南京航空航天大学 计算机科学与技术系学院 计算机组成原理 课程实验

学号：161630220

姓名：赵维康

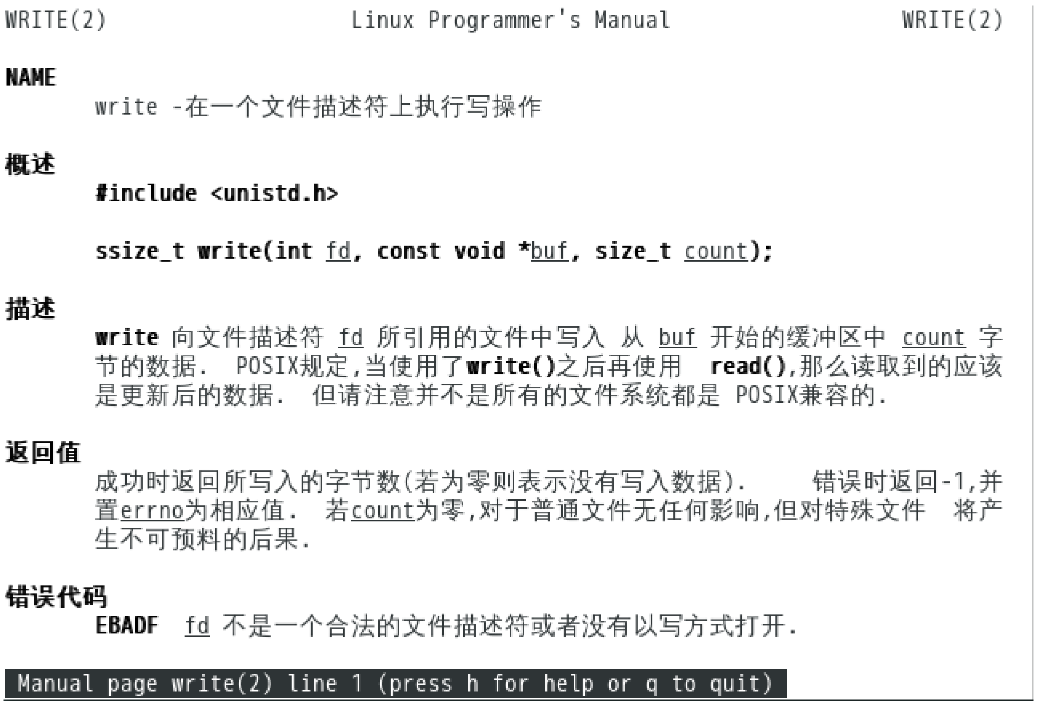
PA3- 穿越时空的旅程： 异常控制流

在操作系统上运行 **Hello World**

标准输出

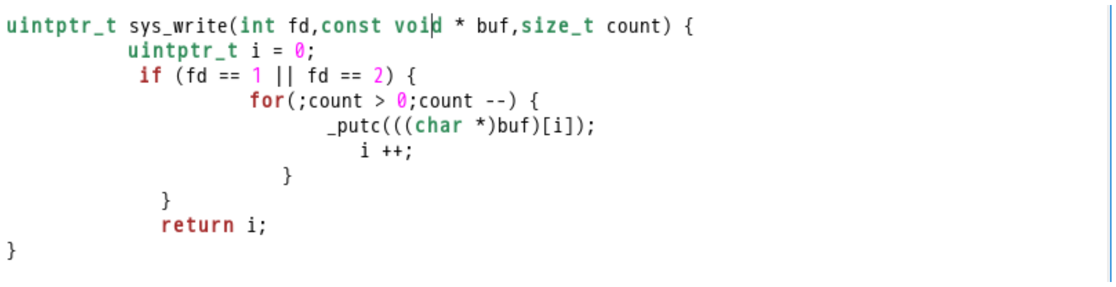
在 **Nanos-lite** 上运行 **Hello world**

使用 **man 2 write** 查看一下 **write** 的声明，如图



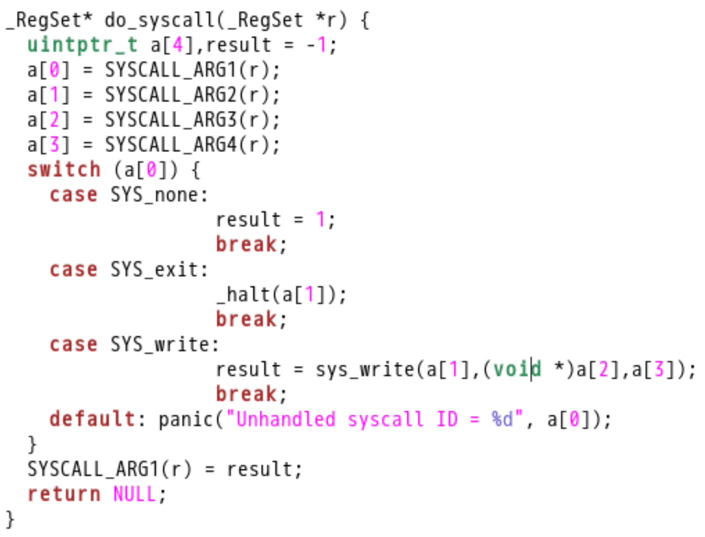
```
WRITE(2)                  Linux Programmer's Manual                 WRITE(2)

NAME
       write -在一个文件描述符上执行写操作

概述
       #include <unistd.h>

       ssize_t write(int fd, const void *buf, size_t count);

描述
       write 向文件描述符 fd 所引用的文件中写入 从 buf 开始的缓冲区中 count 字
       节的数据.  POSIX规定,当使用了write()之后再使用  read(),那么读取到的应该
       是更新后的数据.  但请注意并不是所有的文件系统都是 POSIX兼容的.

返回值
       成功时返回所写入的字节数(若为零则表示没有写入数据).     错误时返回-1,并
       置errno为相应值.  若count为零,对于普通文件无任何影响,但对特殊文件  将产
       生不可预料的后果.

错误代码
       EBADF  fd 不是一个合法的文件描述符或者没有以写方式打开.

 Manual page write(2) line 1 (press h for help or q to quit)
```

下面进入 **nanos-lite/src/syscall.c** 中，定义一个 **SYS_write** 系统调用的函数（所谓的 **write**（））**sys_write()**（这个函数的操作为：检查 **fd** 的值,若 **fd** 是 **1** 或 **2(**分别代表 **stdout** 和 **stderr)**，则将 **buf** 为首地址的 **count** 字节输出到串口**(**使用**_putc()**即可**)**)，而它的返回值，根据 **write** 函数的声明知，返回写入字节数（**0** 表示无写入），如图



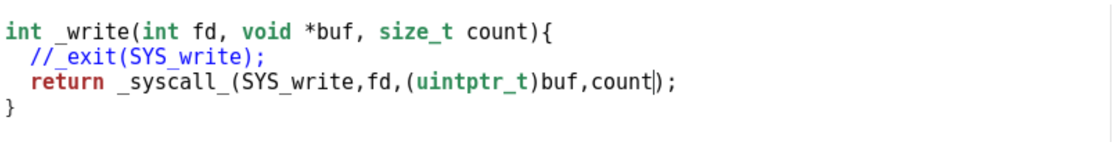
```
uintptr_t sys_write(int fd,const void * buf,size_t count) {
          uintptr_t i = 0;
           if (fd == 1 || fd == 2) {
                    for(;count > 0;count --) {
                           _putc(((char *)buf)[i]);
                              i ++;
                       }
             }
             return i;
}
```

然后，在 do_syscall()中再定义其余三个系统调用参数（eax、ebx、ecx、edx 这四个），同时识别 SYS_write 系统调用号，并设置返回值（放在 eax 中），如图



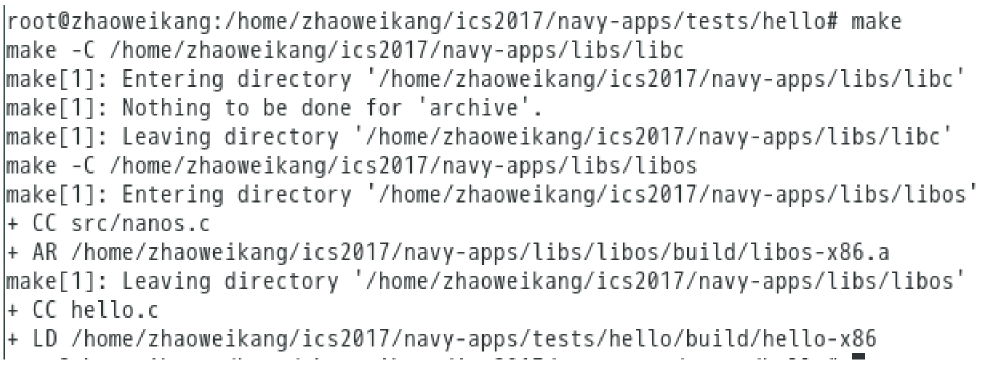
```
_RegSet* do_syscall(_RegSet *r) {
  uintptr_t a[4],result = -1;
  a[0] = SYSCALL_ARG1(r);
  a[1] = SYSCALL_ARG2(r);
  a[2] = SYSCALL_ARG3(r);
  a[3] = SYSCALL_ARG4(r);
  switch (a[0]) {
    case SYS_none:
                  result = 1;
                  break;
    case SYS_exit:
                  _halt(a[1]);
                  break;
    case SYS_write:
                  result = sys_write(a[1],(void *)a[2],a[3]);
                  break;
    default: panic("Unhandled syscall ID = %d", a[0]);
  }
  SYSCALL_ARG1(r) = result;
  return NULL;
}
```

下面进入 navy-apps/libs/libos/src/nanos.c 的_write()函数中调用系统调用接口函数(_syscall_（）函数，为什么要实现_write()函数？因为 SYS_write 系统调用会调用_write()，它总是返回-1，表示写入错误），如图



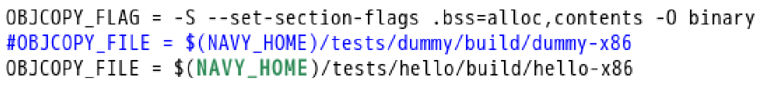
```
int _write(int fd, void *buf, size_t count){
  //_exit(SYS_write);
  return _syscall_(SYS_write,fd,(uintptr_t)buf,count);
}
```

下面把在 Nanos-lit 上运行的用户程序切换成 hello 程序，切换到navy-apps/tests/hello/ 目录下执行 make 编译 hello 程序，如图



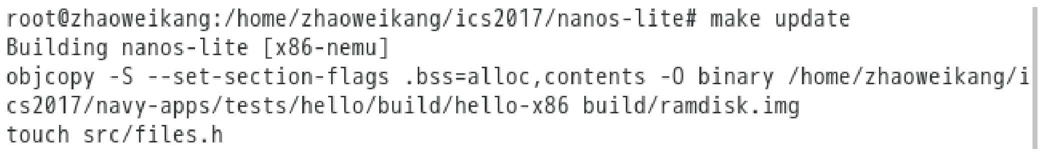
```
root@zhaoweikang:/home/zhaoweikang/ics2017/navy-apps/tests/hello# make
make -C /home/zhaoweikang/ics2017/navy-apps/libs/libc
make[1]: Entering directory '/home/zhaoweikang/ics2017/navy-apps/libs/libc'
make[1]: Nothing to be done for 'archive'.
make[1]: Leaving directory '/home/zhaoweikang/ics2017/navy-apps/libs/libc'
make -C /home/zhaoweikang/ics2017/navy-apps/libs/libos
make[1]: Entering directory '/home/zhaoweikang/ics2017/navy-apps/libs/libos'
+ CC src/nanos.c
+ AR /home/zhaoweikang/ics2017/navy-apps/libs/libos/build/libos-x86.a
make[1]: Leaving directory '/home/zhaoweikang/ics2017/navy-apps/libs/libos'
+ CC hello.c
+ LD /home/zhaoweikang/ics2017/navy-apps/tests/hello/build/hello-x86
```

然后，修改 nanos-lite/Makefile 中 ramdisk 的生成规则,把 ramdisk 中的唯一的文件换成 hello 程序，如图



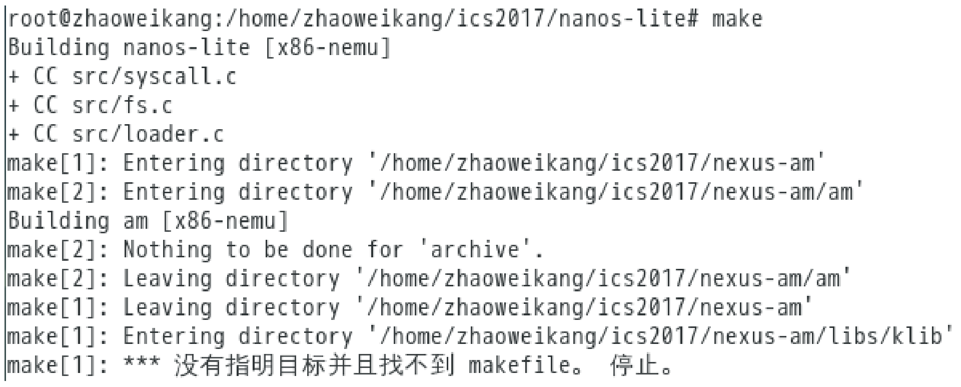
```
OBJCOPY_FLAG = -S --set-section-flags .bss=alloc,contents -O binary
#OBJCOPY_FILE = $(NAVY_HOME)/tests/dummy/build/dummy-x86
OBJCOPY_FILE = $(NAVY_HOME)/tests/hello/build/hello-x86
```

在 nanos-lite/下执行 make update 更新 ramdisk，如图

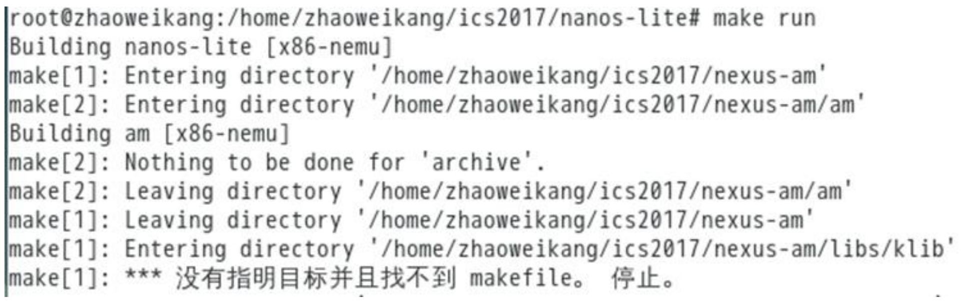


```
root@zhaoweikang:/home/zhaoweikang/ics2017/nanos-lite# make update
Building nanos-lite [x86-nemu]
objcopy -S --set-section-flags .bss=alloc,contents -O binary /home/zhaoweikang/i
cs2017/navy-apps/tests/hello/build/hello-x86 build/ramdisk.img
touch src/files.h
```

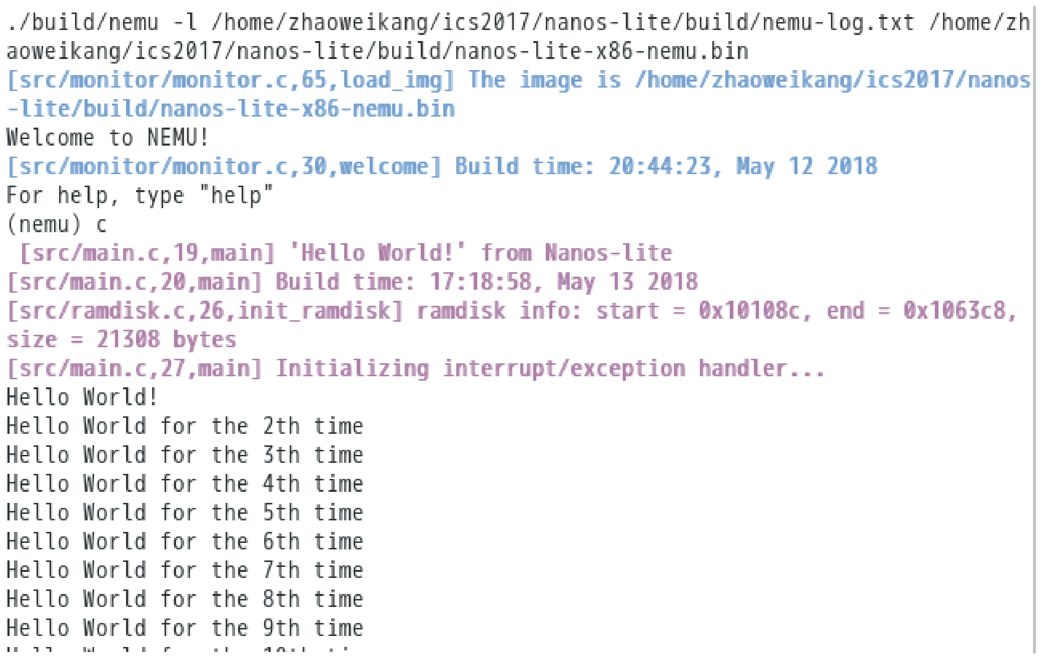
重新编译 Nanos-lite 并运行，如图



```
root@zhaoweikang:/home/zhaoweikang/ics2017/nanos-lite# make
Building nanos-lite [x86-nemu]
+ CC src/syscall.c
+ CC src/fs.c
+ CC src/loader.c
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am'
make[2]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/am'
Building am [x86-nemu]
make[2]: Nothing to be done for 'archive'.
make[2]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am/am'
make[1]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am'
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/libs/klib'
make[1]: *** 没有指明目标并且找不到 makefile。 停止。
```



```
root@zhaoweikang:/home/zhaoweikang/ics2017/nanos-lite# make run
Building nanos-lite [x86-nemu]
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am'
make[2]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/am'
Building am [x86-nemu]
make[2]: Nothing to be done for 'archive'.
make[2]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am/am'
make[1]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am'
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/libs/klib'
make[1]: *** 没有指明目标并且找不到 makefile。 停止。
```



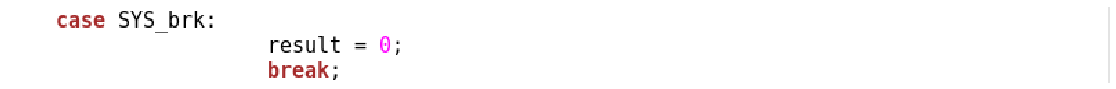
```
./build/nemu -l /home/zhaoweikang/ics2017/nanos-lite/build/nemu-log.txt /home/zh
aoweikang/ics2017/nanos-lite/build/nanos-lite-x86-nemu.bin
[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nanos
-lite/build/nanos-lite-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 20:44:23, May 12 2018
For help, type "help"
(nemu) c
 [src/main.c,19,main] 'Hello World!' from Nanos-lite
[src/main.c,20,main] Build time: 17:18:58, May 13 2018
[src/ramdisk.c,26,init_ramdisk] ramdisk info: start = 0x10108c, end = 0x1063c8,
size = 21308 bytes
[src/main.c,27,main] Initializing interrupt/exception handler...
Hello World!
Hello World for the 2th time
Hello World for the 3th time
Hello World for the 4th time
Hello World for the 5th time
Hello World for the 6th time
Hello World for the 7th time
Hello World for the 8th time
Hello World for the 9th time
```

（后面的 ctrl+c 结束）成功运行 hello world。

堆区管理

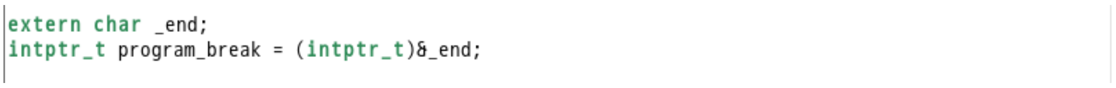
实现堆区管理

下面进入 **nanos-lite/src/syscall.c** 中，在 **do_syscall()**中对 **SYS_brk** 系统调用识别，并且使 **SYS_brk** 系统调用总是返回 **0** 即可,表示堆区大小的调整总是成功，如图



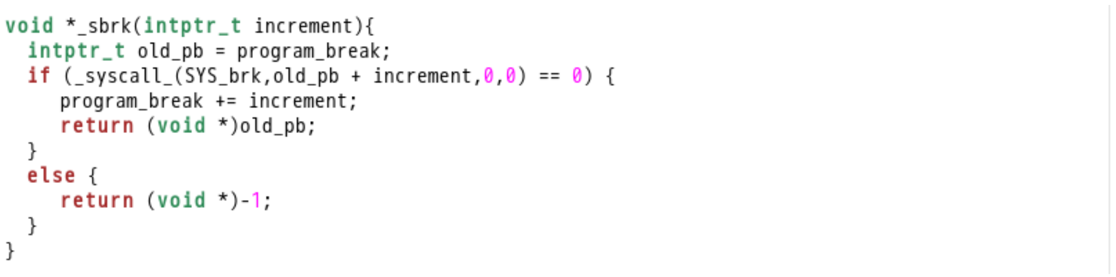
```
case SYS_brk:
                result = 0;
                break;
```

下面进入 **navy-apps/libs/libos/src/nanos.c** 中，先定义**_end** 符号，并且使 **program break** 一开始的位置位于**_end**，如图



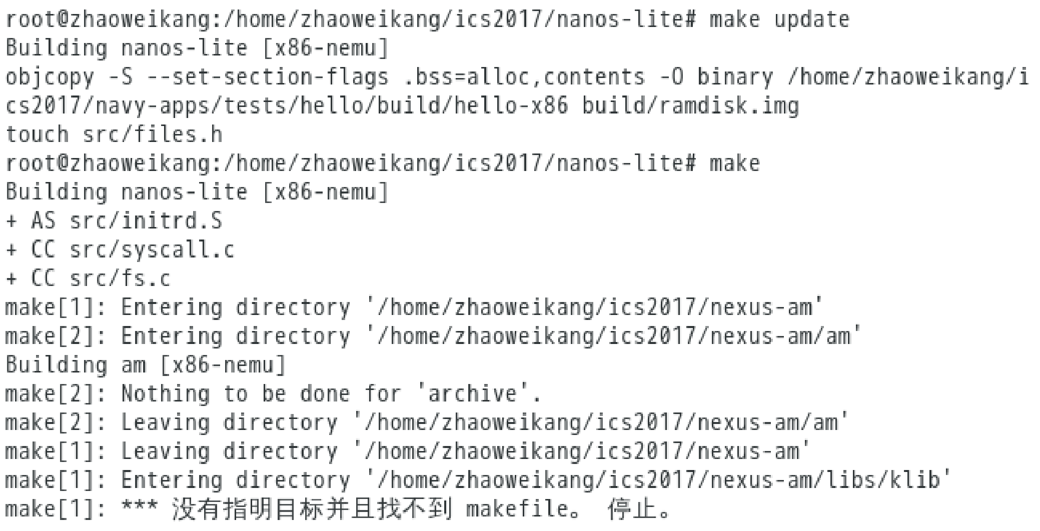
```
extern char _end;
intptr_t program_break = (intptr_t)&_end;
```

然后在**_sbrk()**函数中，进行以下操作：根据记录的 **program break** 位置和参数 **increment,**计算出新 **program break**；通过 **SYS_brk**系统调用（**_syscall_()**）来让操作系统设置新 **program break**；若 **SYS_brk** 系统调用成功（即返回 **0**），此时更新之前记录的 **program break**（**program_brk+= increment**）的位置,并将旧 **program break**（**old_brk**）的位置作为**_sbrk()**的返回值返回, 若系统调用失败，**_sbrk()**会返回**-1**，如图

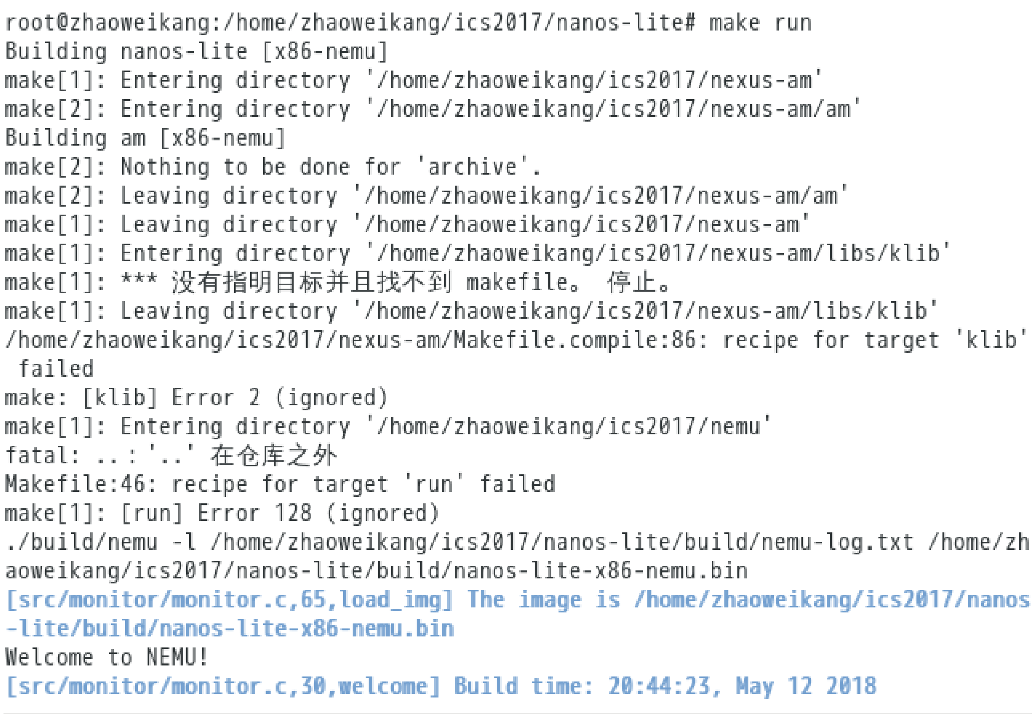


```
void *_sbrk(intptr_t increment){
  intptr_t old_pb = program_break;
  if (_syscall_(SYS_brk,old_pb + increment,0,0) == 0) {
     program_break += increment;
     return (void *)old_pb;
  }
  else {
     return (void *)-1;
  }
}
```

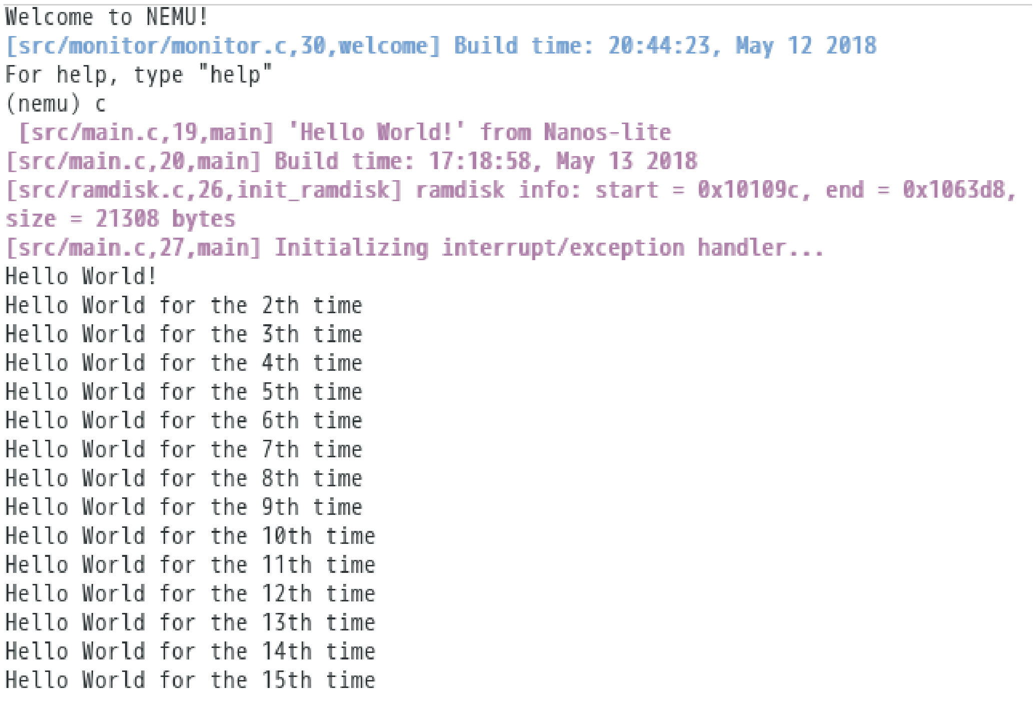
下面在 **nanos-lite** 目录下执行 **make update** 命令，然后编译运行 **hello world,**，如图



```
root@zhaoweikang:/home/zhaoweikang/ics2017/nanos-lite# make update
Building nanos-lite [x86-nemu]
objcopy -S --set-section-flags .bss=alloc,contents -O binary /home/zhaoweikang/i
cs2017/navy-apps/tests/hello/build/hello-x86 build/ramdisk.img
touch src/files.h
root@zhaoweikang:/home/zhaoweikang/ics2017/nanos-lite# make
Building nanos-lite [x86-nemu]
+ AS src/initrd.S
+ CC src/syscall.c
+ CC src/fs.c
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am'
make[2]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/am'
Building am [x86-nemu]
make[2]: Nothing to be done for 'archive'.
make[2]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am/am'
make[1]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am'
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/libs/klib'
make[1]: *** 没有指明目标并且找不到 makefile。 停止。
```



```
root@zhaoweikang:/home/zhaoweikang/ics2017/nanos-lite# make run
Building nanos-lite [x86-nemu]
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am'
make[2]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/am'
Building am [x86-nemu]
make[2]: Nothing to be done for 'archive'.
make[2]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am/am'
make[1]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am'
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/libs/klib'
make[1]: *** 没有指明目标并且找不到 makefile。 停止。
make[1]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am/libs/klib'
/home/zhaoweikang/ics2017/nexus-am/Makefile.compile:86: recipe for target 'klib'
 failed
make: [klib] Error 2 (ignored)
make[1]: Entering directory '/home/zhaoweikang/ics2017/nemu'
fatal: ..: '..' 在仓库之外
Makefile:46: recipe for target 'run' failed
make[1]: [run] Error 128 (ignored)
./build/nemu -l /home/zhaoweikang/ics2017/nanos-lite/build/nemu-log.txt /home/zh
aoweikang/ics2017/nanos-lite/build/nanos-lite-x86-nemu.bin
[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nanos
-lite/build/nanos-lite-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 20:44:23, May 12 2018
```

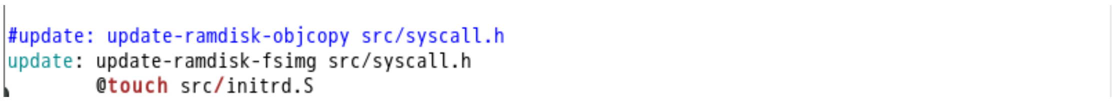


```
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 20:44:23, May 12 2018
For help, type "help"
(nemu) c
 [src/main.c,19,main] 'Hello World!' from Nanos-lite
[src/main.c,20,main] Build time: 17:18:58, May 13 2018
[src/ramdisk.c,26,init_ramdisk] ramdisk info: start = 0x10109c, end = 0x1063d8,
size = 21308 bytes
[src/main.c,27,main] Initializing interrupt/exception handler...
Hello World!
Hello World for the 2th time
Hello World for the 3th time
Hello World for the 4th time
Hello World for the 5th time
Hello World for the 6th time
Hello World for the 7th time
Hello World for the 8th time
Hello World for the 9th time
Hello World for the 10th time
Hello World for the 11th time
Hello World for the 12th time
Hello World for the 13th time
Hello World for the 14th time
Hello World for the 15th time
```

（后面的 ctrl+c 结束）成功运行 hello world。可以看到，字符串通过一 次 write 系统调用进行输出，而不是逐个字符地进行输出，速度明显快了许多。

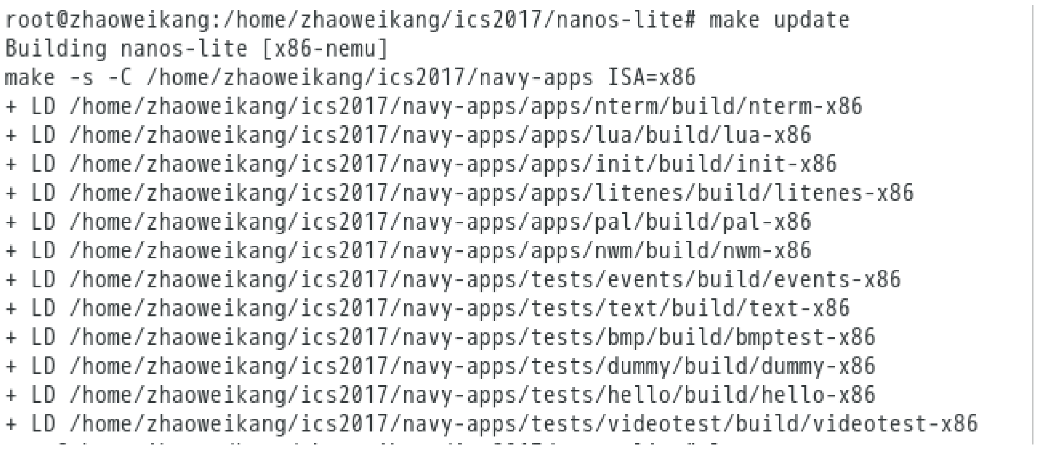
简易文件系统

先进入 nanos-lite/Makefile，做一些修改，以便维护 ramdisk 中各个文件的信息，如图



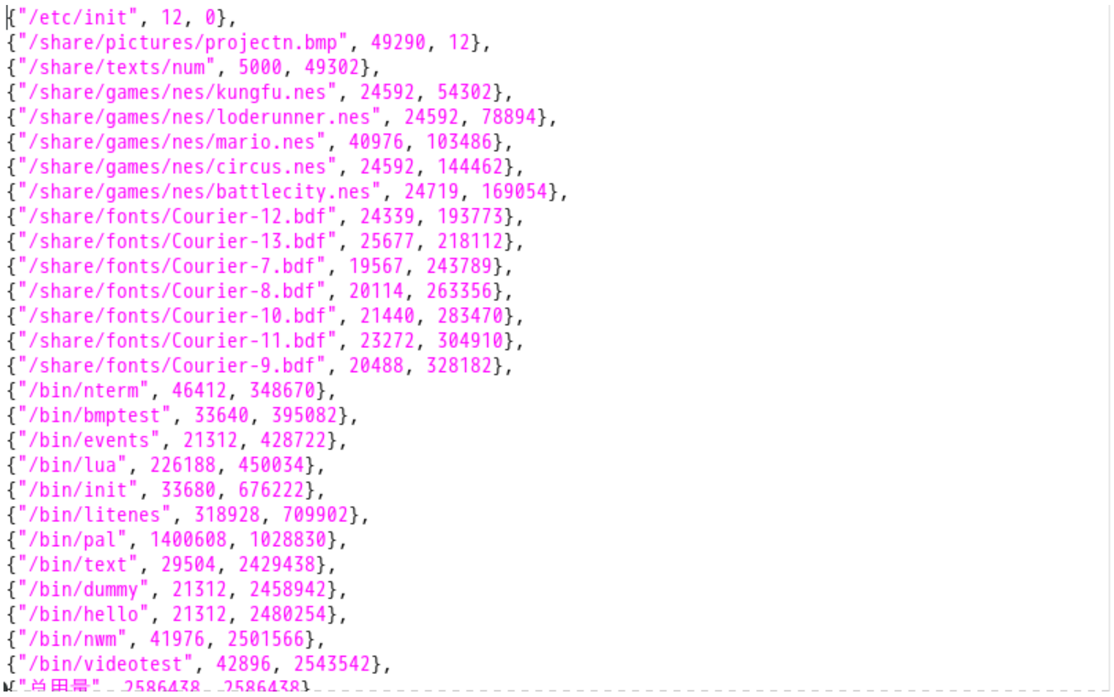
```
#update: update-ramdisk-objcopy src/syscall.h
update: update-ramdisk-fsimg src/syscall.h
	@touch src/initrd.S
```

然后运行 make update 就会自动编译 Navy-apps 里面的所有程序，如图



```
root@zhaoweikang:/home/zhaoweikang/ics2017/nanos-lite# make update
Building nanos-lite [x86-nemu]
make -s -C /home/zhaoweikang/ics2017/navy-apps ISA=x86
+ LD /home/zhaoweikang/ics2017/navy-apps/apps/nterm/build/nterm-x86
+ LD /home/zhaoweikang/ics2017/navy-apps/apps/lua/build/lua-x86
+ LD /home/zhaoweikang/ics2017/navy-apps/apps/init/build/init-x86
+ LD /home/zhaoweikang/ics2017/navy-apps/apps/litenes/build/litenes-x86
+ LD /home/zhaoweikang/ics2017/navy-apps/apps/pal/build/pal-x86
+ LD /home/zhaoweikang/ics2017/navy-apps/apps/nwm/build/nwm-x86
+ LD /home/zhaoweikang/ics2017/navy-apps/tests/events/build/events-x86
+ LD /home/zhaoweikang/ics2017/navy-apps/tests/text/build/text-x86
+ LD /home/zhaoweikang/ics2017/navy-apps/tests/bmp/build/bmptest-x86
+ LD /home/zhaoweikang/ics2017/navy-apps/tests/dummy/build/dummy-x86
+ LD /home/zhaoweikang/ics2017/navy-apps/tests/hello/build/hello-x86
+ LD /home/zhaoweikang/ics2017/navy-apps/tests/videotest/build/videotest-x86
```

并将 **navy-apps/fsimg/** 目录下的所有内容整合成 **ramdisk** 镜像,同时生成这个 **ramdisk** 镜像的文件记录表（在 **nanoslite/src/files.h** 中），**files.h** 中的内容如图

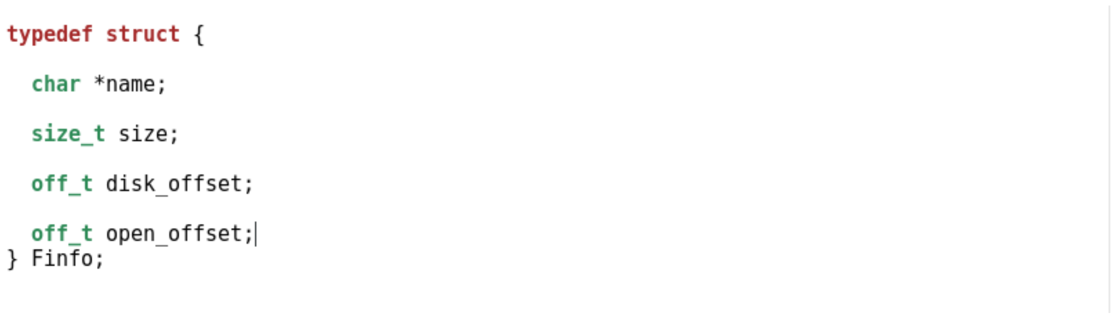


```
{"/etc/init", 12, 0},
{"/share/pictures/projectn.bmp", 49290, 12},
{"/share/texts/num", 5000, 49302},
{"/share/games/nes/kungfu.nes", 24592, 54302},
{"/share/games/nes/loderunner.nes", 24592, 78894},
{"/share/games/nes/mario.nes", 40976, 103486},
{"/share/games/nes/circus.nes", 24592, 144462},
{"/share/games/nes/battlecity.nes", 24719, 169054},
{"/share/fonts/Courier-12.bdf", 24339, 193773},
{"/share/fonts/Courier-13.bdf", 25677, 218112},
{"/share/fonts/Courier-7.bdf", 19567, 243789},
{"/share/fonts/Courier-8.bdf", 20114, 263356},
{"/share/fonts/Courier-10.bdf", 21440, 283470},
{"/share/fonts/Courier-11.bdf", 23272, 304910},
{"/share/fonts/Courier-9.bdf", 20488, 328182},
{"/bin/nterm", 46412, 348670},
{"/bin/bmptest", 33640, 395082},
{"/bin/events", 21312, 428722},
{"/bin/lua", 226188, 450034},
{"/bin/init", 33680, 676222},
{"/bin/litenes", 318928, 709902},
{"/bin/pal", 1400608, 1028830},
{"/bin/text", 29504, 2429438},
{"/bin/dummy", 21312, 2458942},
{"/bin/hello", 21312, 2480254},
{"/bin/nwm", 41976, 2501566},
{"/bin/videotest", 42896, 2543542},
{"总用量", 2586438, 2586438}
```

果然，每一项都符合"文件记录表”的内容（即文件名、大小、在 **ramdisk** 的偏移）

下面进入 **nanos-lite/src/fs.c** 中，实现文件操作

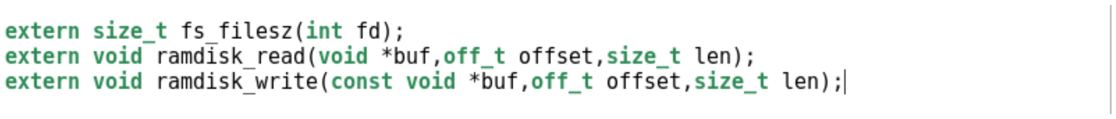
首先，在 **Finfo** 这个结构体中添加文件读写指针偏移量属性 **open_offset**(记录目前文件操作的位置)，如图



```
typedef struct {
  char *name;
  size_t size;
  off_t disk_offset;
  off_t open_offset;
} Finfo;
```

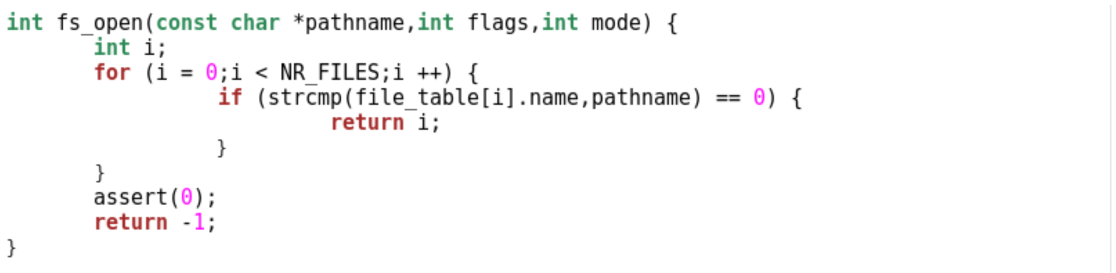
下面声明 **ramdisk_read()**、**ramdisk_write()**函数，以及辅助函数 **fs_filesz()**（实现 **fs_read()**、

fs_write()均会用到），如图



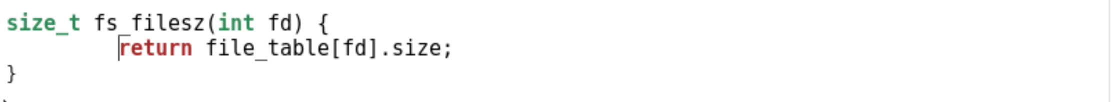
```
extern size_t fs_filesz(int fd);
extern void ramdisk_read(void *buf,off_t offset,size_t len);
extern void ramdisk_write(const void *buf,off_t offset,size_t len);
```

下面实现 fs_open()函数，由 man 2 open 知，open()将 pathname 对应为一个文件描述符（fd），flags 表示读写权限（只读、只写、读写，pa 中忽略），若 pathname=file_table[].name，返回 fd，否则，assert（0）终止程序，并返回-1，如图



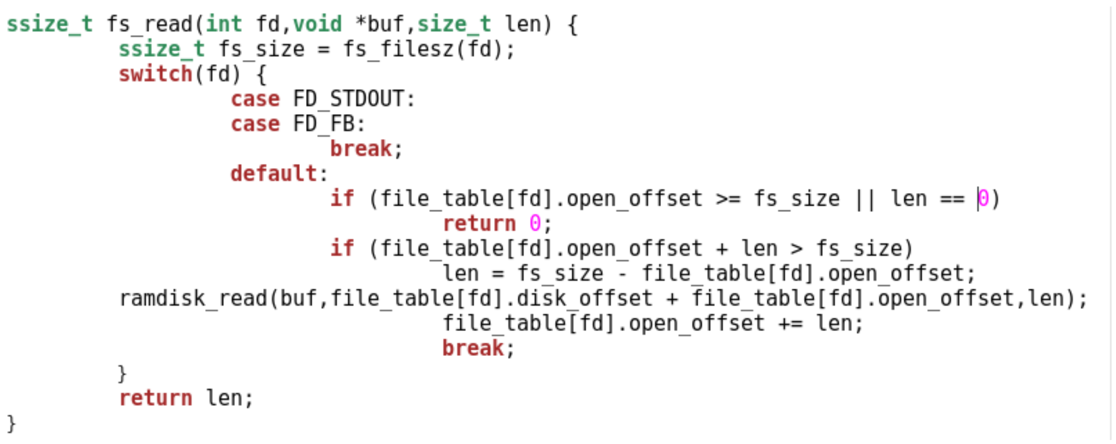
```
int fs_open(const char *pathname,int flags,int mode) {
	int i;
	for (i = 0;i < NR_FILES;i ++) {
		if (strcmp(file_table[i].name,pathname) == 0) {
			return i;
		}
	}
	assert(0);
	return -1;
}
```

在实现 fs_read()、fs_write()之前，先实现辅助函数 fs_filesz()（返回文件描述符 fd 所描述的文件的大小），如图



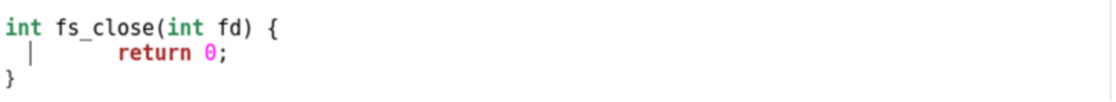
```
size_t fs_filesz(int fd) {
	return file_table[fd].size;
}
```

下面实现 fs_read()函数，在此函数中，对 stdin，stdout 和 stderr这三个特殊文件的操作直接忽略。由 man 2 read 知，若 len 为 0 或 open_offset 位于 fs_size 结尾处，read()返回 0,不执行其他任何操作。若 open_offset+len 大于 fs_size，len=fs_size -open_offset,然后调用 ramdisk_read()读文件，函数返回值为 len。如图



```
ssize_t fs_read(int fd,void *buf,size_t len) {
	ssize_t fs_size = fs_filesz(fd);
	switch(fd) {
		case FD_STDOUT:
		case FD_FB:
			break;
		default:
			if (file_table[fd].open_offset >= fs_size || len == 0)
				return 0;
			if (file_table[fd].open_offset + len > fs_size)
				len = fs_size - file_table[fd].open_offset;
	ramdisk_read(buf,file_table[fd].disk_offset + file_table[fd].open_offset,len);
				file_table[fd].open_offset += len;
				break;
	}
	return len;
}
```

下面实现 fs_close()函数（直接返回 0，表示总是关闭成功），如图



```
int fs_close(int fd) {
	return 0;
}
```

下面进入 nanos-lite/src/loader.c 中，让 loader 使用文件，由于需要用到 fs_read()、fs_filesz()、fs_open()、fs_close()，故先声明，如图

```
extern ssize_t fs_read(int fd, void *buf, size_t len);

extern size_t fs_filesz(int fd);

extern int fs_open(const char *pathname, int flags, int mode);

extern int fs_close(int fd);
```

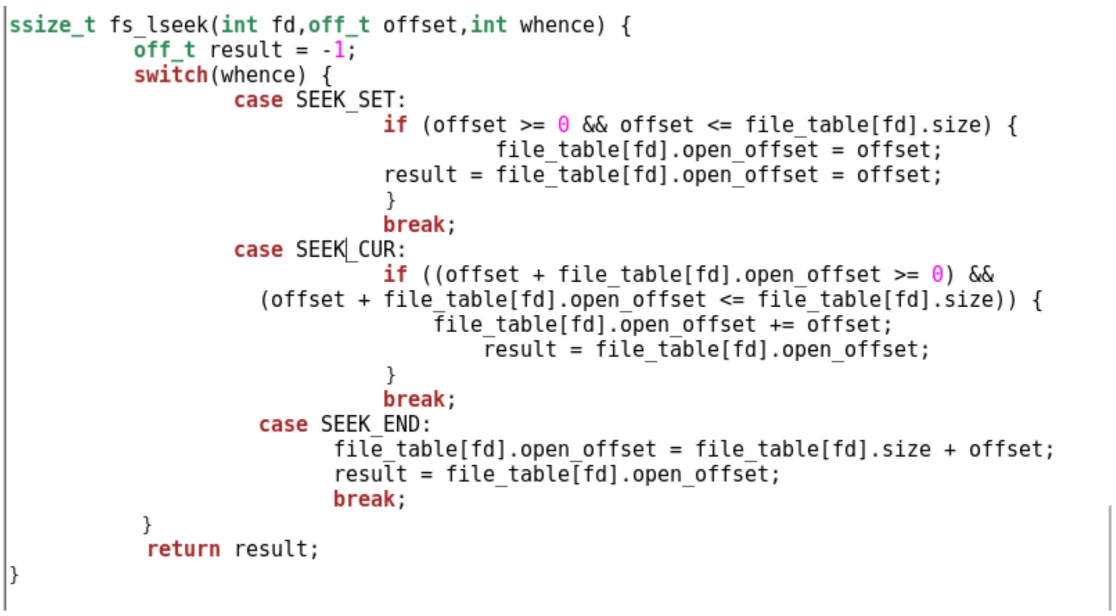
首先将 ramdisk_read()注释掉(不再适用)，调用 fs_open()找到文件描述符 fd，然后用 fs_read()去读，调用 fs_close()关闭文件，如图

```
uintptr_t loader(_Protect *as, const char *filename) {

  //ramdisk_read(DEFAULT_ENTRY, 0, get_ramdisk_size());

      int fd = fs_open(filename,0,0);
      printf("fd = %d\n",fd);
      fs_read(fd,DEFAULT_ENTRY,fs_filesz(fd));
      fs_close(fd);

      return (uintptr_t)DEFAULT_ENTRY;

}
```

下面实现 fs_write()函数，在 fs_write()中对 stdout 和 stderr 所做的操作就是用_putc()输出到串口，其中 default 项同 fs_read()的 default，不再赘述，如图

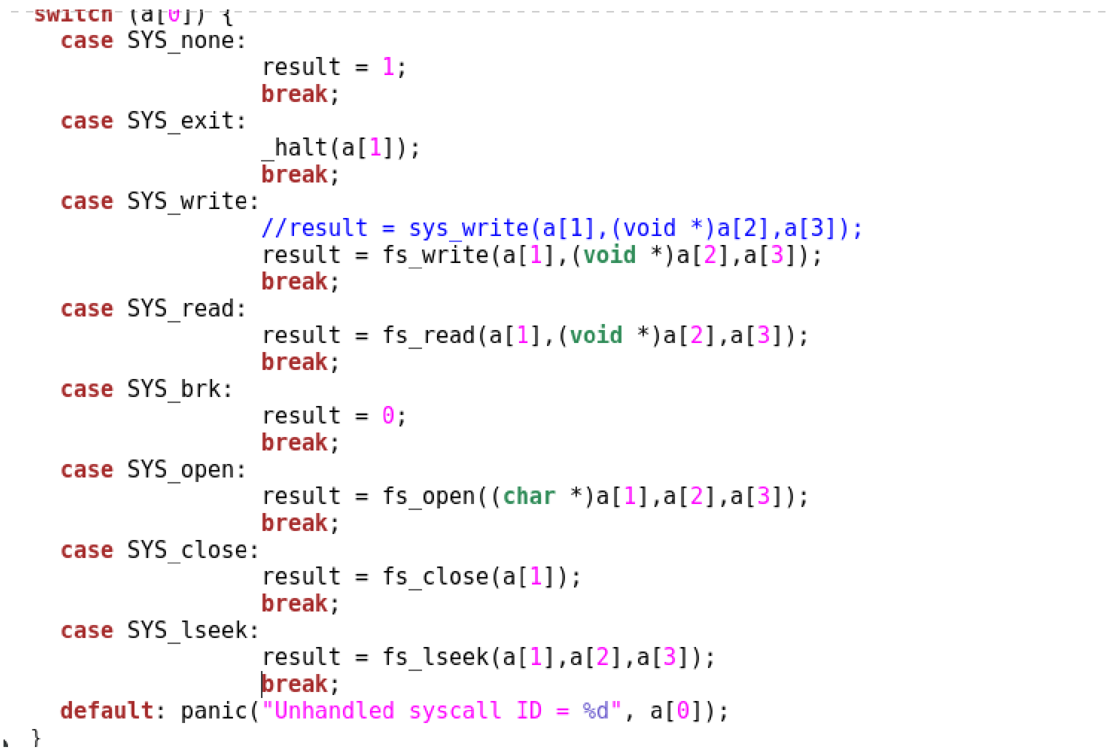
```
ssize_t fs_write(int fd,const void *buf,int len) {
           ssize_t fs_size = fs_filesz(fd);
           switch(fd) {
                      case FD_STDOUT:
                      case FD_STDERR:
                              for (int i = 0;i < len;i ++) {
                              _putc(((char *)buf)[i]);
                              }
                              break;
                      default:
                              if(file_table[fd].open_offset >= fs_size)

                                  return 0;

                              if(file_table[fd].open_offset + len > fs_size)

                              len = fs_size - file_table[fd].open_offset;

       ramdisk_write(buf, file_table[fd].disk_offset + file_table[fd].open_offset, len);

                              file_table[fd].open_offset += len;

                              break;
           }
            return len;
}
```

下面实现 fs_lseek()（对 offset 偏移量的调整）函数，由 man 2 lseek 知，lseek 的 whence 包括 SEEK_SET（the file offset is set to offset bytes）、SEEK_CUR（the file offset is set to its current location plus offset bytes.）、SEEK_END（the file offset is set to the size of the file plus offset bytes.），对于返回值，若成功，则返回 offset 的位置，否则返回-1，如图



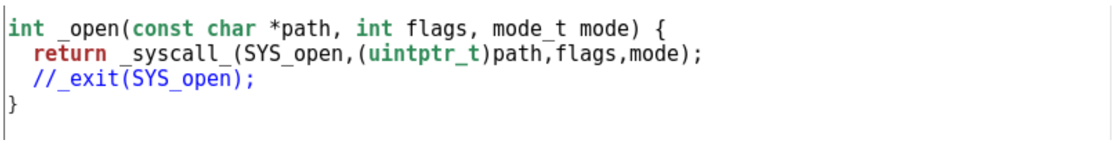
```
ssize_t fs_lseek(int fd,off_t offset,int whence) {
          off_t result = -1;
          switch(whence) {
                  case SEEK_SET:
                              if (offset >= 0 && offset <= file_table[fd].size) {
                                       file_table[fd].open_offset = offset;
                              result = file_table[fd].open_offset = offset;
                              }
                              break;
                  case SEEK_CUR:
                              if ((offset + file_table[fd].open_offset >= 0) &&
                    (offset + file_table[fd].open_offset <= file_table[fd].size)) {
                                  file_table[fd].open_offset += offset;
                                      result = file_table[fd].open_offset;
                              }
                              break;
                    case SEEK_END:
                          file_table[fd].open_offset = file_table[fd].size + offset;
                          result = file_table[fd].open_offset;
                          break;
           }
           return result;
}
```

下面进入 **nanos-lite/src/syscall.c**，在 **do_syscall()**中，识别出 **SYS_open**、**SYS_read**、**SYS_write**、**SYS_close**、**SYS_lseek** 的文件操作的系统调用，如图

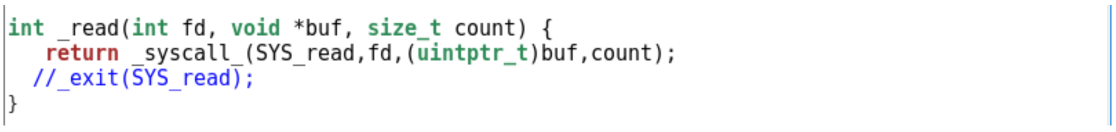


```
  switch (a[0]) {
    case SYS_none:
                    result = 1;
                    break;
    case SYS_exit:
                    _halt(a[1]);
                    break;
    case SYS_write:
                    //result = sys_write(a[1],(void *)a[2],a[3]);
                    result = fs_write(a[1],(void *)a[2],a[3]);
                    break;
    case SYS_read:
                    result = fs_read(a[1],(void *)a[2],a[3]);
                    break;
    case SYS_brk:
                    result = 0;
                    break;
    case SYS_open:
                    result = fs_open((char *)a[1],a[2],a[3]);
                    break;
    case SYS_close:
                    result = fs_close(a[1]);
                    break;
    case SYS_lseek:
                    result = fs_lseek(a[1],a[2],a[3]);
                    break;
    default: panic("Unhandled syscall ID = %d", a[0]);
  }
```

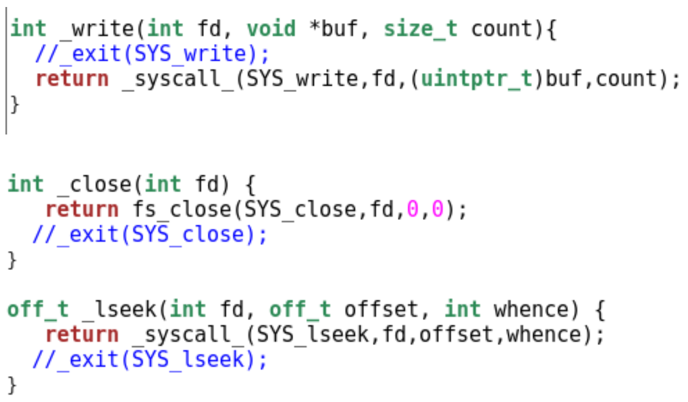
最后进入 **navy-apps/libs/libos/src/nanos.c**，分别在**_open()**、**_read()**、**_write()**、**_close()**、**_lseek()**，中调用系统调用接口函数（**_syscall_()**），如图

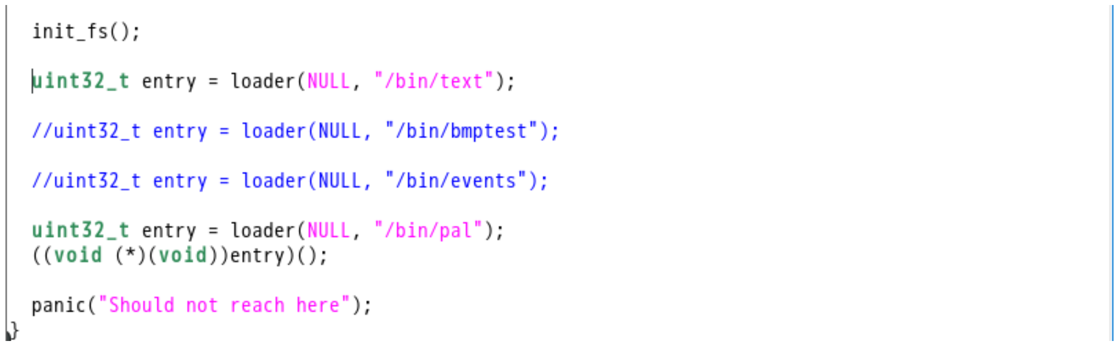


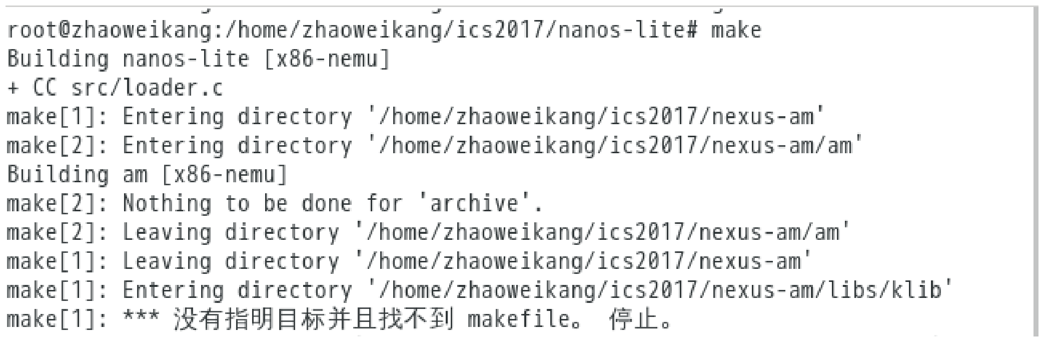
```
int _open(const char *path, int flags, mode_t mode) {
  return _syscall_(SYS_open,(uintptr_t)path,flags,mode);
  //_exit(SYS_open);
}
```

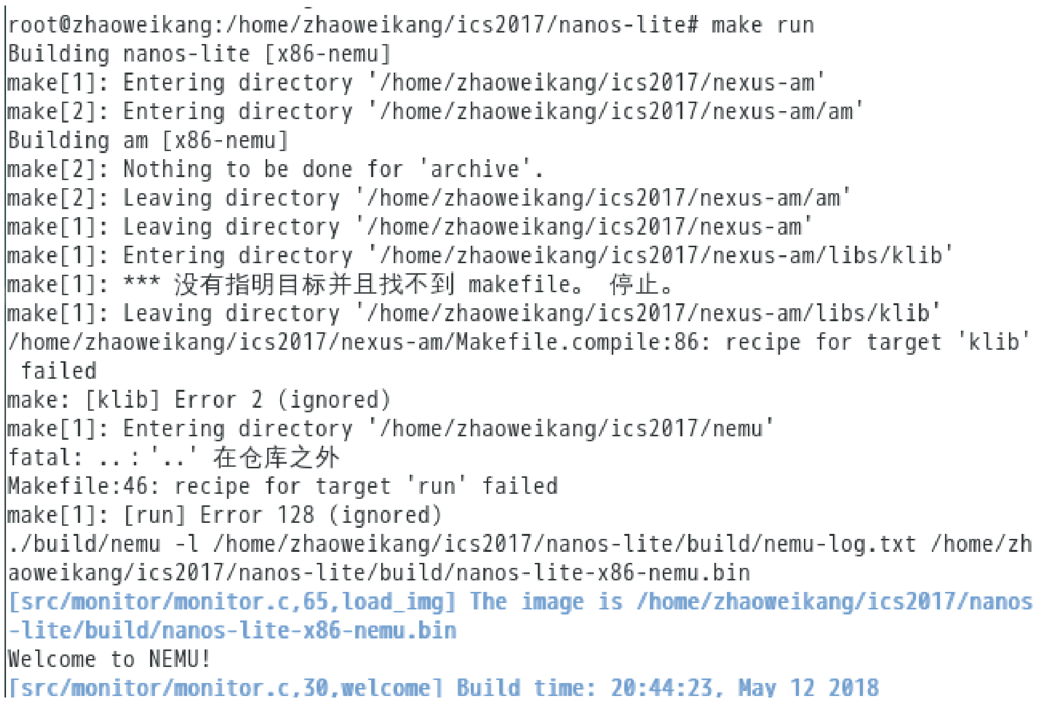


```
int _read(int fd, void *buf, size_t count) {
   return _syscall_(SYS_read,fd,(uintptr_t)buf,count);
  //_exit(SYS_read);
}
```

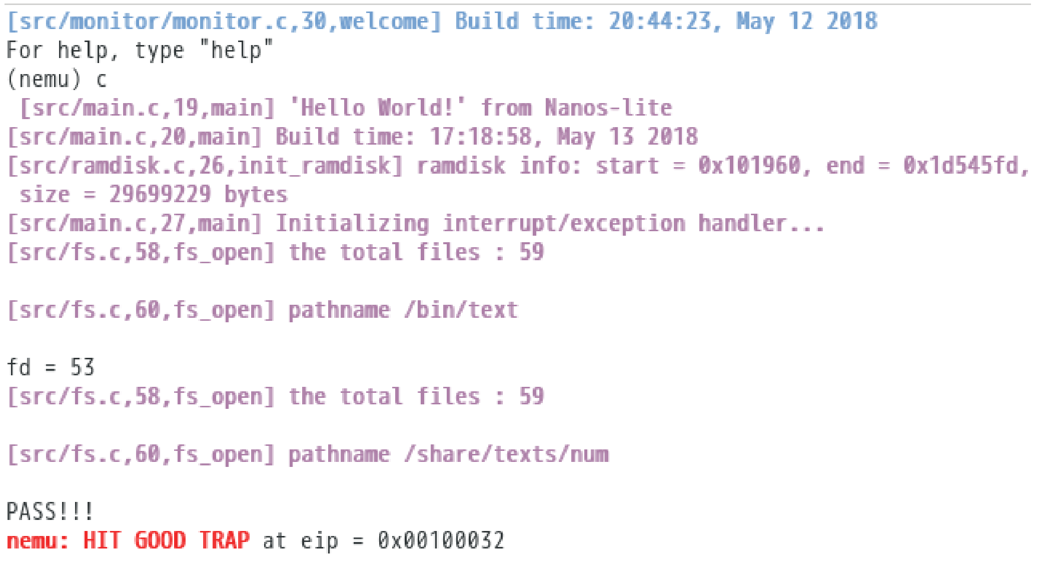

```
int _write(int fd, void *buf, size_t count){
  //_exit(SYS_write);
  return _syscall_(SYS_write,fd,(uintptr_t)buf,count);
}


int _close(int fd) {
   return fs_close(SYS_close,fd,0,0);
  //_exit(SYS_close);
}

off_t _lseek(int fd, off_t offset, int whence) {
   return _syscall_(SYS_lseek,fd,offset,whence);
  //_exit(SYS_lseek);
}
```
下面在 **loader()**的调用点（**src/main.c**）传入文件名“**/bin/text**”，来测试程序**/bin/text,**如图


```
  init_fs();

  uint32_t entry = loader(NULL, "/bin/text");

  //uint32_t entry = loader(NULL, "/bin/bmptest");

  //uint32_t entry = loader(NULL, "/bin/events");

  uint32_t entry = loader(NULL, "/bin/pal");
  ((void (*)(void))entry)();

  panic("Should not reach here");
}
```
然后运行测试程序**/bin/text**，如图


```
root@zhaoweikang:/home/zhaoweikang/ics2017/nanos-lite# make
Building nanos-lite [x86-nemu]
+ CC src/loader.c
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am'
make[2]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/am'
Building am [x86-nemu]
make[2]: Nothing to be done for 'archive'.
make[2]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am/am'
make[1]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am'
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/libs/klib'
make[1]: *** 没有指明目标并且找不到 makefile。 停止。
```



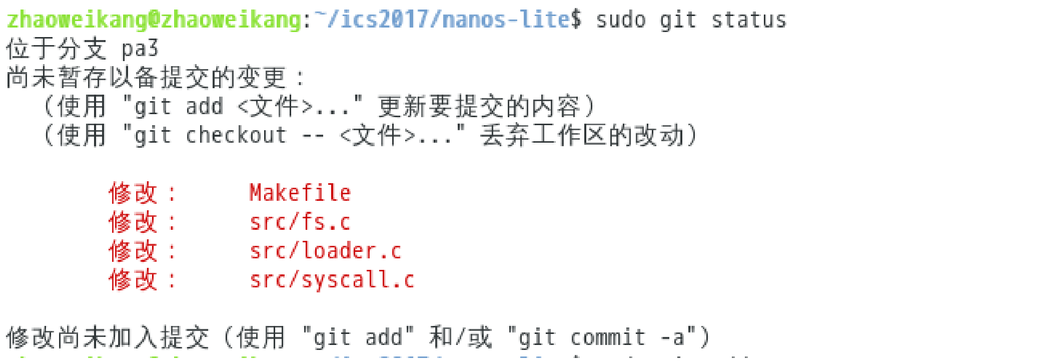
```
root@zhaoweikang:/home/zhaoweikang/ics2017/nanos-lite# make run
Building nanos-lite [x86-nemu]
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am'
make[2]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/am'
Building am [x86-nemu]
make[2]: Nothing to be done for 'archive'.
make[2]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am/am'
make[1]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am'
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/libs/klib'
make[1]: *** 没有指明目标并且找不到 makefile。 停止。
make[1]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am/libs/klib'
/home/zhaoweikang/ics2017/nexus-am/Makefile.compile:86: recipe for target 'klib'
 failed
make: [klib] Error 2 (ignored)
make[1]: Entering directory '/home/zhaoweikang/ics2017/nemu'
fatal: ..: '..' 在仓库之外
Makefile:46: recipe for target 'run' failed
make[1]: [run] Error 128 (ignored)
./build/nemu -l /home/zhaoweikang/ics2017/nanos-lite/build/nemu-log.txt /home/zh
aoweikang/ics2017/nanos-lite/build/nanos-lite-x86-nemu.bin
[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nanos
-lite/build/nanos-lite-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 20:44:23, May 12 2018
```



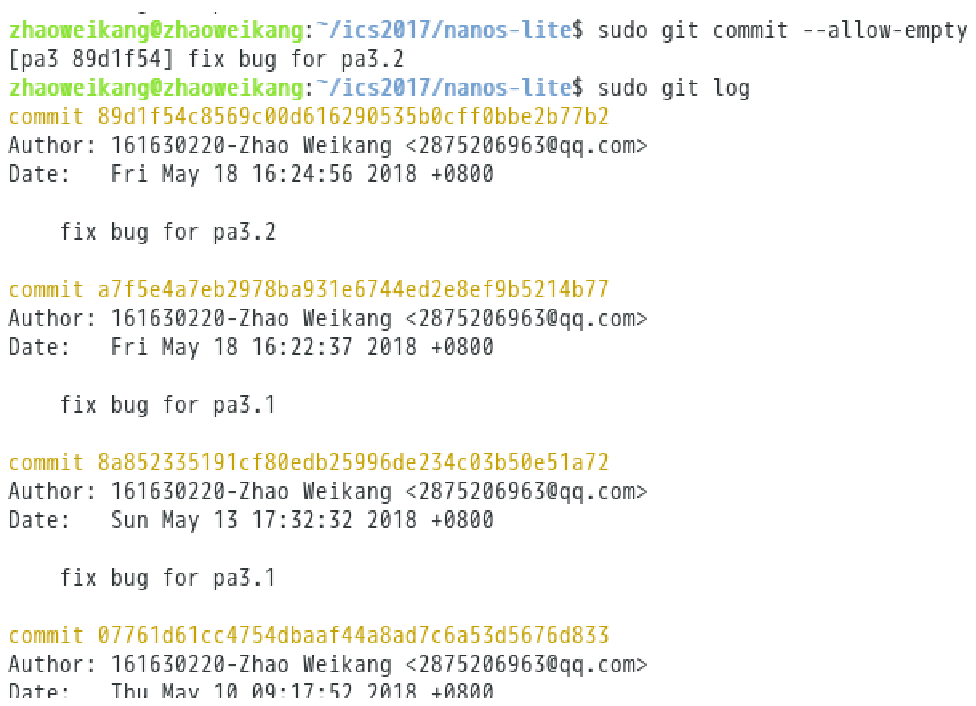
```
[src/monitor/monitor.c,30,welcome] Build time: 20:44:23, May 12 2018
For help, type "help"
(nemu) c
 [src/main.c,19,main] 'Hello World!' from Nanos-lite
[src/main.c,20,main] Build time: 17:18:58, May 13 2018
[src/ramdisk.c,26,init_ramdisk] ramdisk info: start = 0x101960, end = 0x1d545fd,
 size = 29699229 bytes
[src/main.c,27,main] Initializing interrupt/exception handler...
[src/fs.c,58,fs_open] the total files : 59

[src/fs.c,60,fs_open] pathname /bin/text

fd = 53
[src/fs.c,58,fs_open] the total files : 59

[src/fs.c,60,fs_open] pathname /share/texts/num

PASS!!!
nemu: HIT GOOD TRAP at eip = 0x00100032
```

我们看到程序输出 **PASS!!!**的信息，说明实现正确。

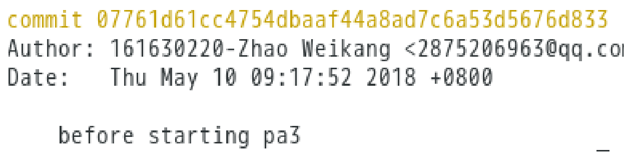
git log 记录



```
zhaoweikang@zhaoweikang:~/ics2017/nanos-lite$ sudo git status
位于分支 pa3
尚未暂存以备提交的变更：
  （使用 "git add <文件>..." 更新要提交的内容）
  （使用 "git checkout -- <文件>..." 丢弃工作区的改动）

        修改：     Makefile
        修改：     src/fs.c
        修改：     src/loader.c
        修改：     src/syscall.c

修改尚未加入提交（使用 "git add" 和/或 "git commit -a"）
```



```
zhaoweikang@zhaoweikang:~/ics2017/nanos-lite$ sudo git commit --allow-empty
[pa3 89d1f54] fix bug for pa3.2
zhaoweikang@zhaoweikang:~/ics2017/nanos-lite$ sudo git log
commit 89d1f54c8569c00d616290535b0cff0bbe2b77b2
Author: 161630220-Zhao Weikang <2875206963@qq.com>
Date:   Fri May 18 16:24:56 2018 +0800

    fix bug for pa3.2

commit a7f5e4a7eb2978ba931e6744ed2e8ef9b5214b77
Author: 161630220-Zhao Weikang <2875206963@qq.com>
Date:   Fri May 18 16:22:37 2018 +0800

    fix bug for pa3.1

commit 8a852335191cf80edb25996de234c03b50e51a72
Author: 161630220-Zhao Weikang <2875206963@qq.com>
Date:   Sun May 13 17:32:32 2018 +0800

    fix bug for pa3.1

commit 07761d61cc4754dbaaf44a8ad7c6a53d5676d833
Author: 161630220-Zhao Weikang <2875206963@qq.com>
Date:   Thu May 10 09:17:52 2018 +0800
```



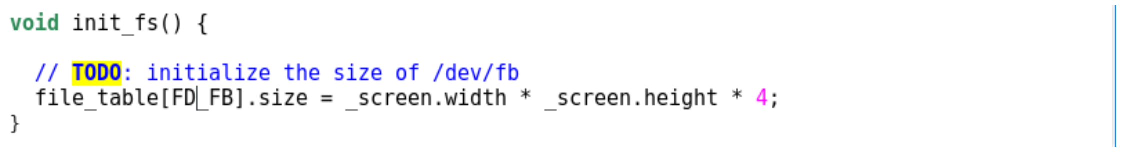
```
commit 07761d61cc4754dbaaf44a8ad7c6a53d5676d833
Author: 161630220-Zhao Weikang <2875206963@qq.com>
Date:   Thu May 10 09:17:52 2018 +0800
```

```
    before starting pa3
```

一切皆文件

把 VGA 显存抽象成文件

首先进入 **nanos-lite/src/fs.c**，在 **init_fs()**函数中，对**/dev/fb**（保存像素信息的 **VGA** 显存抽象后的文件）的大小进行初始化，我们知道在 **nexus-am/am/arch/x86-nemu/src/ioe.c** 中的 **_Screen _screen** 定义了屏幕的 **width** 和 **height**，所以实现如图



```
void init_fs() {
  // TODO: initialize the size of /dev/fb
  file_table[FD_FB].size = _screen.width * _screen.height * 4;
}
```

下面进入 **nanos-lite/src/device.c** 中，实现 **fb_write()**，把 **buf** 中的 **len** 字节写到屏幕上 **offset** 处。首先计算 **col**、**row** 的位置，**raw** 可由偏移量 **offset** 除以屏幕之宽，即 **offset/_screen.width**，**col** 由于不能超过屏幕之宽，对**_screen.width** 取余即可，即 **offset % _screen.width**。**offset**、**len** 每次右移两位，然后调用**_draw_rect()**实现如图

```c
void fb_write(const void *buf, off_t offset, size_t len) {
  int row, col;
  offset /= 4;
  col = offset % _screen.width;
  row = offset / _screen.width;

  _draw_rect(buf, col, row, len/4, 1);
}
```

下面实现 init_device()函数，查看 nexus-am/am/arch/x86-nemu/src/ioe.c 知，屏幕的实际大小为 width：400，height：300，如图

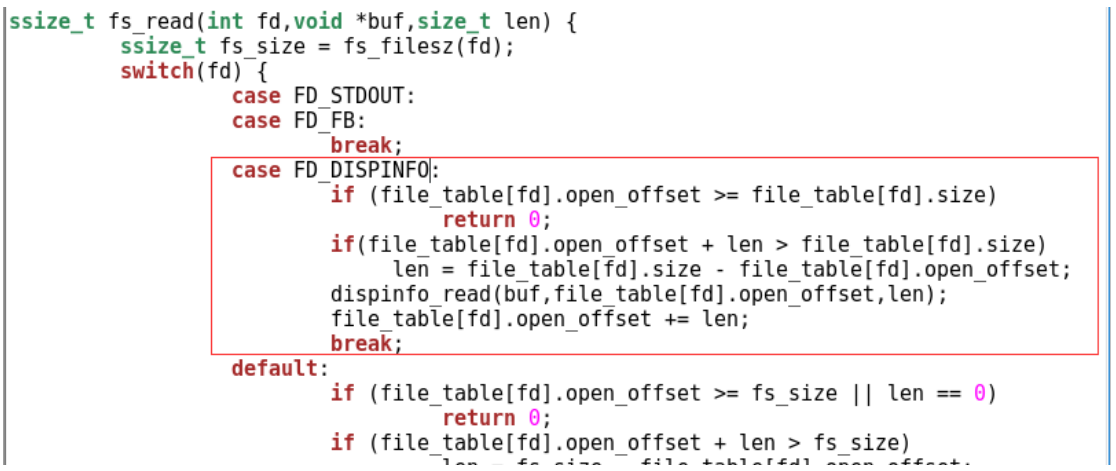
```c
void init_device() {
  _ioe_init();

  // TODO: print the string to array `dispinfo` with the format
  // described in the Navy-apps convention
        sprintf(dispinfo, "WIDTH:%d\nHEIGHT:%d\n", _screen.width, _screen.height);
}
```

下面实现 dispinfo_read()函数，用于把字符串 dispinfo 中 offset 开始的 len 字节写到 buf中，如图

```c
void dispinfo_read(void *buf, off_t offset, size_t len) {
        memcpy(buf, dispinfo + offset, len);
}
```

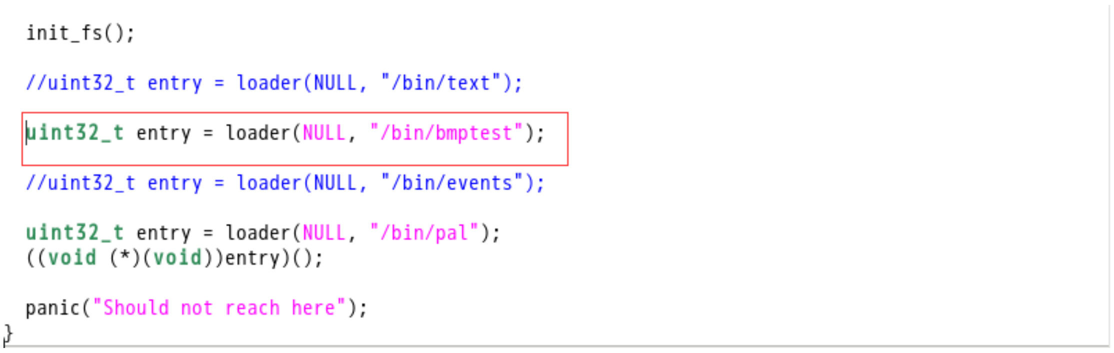
下面在文件系统中添加对/dev/fb 和 /proc/dispinfo 这两个特殊文件的支持，由于 dev/fb 需要支持写操作，/proc/dispinfo 需要支持读操作，进入 nanos-lit/src/fs.c，分别在 fs_write()、fs_read()中，实现之。

```c
ssize_t fs_write(int fd,const void *buf,int len) {
        ssize_t fs_size = fs_filesz(fd);
        switch(fd) {
                case FD_STDOUT:
                case FD_STDERR:
                        for (int i = 0;i < len;i ++) {
                        _putc(((char *)buf)[i]);
                        }
                        break;
                case FD_FB:
                        fb_write(buf, file_table[fd].open_offset,len);
                        file_table[fd].open_offset += len;
                        break;
                default:
                        if(file_table[fd].open_offset >= fs_size)

                            return 0;
```



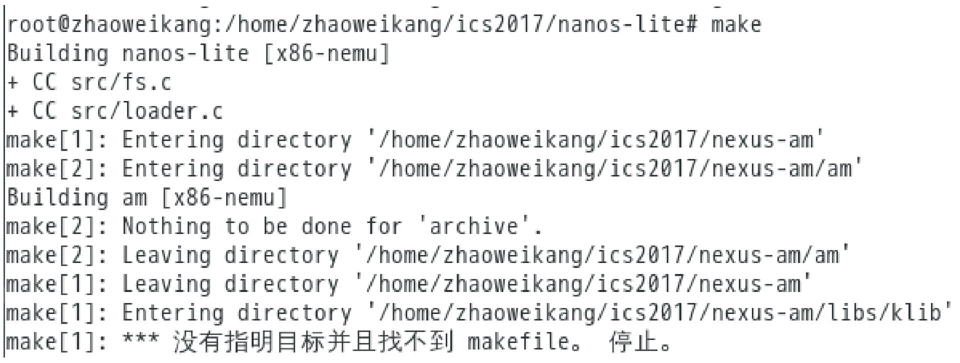
```
ssize_t fs_read(int fd,void *buf,size_t len) {
        ssize_t fs_size = fs_filesz(fd);
        switch(fd) {
                case FD_STDOUT:
                case FD_FB:
                        break;
                case FD_DISPINFO:
                        if (file_table[fd].open_offset >= file_table[fd].size)
                                return 0;
                        if(file_table[fd].open_offset + len > file_table[fd].size)
                            len = file_table[fd].size - file_table[fd].open_offset;
                        dispinfo_read(buf,file_table[fd].open_offset,len);
                        file_table[fd].open_offset += len;
                        break;
                default:
                        if (file_table[fd].open_offset >= fs_size || len == 0)
                                return 0;
                        if (file_table[fd].open_offset + len > fs_size)
                                len = fs_size - file_table[fd].open_offset;
```

下面在 **loader（）**的调用点（**src/main.c**）传入文件**/bin/bmptest,**也就是加载**/bin/bmptest，**如图

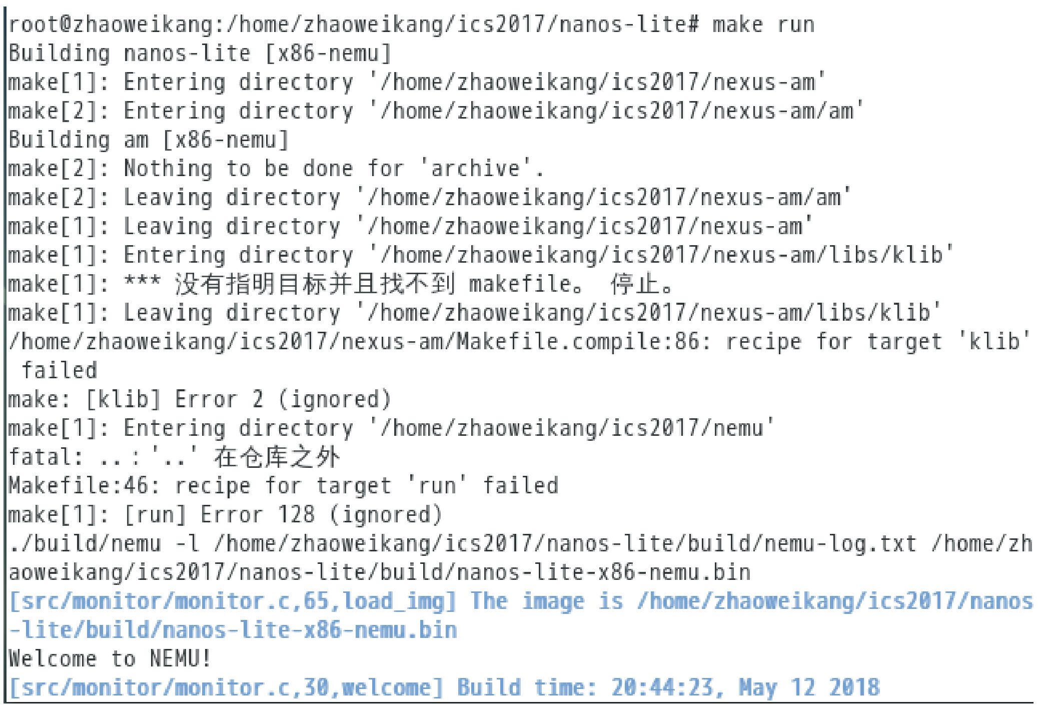


```
  init_fs();

  //uint32_t entry = loader(NULL, "/bin/text");

  uint32_t entry = loader(NULL, "/bin/bmptest");

  //uint32_t entry = loader(NULL, "/bin/events");

  uint32_t entry = loader(NULL, "/bin/pal");
  ((void (*)(void))entry)();

  panic("Should not reach here");
}
```

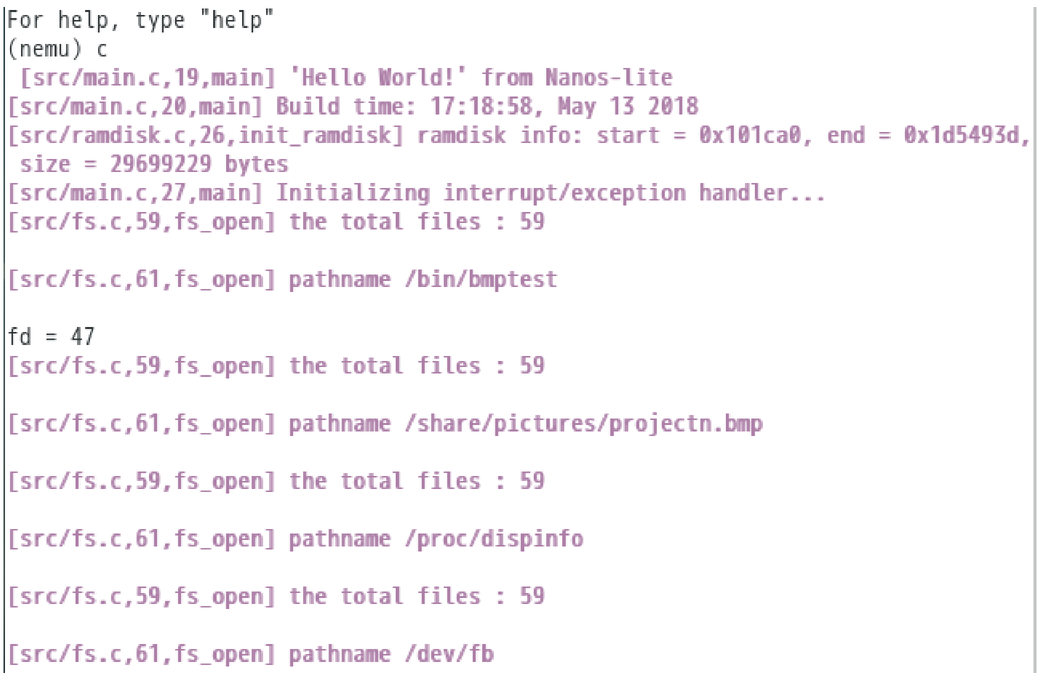
下面执行 **make** 及 **make run** 命令，如图



```
root@zhaoweikang:/home/zhaoweikang/ics2017/nanos-lite# make
Building nanos-lite [x86-nemu]
+ CC src/fs.c
+ CC src/loader.c
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am'
make[2]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/am'
Building am [x86-nemu]
make[2]: Nothing to be done for 'archive'.
make[2]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am/am'
make[1]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am'
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/libs/klib'
make[1]: *** 没有指明目标并且找不到 makefile。 停止。
```



```
root@zhaoweikang:/home/zhaoweikang/ics2017/nanos-lite# make run
Building nanos-lite [x86-nemu]
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am'
make[2]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/am'
Building am [x86-nemu]
make[2]: Nothing to be done for 'archive'.
make[2]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am/am'
make[1]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am'
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/libs/klib'
make[1]: *** 没有指明目标并且找不到 makefile。 停止。
make[1]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am/libs/klib'
/home/zhaoweikang/ics2017/nexus-am/Makefile.compile:86: recipe for target 'klib'
 failed
make: [klib] Error 2 (ignored)
make[1]: Entering directory '/home/zhaoweikang/ics2017/nemu'
fatal: ..: '..' 在仓库之外
Makefile:46: recipe for target 'run' failed
make[1]: [run] Error 128 (ignored)
./build/nemu -l /home/zhaoweikang/ics2017/nanos-lite/build/nemu-log.txt /home/zh
aoweikang/ics2017/nanos-lite/build/nanos-lite-x86-nemu.bin
[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nanos
-lite/build/nanos-lite-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 20:44:23, May 12 2018
```



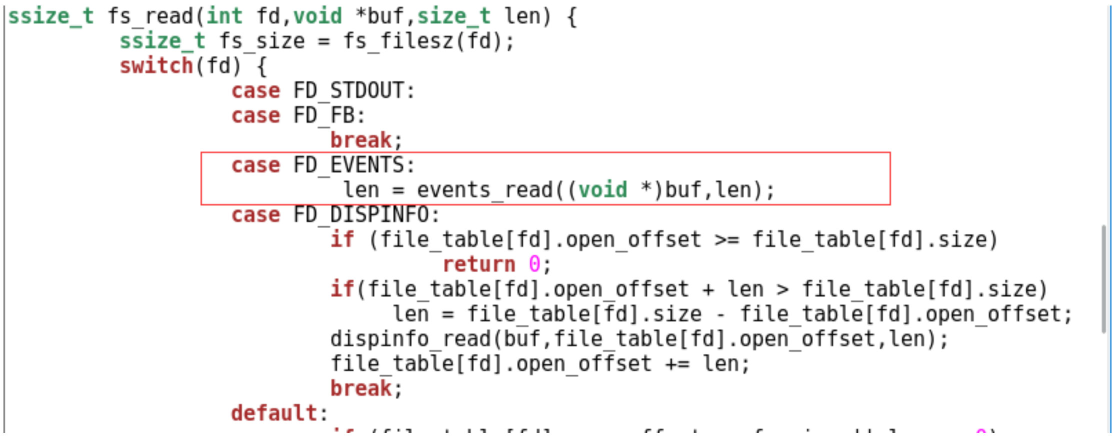
```
For help, type "help"
(nemu) c
 [src/main.c,19,main] 'Hello World!' from Nanos-lite
[src/main.c,20,main] Build time: 17:18:58, May 13 2018
[src/ramdisk.c,26,init_ramdisk] ramdisk info: start = 0x101ca0, end = 0x1d5493d,
 size = 29699229 bytes
[src/main.c,27,main] Initializing interrupt/exception handler...
[src/fs.c,59,fs_open] the total files : 59

[src/fs.c,61,fs_open] pathname /bin/bmptest

fd = 47
[src/fs.c,59,fs_open] the total files : 59

[src/fs.c,61,fs_open] pathname /share/pictures/projectn.bmp

[src/fs.c,59,fs_open] the total files : 59

[src/fs.c,61,fs_open] pathname /proc/dispinfo

[src/fs.c,59,fs_open] the total files : 59

[src/fs.c,61,fs_open] pathname /dev/fb
```

在 **navy-apps/fsimg/share/pictures/projectn.bmp** 中可以查看到 **ProjectN** 的 **Logo**，如图

下面是 **c** 之后的结果，一模一样，说明实现成功。

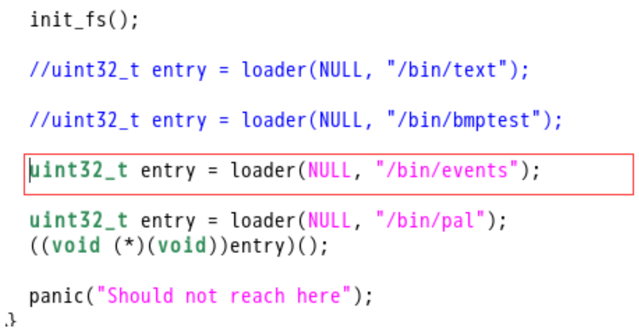
把设备输入抽象成文件
下面进入 nanos-lite/src/device.c，实现 events_read()函数，用于 把事件写入到 buf 中， 最长写入 len 字节， 然后返回写入的实际长度（用 strlen()实现）。调用 nexus-am/am/arch/x-nemu/src/ioe.c 中的_read_key()函数，获取按键消息，我们知道通码是断码+0x8000，所以根据 key 与 0x8000 相与的结果来设置 down(初始化为 false)，并将按键事件写入 buf 中，若无按键事件发生，调用 IOE 中的 API，即_uptime()函数，返回系统启动后的时间，并写入 buf 中，最后返回写入的实际长度，如图

```
size_t events_read(void *buf, size_t len) {
  int key = _read_key();
  bool down = false;
  if (key & 0x8000) {
          key ^=0x8000;
          down = true;
  }
  if (key == _KEY_NONE) {
           unsigned long t = _uptime();
           sprintf(buf,"t %d\n",t);
  }
  else {
            sprintf(buf,"%s %s\n",down ? "kd" : "ku",keyname[key]);
  }
  return strlen(buf);

}
```
下面在文件系统中添加对/dev/events 的支持，由于/dev/events 需要支持读操作，进入 nanos-lite/src/fs.c 中，在 fs_read()函数中，实现之，如图



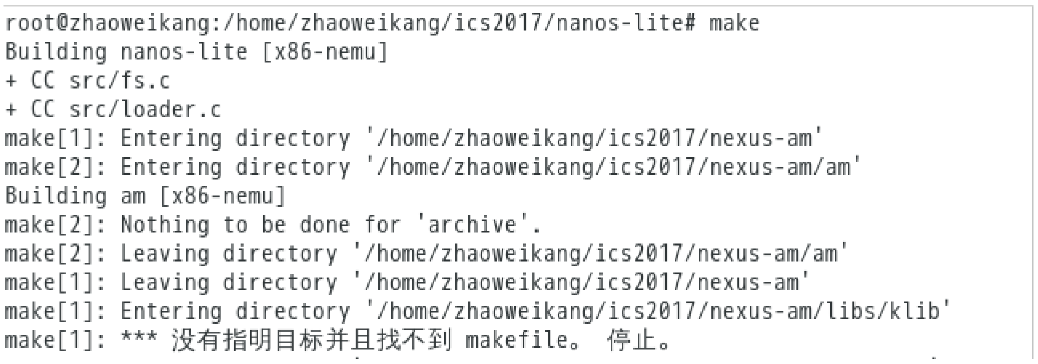
```
ssize_t fs_read(int fd,void *buf,size_t len) {
        ssize_t fs_size = fs_filesz(fd);
        switch(fd) {
                case FD_STDOUT:
                case FD_FB:
                        break;
                case FD_EVENTS:
                         len = events_read((void *)buf,len);
                case FD_DISPINFO:
                        if (file_table[fd].open_offset >= file_table[fd].size)
                                return 0;
                        if(file_table[fd].open_offset + len > file_table[fd].size)
                             len = file_table[fd].size - file_table[fd].open_offset;
                        dispinfo_read(buf,file_table[fd].open_offset,len);
                        file_table[fd].open_offset += len;
                        break;
                default:
```

下面在 nanos-lite/src/loader.c 中传入参数/bin/events，即加载/bin/events，如图

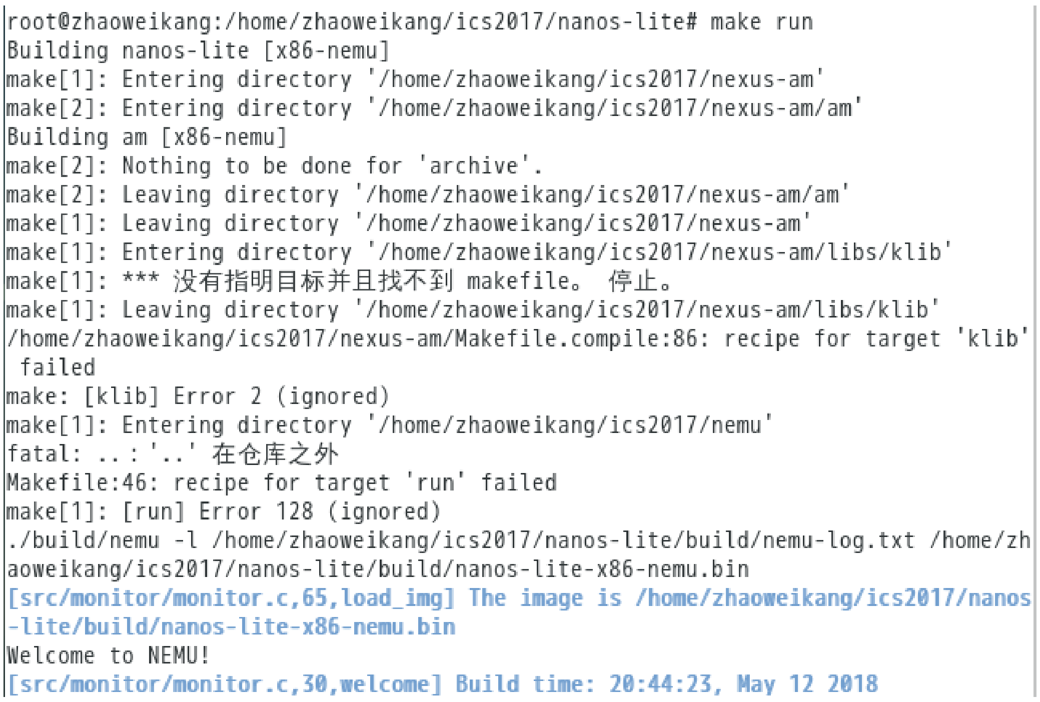


```
  init_fs();

  //uint32_t entry = loader(NULL, "/bin/text");

  //uint32_t entry = loader(NULL, "/bin/bmptest");

  uint32_t entry = loader(NULL, "/bin/events");

  uint32_t entry = loader(NULL, "/bin/pal");
  ((void (*)(void))entry)();

  panic("Should not reach here");
}
```

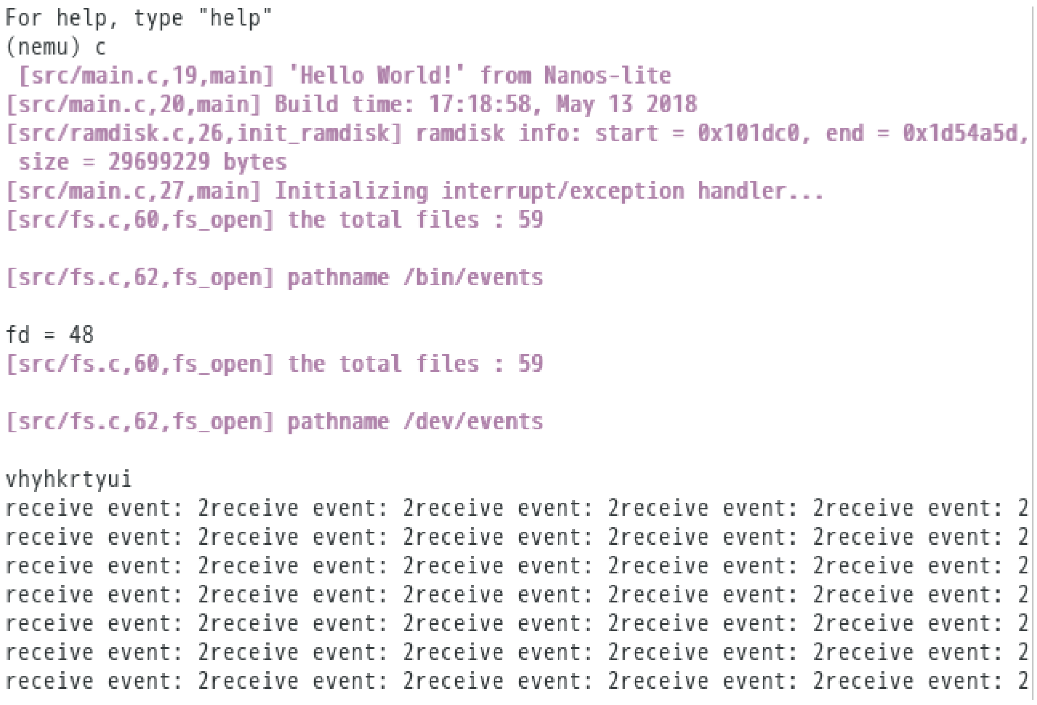
下面执行 make 及 make run 命令，如图



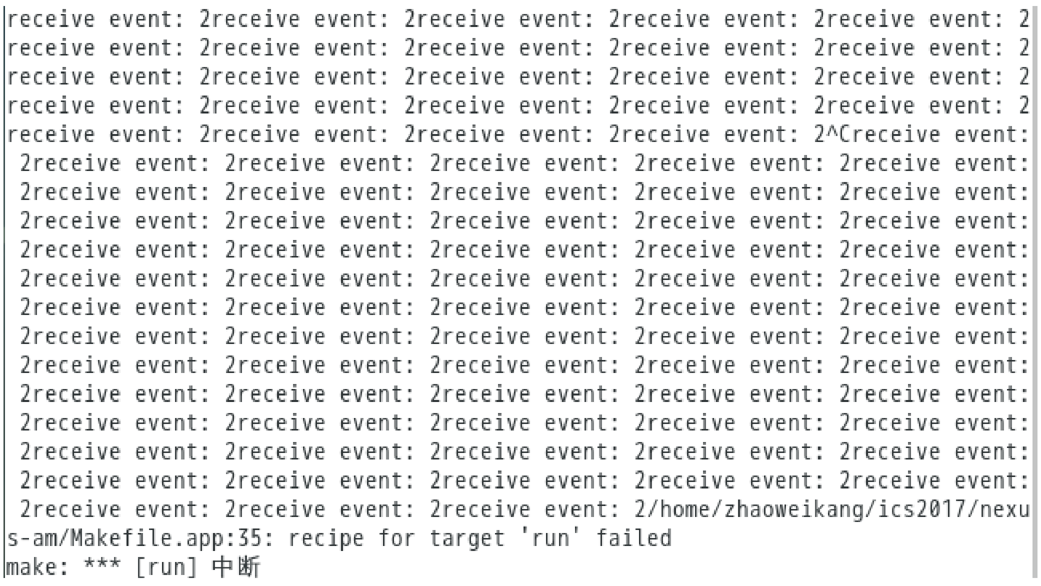
```
root@zhaoweikang:/home/zhaoweikang/ics2017/nanos-lite# make
Building nanos-lite [x86-nemu]
+ CC src/fs.c
+ CC src/loader.c
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am'
make[2]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/am'
Building am [x86-nemu]
make[2]: Nothing to be done for 'archive'.
make[2]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am/am'
make[1]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am'
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/libs/klib'
make[1]: *** 没有指明目标并且找不到 makefile。 停止。
```



```
root@zhaoweikang:/home/zhaoweikang/ics2017/nanos-lite# make run
Building nanos-lite [x86-nemu]
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am'
make[2]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/am'
Building am [x86-nemu]
make[2]: Nothing to be done for 'archive'.
make[2]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am/am'
make[1]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am'
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/libs/klib'
make[1]: *** 没有指明目标并且找不到 makefile。 停止。
make[1]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am/libs/klib'
/home/zhaoweikang/ics2017/nexus-am/Makefile.compile:86: recipe for target 'klib'
 failed
make: [klib] Error 2 (ignored)
make[1]: Entering directory '/home/zhaoweikang/ics2017/nemu'
fatal: ..：'..' 在仓库之外
Makefile:46: recipe for target 'run' failed
make[1]: [run] Error 128 (ignored)
./build/nemu -l /home/zhaoweikang/ics2017/nanos-lite/build/nemu-log.txt /home/zh
aoweikang/ics2017/nanos-lite/build/nanos-lite-x86-nemu.bin
[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nanos
-lite/build/nanos-lite-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 20:44:23, May 12 2018
```



```
For help, type "help"
(nemu) c
 [src/main.c,19,main] 'Hello World!' from Nanos-lite
[src/main.c,20,main] Build time: 17:18:58, May 13 2018
[src/ramdisk.c,26,init_ramdisk] ramdisk info: start = 0x101dc0, end = 0x1d54a5d,
 size = 29699229 bytes
[src/main.c,27,main] Initializing interrupt/exception handler...
[src/fs.c,60,fs_open] the total files : 59

[src/fs.c,62,fs_open] pathname /bin/events

fd = 48
[src/fs.c,60,fs_open] the total files : 59

[src/fs.c,62,fs_open] pathname /dev/events

vhyhkrtyui
receive event: 2receive event: 2receive event: 2receive event: 2receive event: 2
receive event: 2receive event: 2receive event: 2receive event: 2receive event: 2
receive event: 2receive event: 2receive event: 2receive event: 2receive event: 2
receive event: 2receive event: 2receive event: 2receive event: 2receive event: 2
receive event: 2receive event: 2receive event: 2receive event: 2receive event: 2
receive event: 2receive event: 2receive event: 2receive event: 2receive event: 2
receive event: 2receive event: 2receive event: 2receive event: 2receive event: 2
```



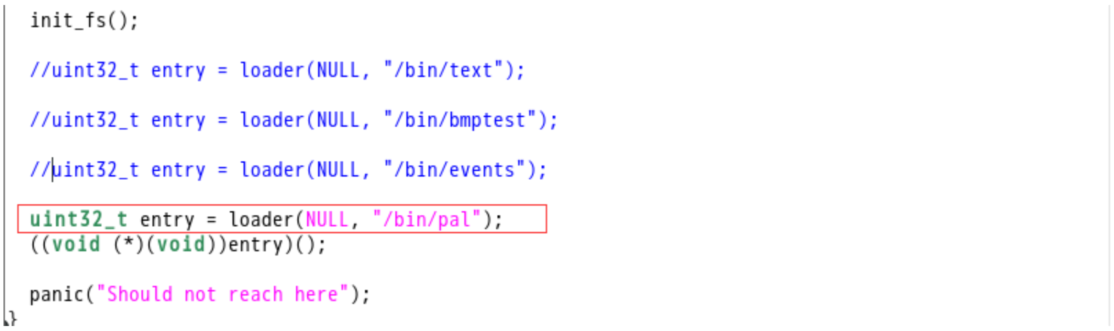
```
receive event: 2receive event: 2receive event: 2receive event: 2receive event: 2
receive event: 2receive event: 2receive event: 2receive event: 2receive event: 2
receive event: 2receive event: 2receive event: 2receive event: 2receive event: 2
receive event: 2receive event: 2receive event: 2receive event: 2receive event: 2
receive event: 2receive event: 2receive event: 2receive event: 2^Creceive event:
 2receive event: 2receive event: 2receive event: 2receive event: 2receive event:
 2receive event: 2receive event: 2receive event: 2receive event: 2receive event:
 2receive event: 2receive event: 2receive event: 2receive event: 2receive event:
 2receive event: 2receive event: 2receive event: 2receive event: 2receive event:
 2receive event: 2receive event: 2receive event: 2receive event: 2receive event:
 2receive event: 2receive event: 2receive event: 2receive event: 2receive event:
 2receive event: 2receive event: 2receive event: 2receive event: 2receive event:
 2receive event: 2receive event: 2receive event: 2receive event: 2receive event:
 2receive event: 2receive event: 2receive event: 2receive event: 2receive event:
 2receive event: 2receive event: 2receive event: 2receive event: 2receive event:
 2receive event: 2receive event: 2receive event: 2receive event: 2receive event:
 2receive event: 2receive event: 2receive event: 2receive event: 2receive event:
 2receive event: 2receive event: 2receive event: 2/home/zhaoweikang/ics2017/nexu
s-am/Makefile.app:35: recipe for target 'run' failed
make: *** [run] 中断
```

由运行结果知，我依次按下了 **vhyhkrtyui** 这些键（按键事件），同时输出了时间事件信息，按 ctrl+c 结束程序运行。

运行仙剑奇侠传

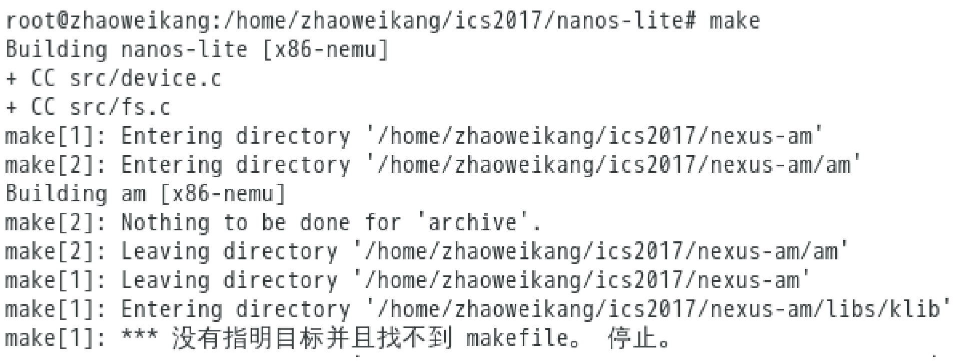
在 NEMU 中运行仙剑奇侠传

下面现在 loader（）的调用点（src/main.c）中传入参数/bin/pal，即加载/bin/pal，如图

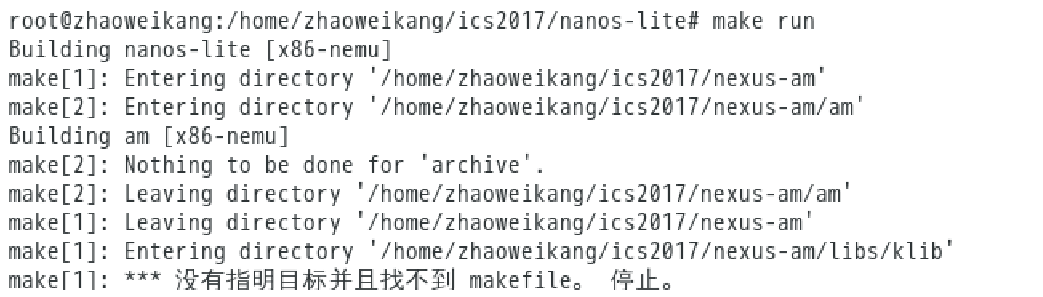


```
  init_fs();

  //uint32_t entry = loader(NULL, "/bin/text");

  //uint32_t entry = loader(NULL, "/bin/bmptest");

  //uint32_t entry = loader(NULL, "/bin/events");

  uint32_t entry = loader(NULL, "/bin/pal");
  ((void (*)(void))entry)();

  panic("Should not reach here");
}
```

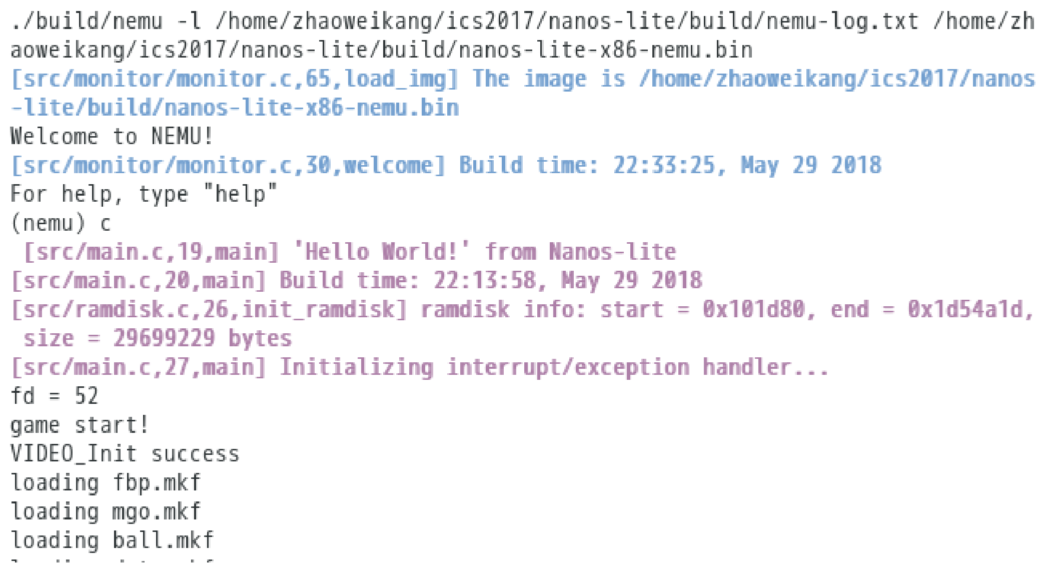
下面执行 make update、make 及 make run 命令，如图



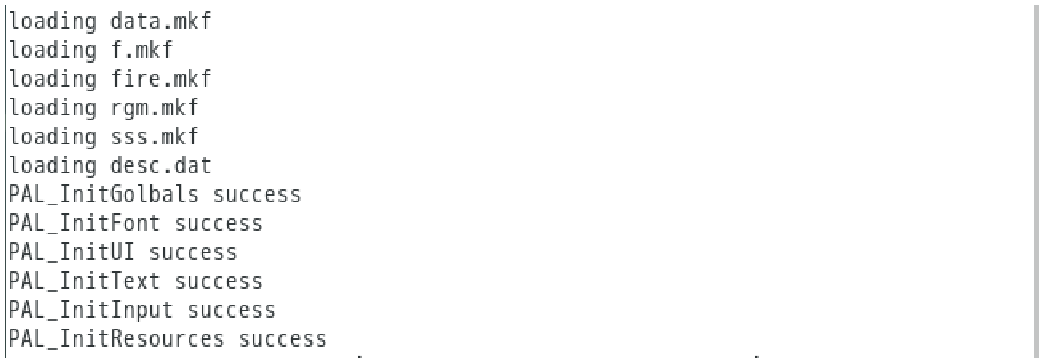
```
root@zhaoweikang:/home/zhaoweikang/ics2017/nanos-lite# make
Building nanos-lite [x86-nemu]
+ CC src/device.c
+ CC src/fs.c
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am'
make[2]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/am'
Building am [x86-nemu]
make[2]: Nothing to be done for 'archive'.
make[2]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am/am'
make[1]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am'
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/libs/klib'
make[1]: *** 没有指明目标并且找不到 makefile。 停止。
```



```
root@zhaoweikang:/home/zhaoweikang/ics2017/nanos-lite# make run
Building nanos-lite [x86-nemu]
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am'
make[2]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/am'
Building am [x86-nemu]
make[2]: Nothing to be done for 'archive'.
make[2]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am/am'
make[1]: Leaving directory '/home/zhaoweikang/ics2017/nexus-am'
make[1]: Entering directory '/home/zhaoweikang/ics2017/nexus-am/libs/klib'
make[1]: *** 没有指明目标并且找不到 makefile。 停止。
```



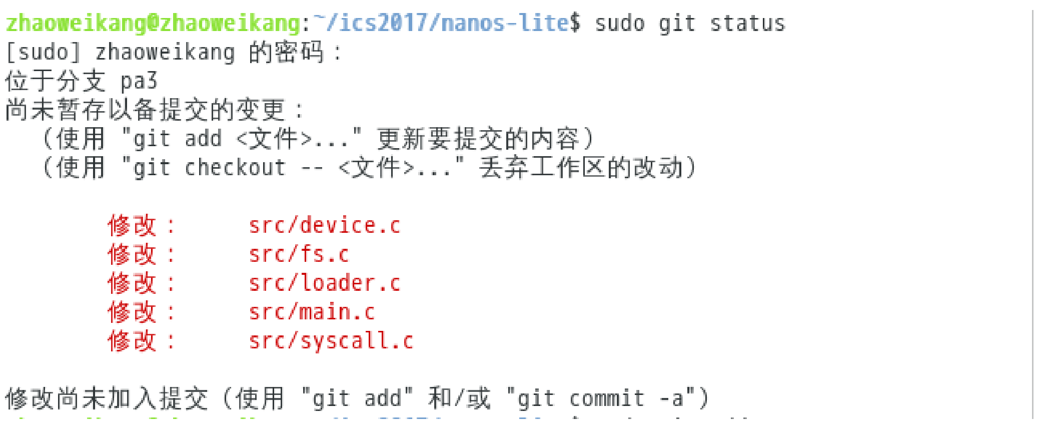
```
./build/nemu -l /home/zhaoweikang/ics2017/nanos-lite/build/nemu-log.txt /home/zh
aoweikang/ics2017/nanos-lite/build/nanos-lite-x86-nemu.bin
[src/monitor/monitor.c,65,load_img] The image is /home/zhaoweikang/ics2017/nanos
-lite/build/nanos-lite-x86-nemu.bin
Welcome to NEMU!
[src/monitor/monitor.c,30,welcome] Build time: 22:33:25, May 29 2018
For help, type "help"
(nemu) c
 [src/main.c,19,main] 'Hello World!' from Nanos-lite
[src/main.c,20,main] Build time: 22:13:58, May 29 2018
[src/ramdisk.c,26,init_ramdisk] ramdisk info: start = 0x101d80, end = 0x1d54a1d,
 size = 29699229 bytes
[src/main.c,27,main] Initializing interrupt/exception handler...
fd = 52
game start!
VIDEO_Init success
loading fbp.mkf
loading mgo.mkf
loading ball.mkf
```



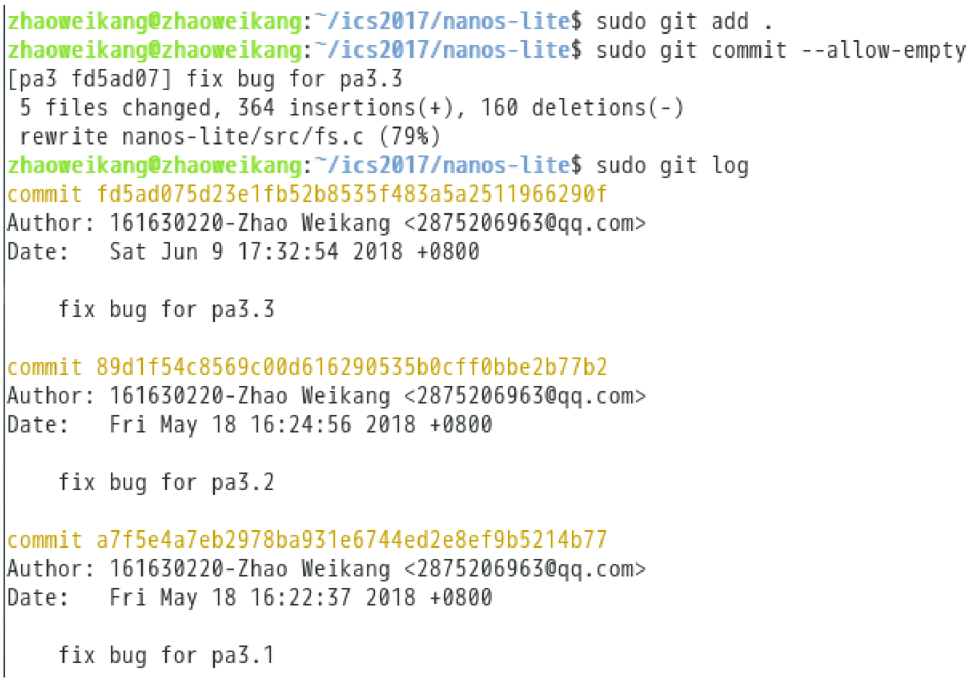
```
loading data.mkf
loading f.mkf
loading fire.mkf
loading rgm.mkf
loading sss.mkf
loading desc.dat
PAL_InitGolbals success
PAL_InitFont success
PAL_InitUI success
PAL_InitText success
PAL_InitInput success
PAL_InitResources success
```

下面是 **c** 之后的仙剑奇侠传其中两张运行结果，如图

git log 记录



```
zhaoweikang@zhaoweikang:~/ics2017/nanos-lite$ sudo git status
[sudo] zhaoweikang 的密码：
位于分支 pa3
尚未暂存以备提交的变更：
  （使用 "git add <文件>..." 更新要提交的内容）
  （使用 "git checkout -- <文件>..." 丢弃工作区的改动）

	修改：     src/device.c
	修改：     src/fs.c
	修改：     src/loader.c
	修改：     src/main.c
	修改：     src/syscall.c

修改尚未加入提交（使用 "git add" 和/或 "git commit -a"）
```



```
zhaoweikang@zhaoweikang:~/ics2017/nanos-lite$ sudo git add .
zhaoweikang@zhaoweikang:~/ics2017/nanos-lite$ sudo git commit --allow-empty
[pa3 fd5ad07] fix bug for pa3.3
 5 files changed, 364 insertions(+), 160 deletions(-)
 rewrite nanos-lite/src/fs.c (79%)
zhaoweikang@zhaoweikang:~/ics2017/nanos-lite$ sudo git log
commit fd5ad075d23e1fb52b8535f483a5a2511966290f
Author: 161630220-Zhao Weikang <2875206963@qq.com>
Date:   Sat Jun 9 17:32:54 2018 +0800

    fix bug for pa3.3

commit 89d1f54c8569c00d616290535b0cff0bbe2b77b2
Author: 161630220-Zhao Weikang <2875206963@qq.com>
Date:   Fri May 18 16:24:56 2018 +0800

    fix bug for pa3.2

commit a7f5e4a7eb2978ba931e6744ed2e8ef9b5214b77
Author: 161630220-Zhao Weikang <2875206963@qq.com>
Date:   Fri May 18 16:22:37 2018 +0800

    fix bug for pa3.1
```

必答题

文件读写的具体过程仙剑奇侠传中有以下行为:在 navy-apps/apps/pal/src/global/global.c 的 PAL_LoadGame()中通过 fread()读取游戏存档，在 navy-apps/apps/pal/src/hal/hal.c 的 redraw()中通过 NDL_DrawRect()更新屏幕请结合代码解释仙剑奇侠传，库函数，libos,Nanos-lite，AM，NEMU 是如何相互协助，来分别完成游戏存档的读取和屏幕的更新。

答：库函数提供一些运行客户程序所需的函数的支持（检索），libos 中提供一些系统调用（如_syscall_()的系统调用接口、_read()、_write()、_exit()等），仙剑奇侠传是 Nanos-lite 的客户程序，运行在 Nanos-lite 之上，Nanos-lite 是 NEMU 的客户程序，运行在 AM 之上。