

PA 4-2 实验报告

221220085 时昌军

一、实验目的

我们要在模拟器中增加与外部设备进行I/O的功能。如此我们的模拟器就能够实现包括键盘输入、屏幕输出等功能，能够与用户互动起来，完成除了运算以外更加丰富的功能。

二、实验过程

§4-2.3.1 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;
3. 实现 `serial_printc()` 函数;
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。

区别：运行结果没有NEMU trap out 字样。

§4-2.3.2 通过硬盘加载程序

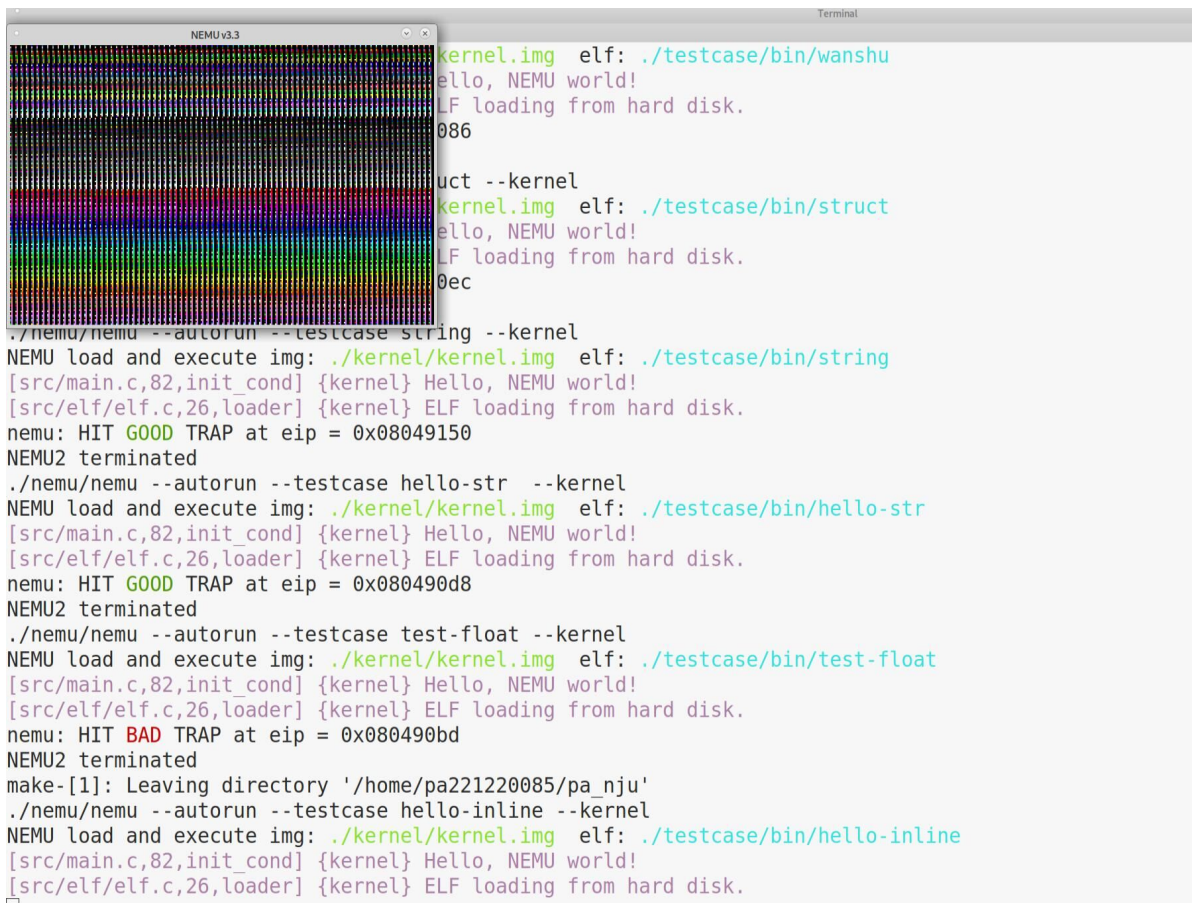
1. 在 `include/config.h` 中定义宏 `HAS_DEVICE_ID` 并 `make clean`;
2. 修改Kernel中的 `loader()`，使其通过 `ide_read()` 和 `ide_write()` 接口实现从模拟硬盘加载用户程序;
3. 通过 `make test_pa-4-2` 执行测试用例，验证加载过程是否正确。

§4-2.3.3 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`）

§4-2.3.4 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;
3. 在 `kernel/src/memory/vmem.c` 中完成显存的恒等映射;
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()`。



```
NEMU v3.3
kernel.img elf: ./testcase/bin/wanshu
Hello, NEMU world!
ELF loading from hard disk.
086

uct --kernel
kernel.img elf: ./testcase/bin/struct
Hello, NEMU world!
ELF loading from hard disk.
0ec

./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/pa221220085/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.
```

三、思考题

针对echo测试用例，在实验报告中，结合代码详细描述：

1. 注册监听键盘事件是怎么完成的？

答：开启 `HAS_DEVICE_KEYBOARD` 后，在 `testcase/srt/echo.c` 中，main函数通过调用 `add_irq_handler`，该函数执行 `int 0x80`，通过系统调用陷入内核态并执行 `do_syscall()` 函数。在 `do_syscall(TrapFrame *tf)` 函数中，由于 `eax=0`，会调用 `add_irq_handle(tf->ebx, (void *)tf->ecx)` 函数，将 `IRQ_t` 类型存入 `handle` 数组中，从而完成注册监听键盘事件。

2. 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。

答：对键盘展开模拟时，键盘事件首先在 `nemu/src/device/sdl.c` 中由 `NEMU_SDL_Thread()` 线程捕获。NEMU捕获两类事件：键盘按下和抬起。当检测到相应事件后，将对应键的扫描码作为参数传送给 `keyboard.c` 中的模拟键盘函数。模拟键盘缓存扫描码，并通过中断请求的方式通知CPU有按键或抬起的事件，键盘的中断请求号为1。CPU收到中断请求后调用Kernel的中断响应程序。在响应程序中，Kernel会查找是否有应用程序注册了对键盘事件的响应，若有，则通过调用注册的响应函数的方式来通知应用程序。此时在应用程序的键盘响应函数中，可以通过 `in` 指令从键盘的数据端口读取扫描码完成数据交换。键盘数据端口约定为 `0x60`，键盘扫描码的编码方式参照这个约定。