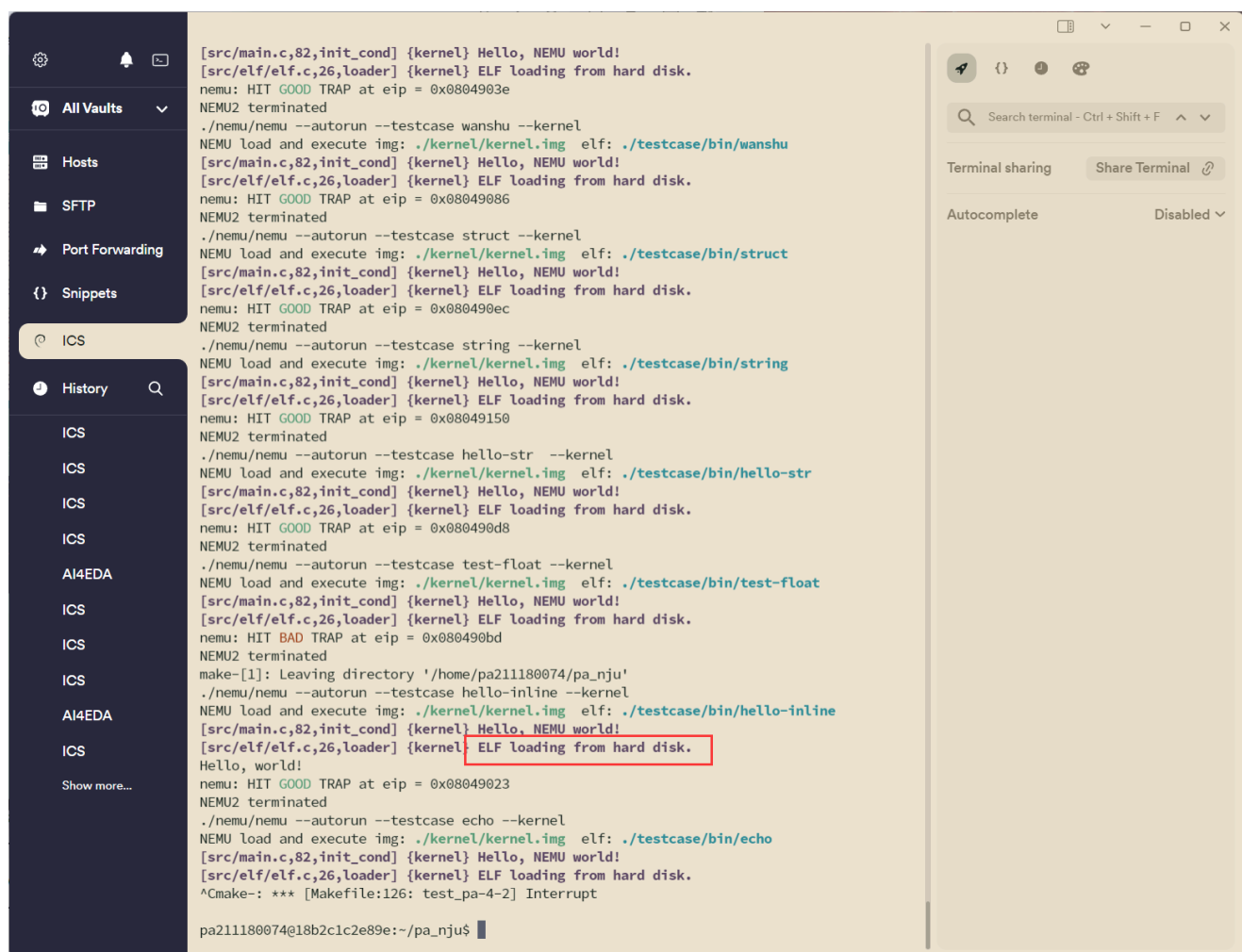
```
./nemu/nemu --autorun --testcase sum --kernel
NEMU load and execute img: ./kernel/kernel.img elf: ./testcase/bin/sum
[src/main.c,82,init_cond] {kernel} Hello, NEMU world!
[src/elf/elf.c,29,loader] {kernel} ELF loading from ram disk.
nemu: HIT GOOD TRAP at eip = 0x0804903e
NEMU2 terminated
./nemu/nemu --autorun --testcase wanshu --kernel
NEMU load and execute img: ./kernel/kernel.img elf: ./testcase/bin/wanshu
[src/main.c,82,init_cond] {kernel} Hello, NEMU world!
[src/elf/elf.c,29,loader] {kernel} ELF loading from ram disk.
nemu: HIT GOOD TRAP at eip = 0x08049086
NEMU2 terminated
./nemu/nemu --autorun --testcase struct --kernel
NEMU load and execute img: ./kernel/kernel.img elf: ./testcase/bin/struct
[src/main.c,82,init_cond] {kernel} Hello, NEMU world!
[src/elf/elf.c,29,loader] {kernel} ELF loading from ram disk.
nemu: HIT GOOD TRAP at eip = 0x080490ec
NEMU2 terminated
./nemu/nemu --autorun --testcase string --kernel
NEMU load and execute img: ./kernel/kernel.img elf: ./testcase/bin/string
[src/main.c,82,init_cond] {kernel} Hello, NEMU world!
[src/elf/elf.c,29,loader] {kernel} ELF loading from ram disk.
nemu: HIT GOOD TRAP at eip = 0x08049150
NEMU2 terminated
./nemu/nemu --autorun --testcase hello-str --kernel
NEMU load and execute img: ./kernel/kernel.img elf: ./testcase/bin/hello-str
[src/main.c,82,init_cond] {kernel} Hello, NEMU world!
[src/elf/elf.c,29,loader] {kernel} ELF loading from ram disk.
nemu: HIT GOOD TRAP at eip = 0x080490d8
NEMU2 terminated
./nemu/nemu --autorun --testcase test-float --kernel
NEMU load and execute img: ./kernel/kernel.img elf: ./testcase/bin/test-float
[src/main.c,82,init_cond] {kernel} Hello, NEMU world!
[src/elf/elf.c,29,loader] {kernel} ELF loading from ram disk.
nemu: HIT BAD TRAP at eip = 0x080490bd
NEMU2 terminated
make-[1]: Leaving directory '/home/pa211180074/pa_nju'
./nemu/nemu --autorun --testcase hello-inline --kernel
NEMU load and execute img: ./kernel/kernel.img elf: ./testcase/bin/hello-inline
[src/main.c,82,init_cond] {kernel} Hello, NEMU world!
[src/elf/elf.c,29,loader] {kernel} ELF loading from ram disk.
Hello, world!
nemu: HIT GOOD TRAP at eip = 0x08049023
NEMU2 terminated
./nemu/nemu --autorun --testcase echo --kernel
NEMU load and execute img: ./kernel/kernel.img elf: ./testcase/bin/echo
[src/main.c,82,init_cond] {kernel} Hello, NEMU world!
[src/elf/elf.c,29,loader] {kernel} ELF loading from ram disk.
```

2 通过硬盘加载程序

1. 在 `include/config.h` 中定义宏 `HAS_DEVICE_ID` 并 `make clean`;
2. 修改 Kernel 中的 `loader()`，使其通过 `ide_read()` 和 `ide_write()` 接口实现从模拟硬盘加载用户程序；
 - 。这一部分手册提示的不多，需要自己理解代码并做出修改，源代码只在开头获取 elf 头使用了 `#ifdef HAS_DEVICE_ID` 并给出从硬盘读数据的例子，实际上除了读 elf 头的操作要做修改，后续读 elf 文件，将每一段（segment）加载的步骤，也要做修改，使用 `#ifdef HAS_DEVICE_ID`，并调用 `ide_read` 对 elf 文件进行读操作：

```
#ifdef IA32_PAGE
uint32_t paddrBase = mm_malloc(ph->p_vaddr, ph->p_memsz);
for (addr_shift = 0; addr_shift < ph->p_filesz; addr_shift++)
{
    // vaddr_write(ph->p_vaddr+addr_shift, 0, 1, vaddr_read(ph->p_offset+addr_shift, 0, 1));
    pdata = (void *)ph->p_offset + addr_shift;
    data = *pdata;
    pdata = (void *)paddrBase + addr_shift;
    *pdata = data;
}
for (addr_shift = ph->p_filesz; addr_shift < ph->p_memsz; addr_shift++)
{
    // vaddr_write(ph->p_vaddr+addr_shift, 0, 1, 0);
    pdata = (void *)paddrBase + addr_shift;
    *pdata = 0;
}
```

3. 通过 `make test_pa-4-2` 执行测试用例，验证加载过程是否正确。正确加载，观察到 elf 文件不再从 ram disk 加载，而是从 hard disk（硬盘）加载。

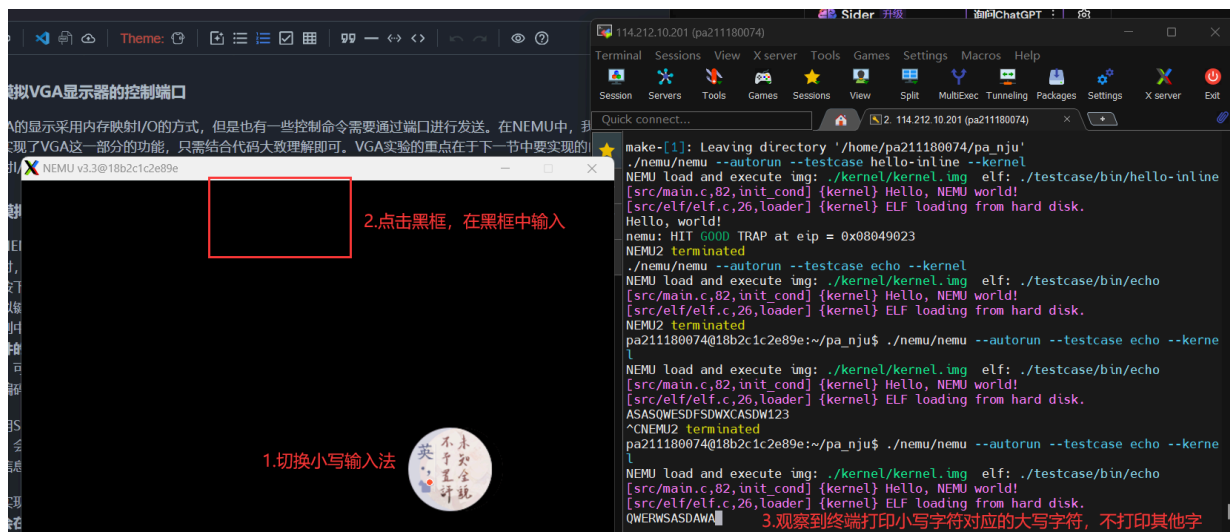


```
[src/main.c,82,init_cond] {kernel} Hello, NEMU world!
[src/elf/elf.c,26,loader] {kernel} ELF loading from hard disk.
nemu: HIT GOOD TRAP at eip = 0x0804903e
NEMU2 terminated
./nemu/nemu --autorun --testcase wanshu --kernel
NEMU load and execute img: ./kernel/kernel.img elf: ./testcase/bin/wanshu
[src/main.c,82,init_cond] {kernel} Hello, NEMU world!
[src/elf/elf.c,26,loader] {kernel} ELF loading from hard disk.
nemu: HIT GOOD TRAP at eip = 0x08049086
NEMU2 terminated
./nemu/nemu --autorun --testcase struct --kernel
NEMU load and execute img: ./kernel/kernel.img elf: ./testcase/bin/struct
[src/main.c,82,init_cond] {kernel} Hello, NEMU world!
[src/elf/elf.c,26,loader] {kernel} ELF loading from hard disk.
nemu: HIT GOOD TRAP at eip = 0x080490ec
NEMU2 terminated
./nemu/nemu --autorun --testcase string --kernel
NEMU load and execute img: ./kernel/kernel.img elf: ./testcase/bin/string
[src/main.c,82,init_cond] {kernel} Hello, NEMU world!
[src/elf/elf.c,26,loader] {kernel} ELF loading from hard disk.
nemu: HIT GOOD TRAP at eip = 0x08049150
NEMU2 terminated
./nemu/nemu --autorun --testcase hello-str --kernel
NEMU load and execute img: ./kernel/kernel.img elf: ./testcase/bin/hello-str
[src/main.c,82,init_cond] {kernel} Hello, NEMU world!
[src/elf/elf.c,26,loader] {kernel} ELF loading from hard disk.
nemu: HIT GOOD TRAP at eip = 0x080490d8
NEMU2 terminated
./nemu/nemu --autorun --testcase test-float --kernel
NEMU load and execute img: ./kernel/kernel.img elf: ./testcase/bin/test-float
[src/main.c,82,init_cond] {kernel} Hello, NEMU world!
[src/elf/elf.c,26,loader] {kernel} ELF loading from hard disk.
nemu: HIT BAD TRAP at eip = 0x080490bd
NEMU2 terminated
make[1]: Leaving directory '/home/pa211180074/pa_nju'
./nemu/nemu --autorun --testcase hello-inline --kernel
NEMU load and execute img: ./kernel/kernel.img elf: ./testcase/bin/hello-inline
[src/main.c,82,init_cond] {kernel} Hello, NEMU world!
[src/elf/elf.c,26,loader] {kernel} ELF loading from hard disk.
Hello, world!
nemu: HIT GOOD TRAP at eip = 0x08049023
NEMU2 terminated
./nemu/nemu --autorun --testcase echo --kernel
NEMU load and execute img: ./kernel/kernel.img elf: ./testcase/bin/echo
[src/main.c,82,init_cond] {kernel} Hello, NEMU world!
[src/elf/elf.c,26,loader] {kernel} ELF loading from hard disk.
^Cmake-: *** [Makefile:126: test_pa-4-2] Interrupt

pa211180074@18b2c1c2e89e:~/pa_nju$
```

3 完成键盘的模拟

1. 在 `include/config.h` 中定义宏 `HAS_DEVICE_KEYBOARD` 并 `make clean`;
2. 通过 `make test_pa-4-2` 运行 `echo` 测试用例；（可以通过关闭窗口或在控制台 `Ctrl-c` 的方式退出 `echo`）
 - 注意此时通过SDL调用键盘功能，必须有窗口，因此此时需要使用支持x11的 `mobaxterm` 等终端运行测试样例，否则终端界面可能发生错误。
 - 运行效果如下：

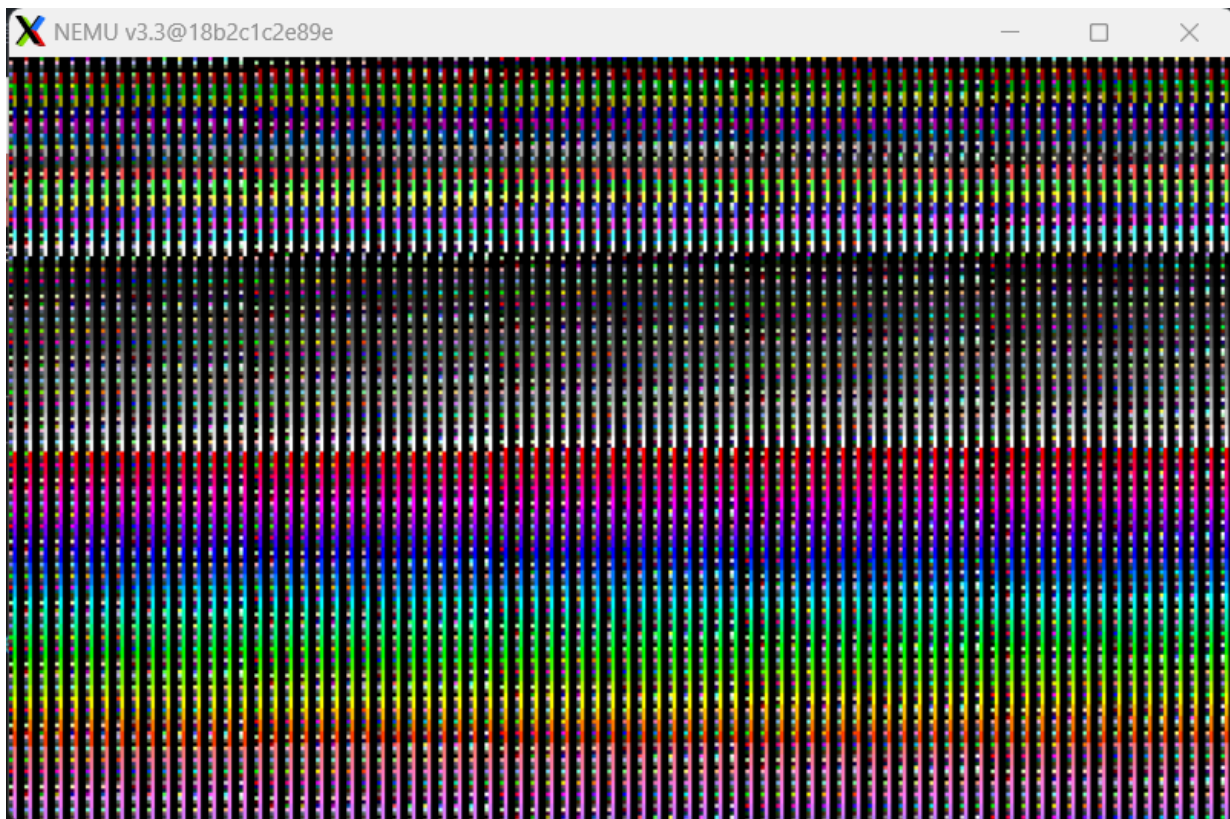


- 注意如果点击终端，则还能输入，且此时的小写转大写规则不生效，此时不是 `nemu` 的键盘在起作用，而是在终端中直接输入并显示字符。

4 实现VGA的MMIO

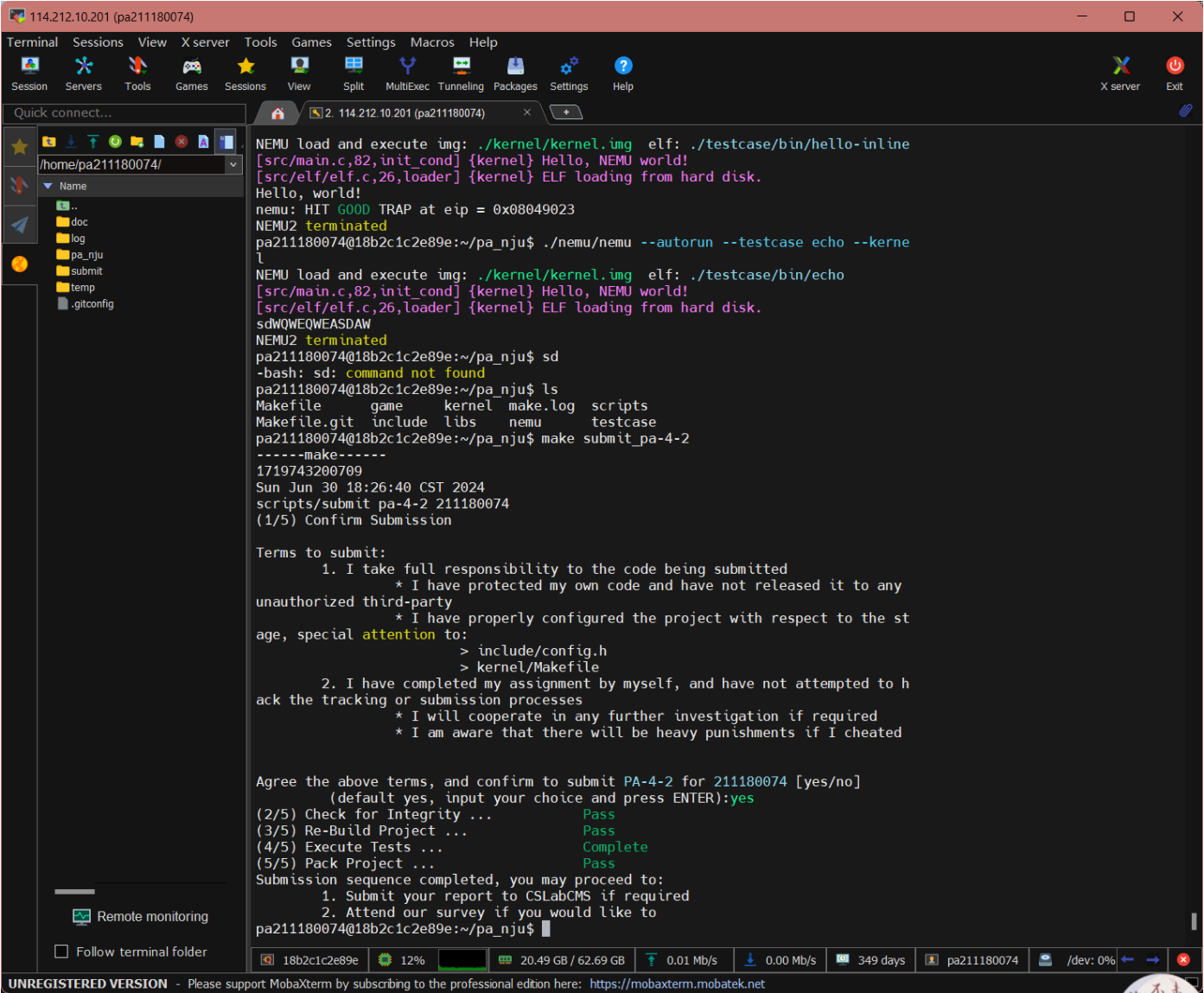
1. 在 `include/config.h` 中定义宏 `HAS_DEVICE_VGA`;
2. 在 `nemu/src/memory/memory.c` 中添加 `mm_io` 判断和对应的读写操作；
 - 此处注意不是在 `memory.c` 中实现 `is_mmio()` 等函数（`mmio` 部分已经在 `nemu/src/device/io/mm_io.c` 中被布置好了），重点是修改 `vaddr_read` 和 `vaddr_write` 函数，使得在访问内存区间时，实际上是访问显存 `io` 设备。
3. 在 `kernel/src/memory/vmem.c` 中完成显存的恒等映射；
 - 也就是在页表中增加几项，使得虚拟地址 `0xa0000` 到 `0xa0000+SCR_SIZE` 映射到物理地址 `0xa0000` 到 `0xa0000+SCR_SIZE`；注意按页对齐，二级页表，页目录项和页表项都要完成，用定义好的宏。
4. 通过 `make test_pa-4-2` 执行测试用例，观察输出测试颜色信息，并通过 `video_mapping_read_test()`。

- 获得测试颜色信息:



运行结果

执行 `make test_pa-4-2`，通过全部测试案例，并完成提交：



思考题

针对echo测试用例，在实验报告中，结合代码详细描述：

- 注册监听键盘事件是怎么完成的？
 - 首先建立了端口映射，将端口号0x60分配给了键盘，并指定了相应的端口处理程序；这样从指令in到pio_read就能一路访问端口数据；
 - 然后设计端口处理程序，能够调用对应的设备驱动程序，将端口接受到的信息按照约定返回；
 - 最后还要创建模拟设备，按照约定与驱动程序交互，同时从用户真实的键盘设备接受信息（主要依靠窗口）。
- 从键盘按下一个键到控制台输出对应的字符，系统的执行过程是什么？如果涉及与之前报告重复的内容，简单引用之前的内容即可。
 - 首先通过SDL注册监听键盘事件；

- 检测到键盘事件时，引起中断，在CPU每次执行exec函数并检测中断时被检测到，CPU转向处理中断请求；
- CPU执行异常控制流，需要保护现场、根据中断号执行中断服务程序、恢复现场，最后回到正常控制流继续执行，此部分与PA4-1内容一致。