操作系统课设之进程

小组主要工作

可视化以下过程：

进程初始化

进程创建

进程切换

进程结束

进程通信

我的工作

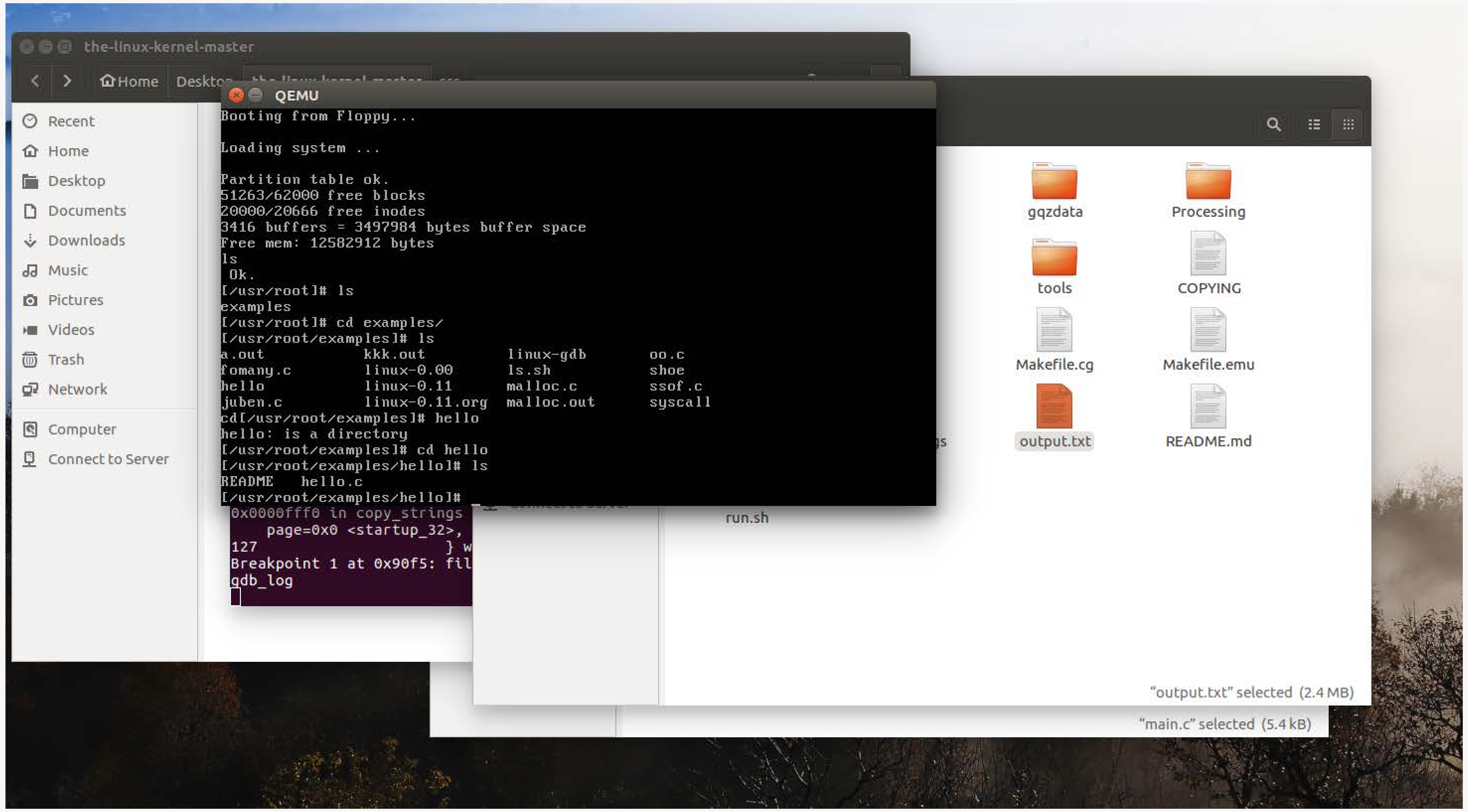
进程初始化

进程通信

Mr.T

实验环境：

Linux-lab-0.11



```
Booting from Floppy...

Loading system ...

Partition table ok.
51263/62000 free blocks
20000/20666 free inodes
3416 buffers = 3497984 bytes buffer space
Free mem: 12582912 bytes
ls
 Ok.
[/usr/root]# ls
examples
[/usr/root]# cd examples/
[/usr/root/examples]# ls
a.out           kkk.out         linux-gdb       oo.c
fomany.c        linux-0.00      ls.sh           shoe
hello           linux-0.11      malloc.c        ssof.c
juben.c         linux-0.11.org  malloc.out      syscall
cd[/usr/root/examples]# hello
hello: is a directory
[/usr/root/examples]# cd hello
[/usr/root/examples/hello]# ls
README   hello.c
[/usr/root/examples/hello]#
```

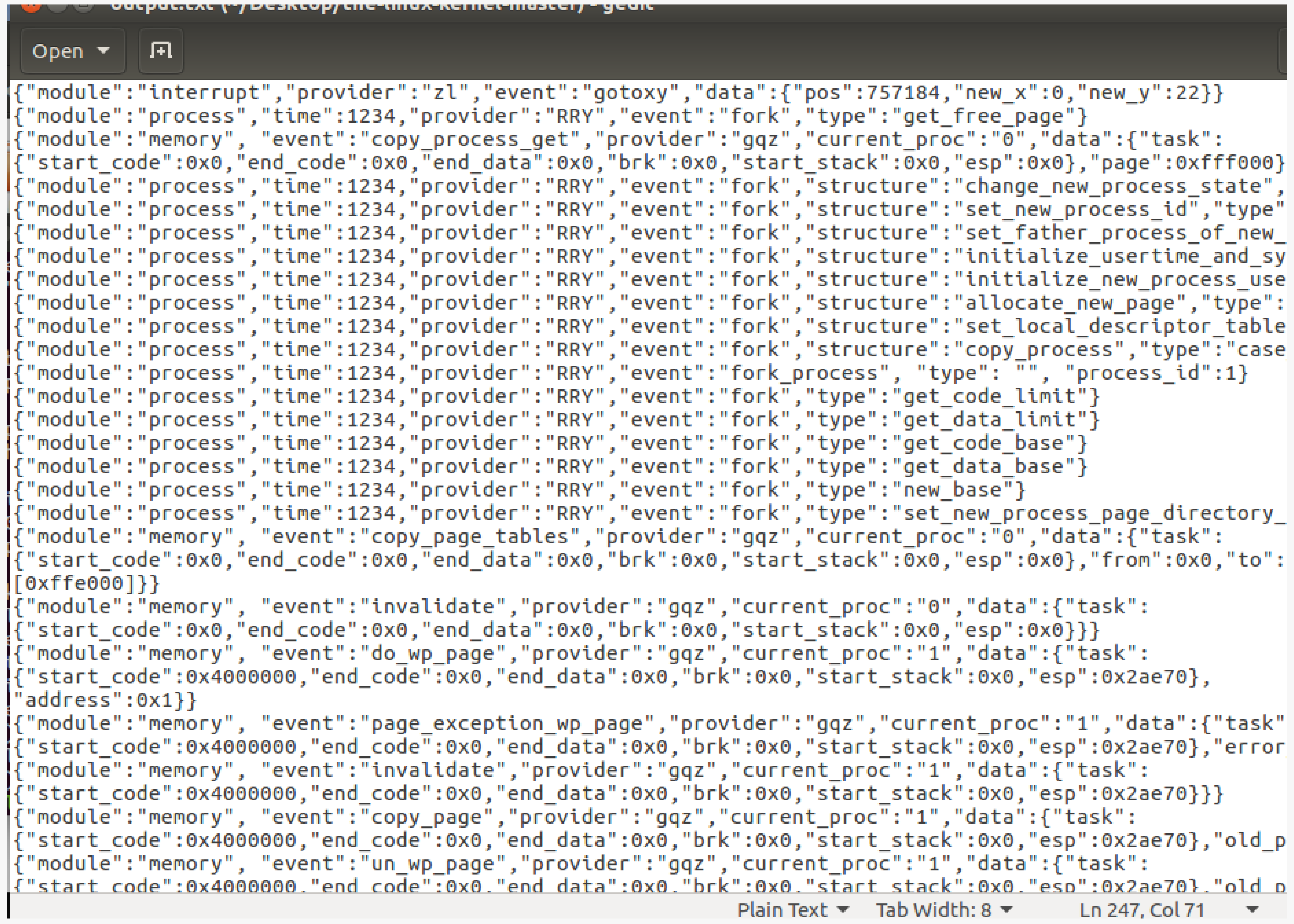
```
0x0000fff0 in copy_strings
    page=0x0 <startup_32>,
127                        } w
Breakpoint 1 at 0x90f5: fil
gdb_log
```

Mr.T

数据部分

使用WTH学长给出的JSON数据封装格式：

```
{
"module": "memory",
"event": "malloc",
"provider": “ZT",
"time": 2437,
"data": {
        "addr": 72000,
        "len": 512
    }
}
```



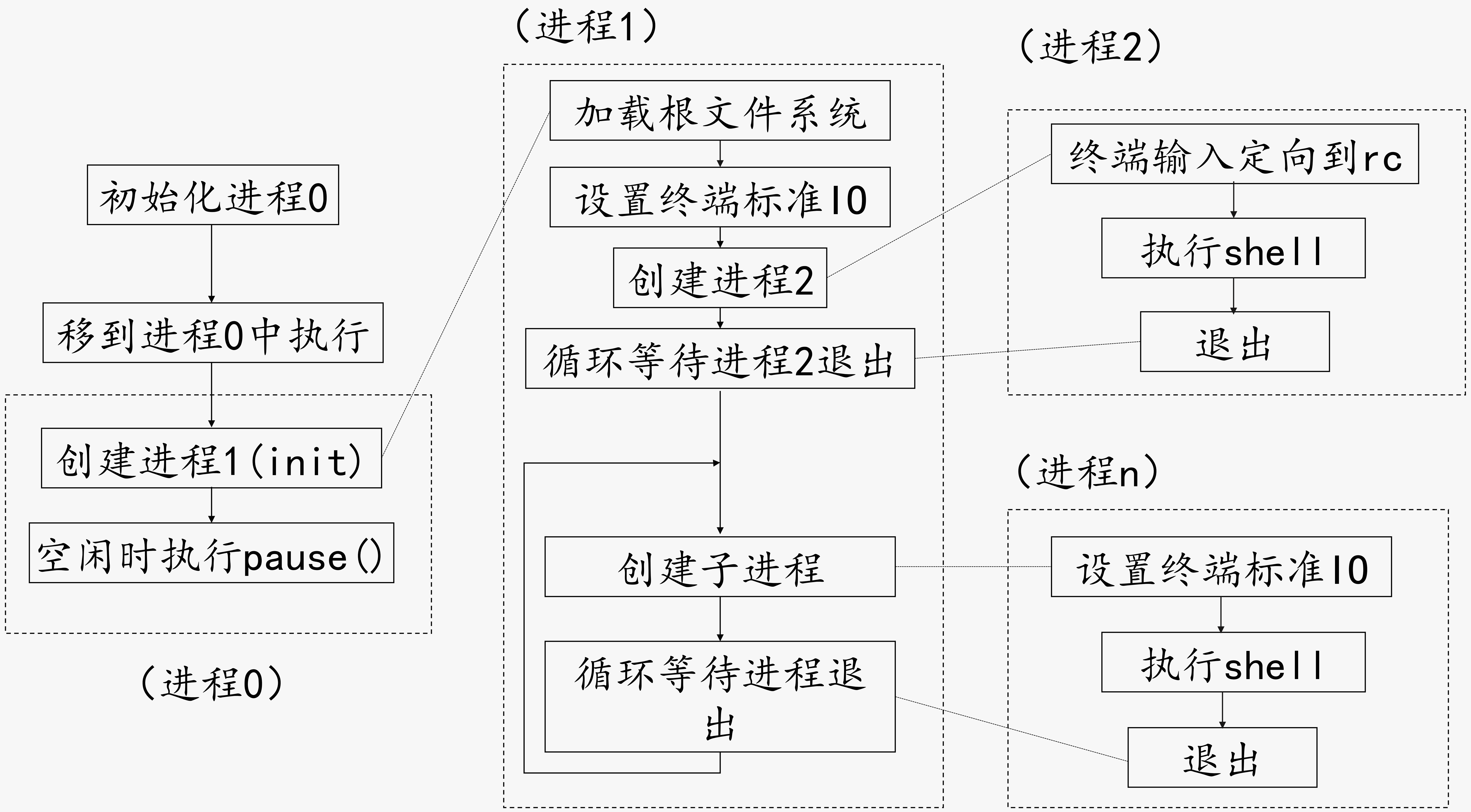
```
{"module":"interrupt","provider":"zl","event":"gotoxy","data":{"pos":757184,"new_x":0,"new_y":22}}
{"module":"process","time":1234,"provider":"RRY","event":"fork","type":"get_free_page"}
{"module":"memory", "event":"copy_process_get","provider":"gqz","current_proc":"0","data":{"task":
{"start_code":0x0,"end_code":0x0,"end_data":0x0,"brk":0x0,"start_stack":0x0,"esp":0x0},"page":0xfff000}
{"module":"process","time":1234,"provider":"RRY","event":"fork","structure":"change_new_process_state",
{"module":"process","time":1234,"provider":"RRY","event":"fork","structure":"set_new_process_id","type"
{"module":"process","time":1234,"provider":"RRY","event":"fork","structure":"set_father_process_of_new_
{"module":"process","time":1234,"provider":"RRY","event":"fork","structure":"initialize_usertime_and_sy
{"module":"process","time":1234,"provider":"RRY","event":"fork","structure":"initialize_new_process_use
{"module":"process","time":1234,"provider":"RRY","event":"fork","structure":"allocate_new_page","type":
{"module":"process","time":1234,"provider":"RRY","event":"fork","structure":"set_local_descriptor_table
{"module":"process","time":1234,"provider":"RRY","event":"fork","structure":"copy_process","type":"case
{"module":"process","time":1234,"provider":"RRY","event":"fork_process", "type": "", "process_id":1}
{"module":"process","time":1234,"provider":"RRY","event":"fork","type":"get_code_limit"}
{"module":"process","time":1234,"provider":"RRY","event":"fork","type":"get_data_limit"}
{"module":"process","time":1234,"provider":"RRY","event":"fork","type":"get_code_base"}
{"module":"process","time":1234,"provider":"RRY","event":"fork","type":"get_data_base"}
{"module":"process","time":1234,"provider":"RRY","event":"fork","type":"new_base"}
{"module":"process","time":1234,"provider":"RRY","event":"fork","type":"set_new_process_page_directory_
{"module":"memory", "event":"copy_page_tables","provider":"gqz","current_proc":"0","data":{"task":
{"start_code":0x0,"end_code":0x0,"end_data":0x0,"brk":0x0,"start_stack":0x0,"esp":0x0},"from":0x0,"to":
[0xffe000]}}
{"module":"memory", "event":"invalidate","provider":"gqz","current_proc":"0","data":{"task":
{"start_code":0x0,"end_code":0x0,"end_data":0x0,"brk":0x0,"start_stack":0x0,"esp":0x0}}}
{"module":"memory", "event":"do_wp_page","provider":"gqz","current_proc":"1","data":{"task":
{"start_code":0x4000000,"end_code":0x0,"end_data":0x0,"brk":0x0,"start_stack":0x0,"esp":0x2ae70},
"address":0x1}}
{"module":"memory", "event":"page_exception_wp_page","provider":"gqz","current_proc":"1","data":{"task"
{"start_code":0x4000000,"end_code":0x0,"end_data":0x0,"brk":0x0,"start_stack":0x0,"esp":0x2ae70},"error
{"module":"memory", "event":"invalidate","provider":"gqz","current_proc":"1","data":{"task":
{"start_code":0x4000000,"end_code":0x0,"end_data":0x0,"brk":0x0,"start_stack":0x0,"esp":0x2ae70}}}
{"module":"memory", "event":"copy_page","provider":"gqz","current_proc":"1","data":{"task":
{"start_code":0x4000000,"end_code":0x0,"end_data":0x0,"brk":0x0,"start_stack":0x0,"esp":0x2ae70},"old_p
{"module":"memory", "event":"un_wp_page","provider":"gqz","current_proc":"1","data":{"task":
{"start_code":0x4000000,"end_code":0x0,"end_data":0x0,"brk":0x0,"start_stack":0x0,"esp":0x2ae70},"old_p
```

样例展示

任何一个事件都包含 module、event、provider、time 和 data 这五个字段，其中 module、event 和 provider 的类型为 string，time 的类型为 number，data 的类型为 object。
在进程初始化时，module为memory，在进程间通信时，module为memory和fs。

进程初始化

进程0：0号进程拥有的所有信息和资源都是强制设置的。
信息包括：创建进程0运行时所需的所有信息和调度0号进程的执行。



Mr.T

进程1 : init进程，由进程0创建，系统启动的第一个用户级进程，是所有其它进程的父进程，引导用户空间服务。

进程2 : kthreadd，由进程1创建，用于内核线程管理。

进程3 : migration，由进程2创建，用于进程在不同的CPU间迁移。

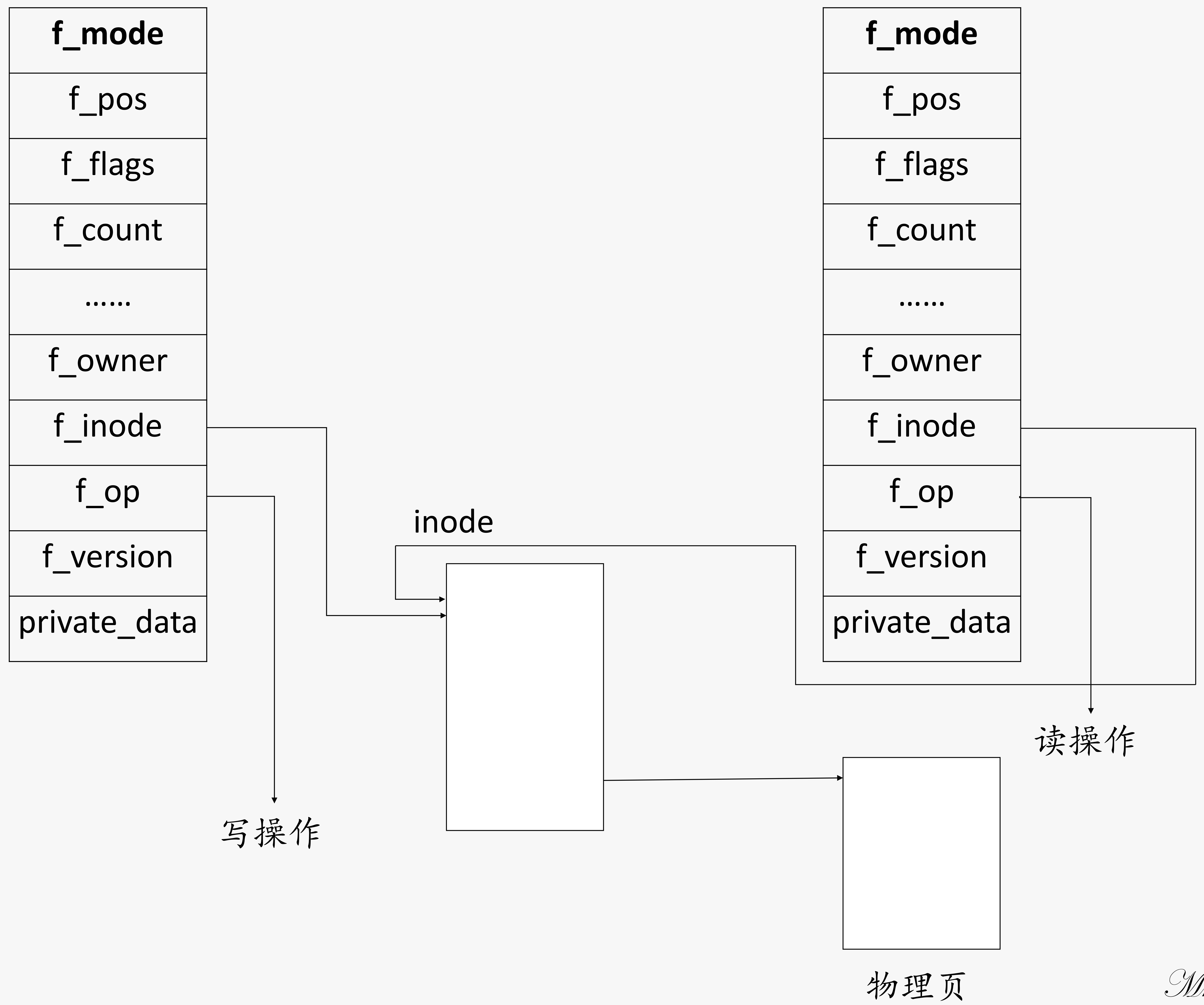
进程4 : ksoftirqd，由进程1创建，内核里的软中断守护线程，用于在系统空闲时定时处理软中断事务。

进程5 : watchdog，由进程4创建，此进程是看门狗进程，用于监听内核异常。当系统出现宕机，可以利用watchdog进程将宕机时的一些堆栈信息写入指定文件，用于事后分析宕机的原因。

进程通信

借助文件系统中file结构和索引节点inode

通过共享物理内存页来实现



Mr.T

Thank you!