Linux pipe and subshell

@k1ic

chenkai116@gmail.com

缘起

管道方式

```
[redacted]:/home/chenkai3/tmp$ cat get_file_via_pipe.sh
#!/bin/bash

function get_content_via_pipe() {
    res_str='';
    file_path='./data.txt';

    cat ${file_path} | while read line
    do
        res_str=${res_str}[br]${line};
        echo ${res_str};
    done;
    echo "The Result: "${res_str};
}

get_content_via_pipe
[redacted]:/home/chenkai3/tmp$ sh get_file_via_pipe.sh
[br]qqq
[br]qqq[br]aaa
[br]qqq[br]aaa[br]zzz
The Result:
```

重定向方式

```
[redacted]:/home/chenkai3/tmp$ cat get_file_via_redirect.sh
#!/bin/bash

function get_file_via_redirect() {
    res_str='';
    file_path='./data.txt';

    while read line
    do
        res_str=${res_str}[br]${line};
        echo ${res_str};
    done < ${file_path};
    echo "The Result: "${res_str};
}

get_file_via_redirect
[redacted]:/home/chenkai3/tmp$ sh get_file_via_redirect.sh
[br]qqq
[br]qqq[br]aaa
[br]qqq[br]aaa[br]zzz
The Result: [br]qqq[br]aaa[br]zzz
```

系统调用对比

```
 1 Process 2153 attached - interrupt to quit
 2 wait4(-1, [{WIFEXITED(s) && WEXITSTATUS(s) == 0}],
   0, NULL) = 2154
 3 rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
 4 --- SIGCHLD (Child exited) @ 0 (0) ---
 5 wait4(-1, 0x7fffeb193fd4, WNOHANG, NULL) = -1 ECHIL
   D (No child processes)
 6 rt_sigreturn(0xffffffffffffffff)         = 0
 7 rt_sigaction(SIGINT, {SIG_DFL, [], SA_RESTORER, 0x3
   80fe302d0}, {0x436c60, [], SA_RESTORER, 0x380fe302d
   0}, 8) = 0
 8 rt_sigprocmask(SIG_BLOCK, NULL, [], 8)   = 0
 9 read(255, "get_file_via_pipe\n", 273)    = 18
10 rt_sigprocmask(SIG_BLOCK, NULL, [], 8)   = 0
11 rt_sigprocmask(SIG_BLOCK, [CHLD], [], 8) = 0
12 pipe([3, 4])                             = 0
13 rt_sigprocmask(SIG_BLOCK, [INT CHLD], [CHLD], 8) =
   0
14 clone(child_stack=0, flags=CLONE_CHILD_CLEARTID|CLO
   NE_CHILD_SETTID|SIGCHLD, child_tidptr=0x2b1bcb347e6
   0) = 2162
15 rt_sigprocmask(SIG_SETMASK, [CHLD], NULL, 8) = 0
16 close(4)                                 = 0
17 close(4)                                 = -1 EBADF
   (Bad file descriptor)
18 rt_sigprocmask(SIG_BLOCK, [INT CHLD], [CHLD], 8) =
   0
19 clone(child_stack=0, flags=CLONE_CHILD_CLEARTID|CLO
   NE_CHILD_SETTID|SIGCHLD, child_tidptr=0x2b1bcb347e6
   0) = 2163
20 rt_sigprocmask(SIG_SETMASK, [CHLD], NULL, 8) = 0
21 close(3)                                 = 0
22 rt_sigprocmask(SIG_BLOCK, [CHLD], [CHLD], 8) = 0
```

```
 1 Process 3694 attached - interrupt to quit
 2 wait4(-1, [{WIFEXITED(s) && WEXITSTATUS(s) == 0}],
   0, NULL) = 3695
 3 rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
 4 --- SIGCHLD (Child exited) @ 0 (0) ---
 5 wait4(-1, 0x7fff8960a224, WNOHANG, NULL) = -1 ECHIL
   D (No child processes)
 6 rt_sigreturn(0xffffffffffffffff)         = 0
 7 rt_sigaction(SIGINT, {SIG_DFL, [], SA_RESTORER, 0x3
   80fe302d0}, {0x436c60, [], SA_RESTORER, 0x380fe302d
   0}, 8) = 0
 8 rt_sigprocmask(SIG_BLOCK, NULL, [], 8)   = 0
 9 read(255, "get_file_via_redirect\n", 277) = 22
10 rt_sigprocmask(SIG_BLOCK, NULL, [], 8)   = 0
11 open("./data.txt", O_RDONLY)             = 3
12 fcntl(0, F_GETFD)                        = 0
13 fcntl(0, F_DUPFD, 10)                    = 10
14 fcntl(0, F_GETFD)                        = 0
15 fcntl(10, F_SETFD, FD_CLOEXEC)           = 0
16 dup2(3, 0)                               = 0
17 close(3)                                 = 0
18 ioctl(0, SNDCTL_TMR_TIMEBASE or TCGETS, 0x7fff89609
   df0) = -1 ENOTTY (Inappropriate ioctl for device)
19 lseek(0, 0, SEEK_CUR)                    = 0
20 read(0, "qqq\naaa\nzzz\n", 128)          = 12
21 lseek(0, -8, SEEK_CUR)                   = 4
22 rt_sigprocmask(SIG_BLOCK, [CHLD], [], 8) = 0
23 rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
24 open(".", O_RDONLY|O_NONBLOCK|O_DIRECTORY) = 3
25 fcntl(3, F_SETFD, FD_CLOEXEC)           = 0
26 getdents(3, /* 9 entries */, 32768)     = 296
27 getdents(3, /* 0 entries */, 32768)     = 0
28 close(3)                                 = 0
```

VFS inode

VFS inode

- Sector & Block

  1Block = 8Sector = 8 * 0.5K = 4K

  (Linux OS Default Memery PgaeSize = 4K)

- File = Blocks + inode

- Inode = file size + uid + gid + rwx permission
  + time stamp + link total + block Loc

VFS inode

```
:/home/chenkai3/tmp$ ll -ai
total 12
1527658 drwxr-xr-x 2 root    root    4096 Mar 25 15:25 .
1527545 drwx------ 9 jianxin jianxin 4096 Mar 25 14:58 ..
 419875 -rw-r--r-- 1 root    root       4 Mar 25 15:25 test.txt
:/home/chenkai3/tmp$ cat test.txt
123
:/home/chenkai3/tmp$ stat test.txt
  File: `test.txt'
  Size: 4               Blocks: 8          IO Block: 4096   regular file
Device: 301h/769d       Inode: 419875      Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2015-03-25 15:25:39.000000000 +0800
Modify: 2015-03-25 15:25:19.000000000 +0800
Change: 2015-03-25 15:25:19.000000000 +0800
:/home/chenkai3/tmp$ ln test.txt test_ln.txt && ln -s test.txt test_s.txt
:/home/chenkai3/tmp$ stat test.txt
  File: `test.txt'
  Size: 4               Blocks: 8          IO Block: 4096   regular file
Device: 301h/769d       Inode: 419875      Links: 2
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2015-03-25 15:25:39.000000000 +0800
Modify: 2015-03-25 15:25:19.000000000 +0800
Change: 2015-03-25 15:26:49.000000000 +0800
:/home/chenkai3/tmp$ ll -ai
total 16
1527658 drwxr-xr-x 2 root    root    4096 Mar 25 15:26 .
1527545 drwx------ 9 jianxin jianxin 4096 Mar 25 14:58 ..
 419875 -rw-r--r-- 2 root    root       4 Mar 25 15:25 test_ln.txt
 419876 lrwxrwxrwx 1 root    root       8 Mar 25 15:26 test_s.txt -> test.txt
 419875 -rw-r--r-- 2 root    root       4 Mar 25 15:25 test.txt
```

shell子进程

子进程

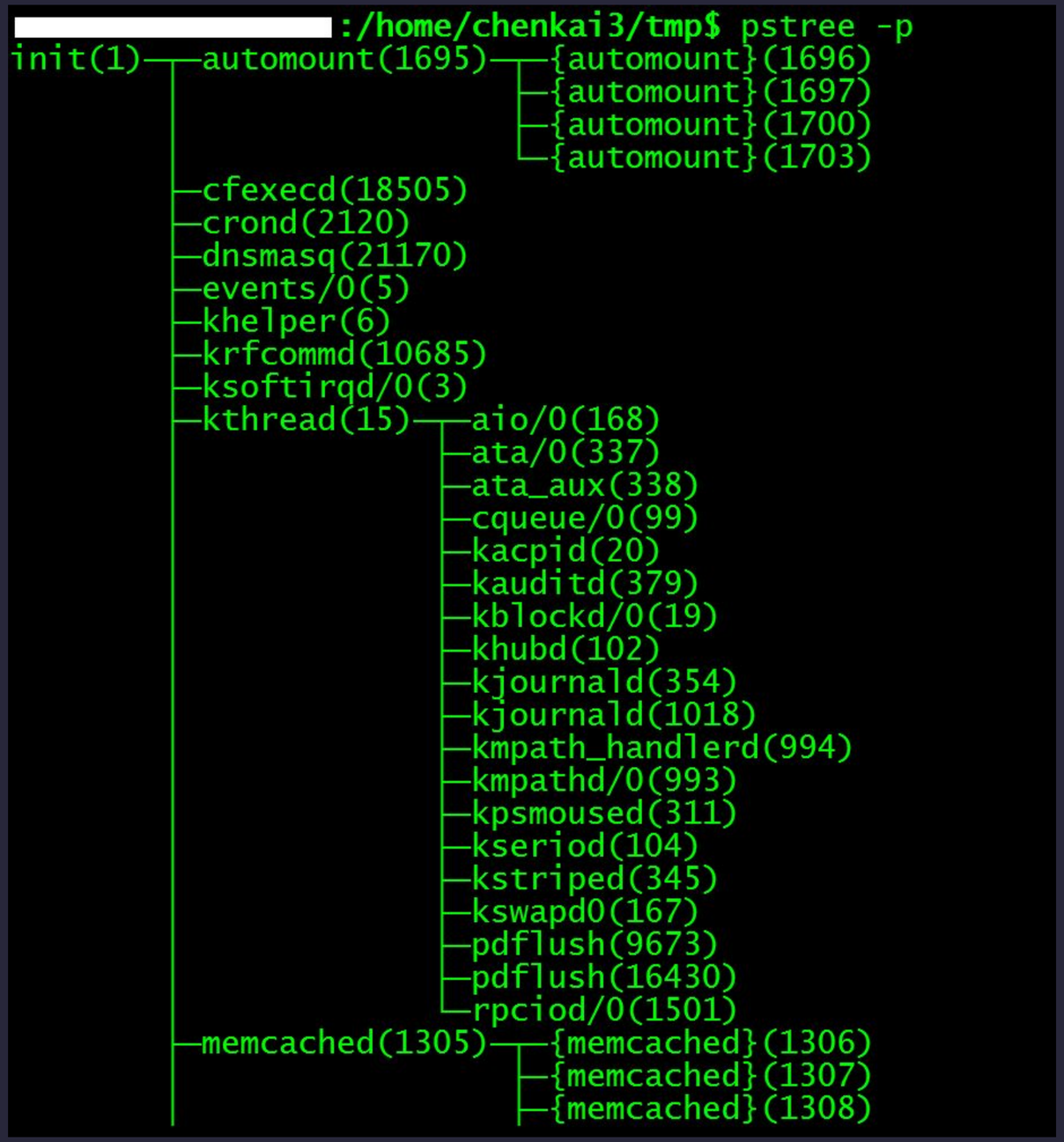
- **概念：**
  init进程（pid=1）以后创建的所有进程均称为子进程
  孤儿进程的ppid=1

- 创建：
  fork()创建子进程环境
  exec()加载、执行进程代码

子进程 - 产生场景

- 提交后台作业 &
- 管道 |
- 括号命令列表（）
- 执行外部脚本

进程树


[redacted]:/home/chenkai3/tmp$ pstree -p
init(1)─┬─automount(1695)─┬─{automount}(1696)
├─{automount}(1697)
├─{automount}(1700)
└─{automount}(1703)
├─cfexecd(18505)
├─crond(2120)
├─dnsmasq(21170)
├─events/0(5)
├─khelper(6)
├─krfcommd(10685)
├─ksoftirqd/0(3)
├─kthread(15)─┬─aio/0(168)
├─ata/0(337)
├─ata_aux(338)
├─cqueue/0(99)
├─kacpid(20)
├─kauditd(379)
├─kblockd/0(19)
├─khubd(102)
├─kjournald(354)
├─kjournald(1018)
├─kmpath_handlerd(994)
├─kmpathd/0(993)
├─kpsmoused(311)
├─kseriod(104)
├─kstriped(345)
├─kswapd0(167)
├─pdflush(9673)
├─pdflush(16430)
└─rpciod/0(1501)
├─memcached(1305)─┬─{memcached}(1306)
├─{memcached}(1307)
├─{memcached}(1308)

管道是文件

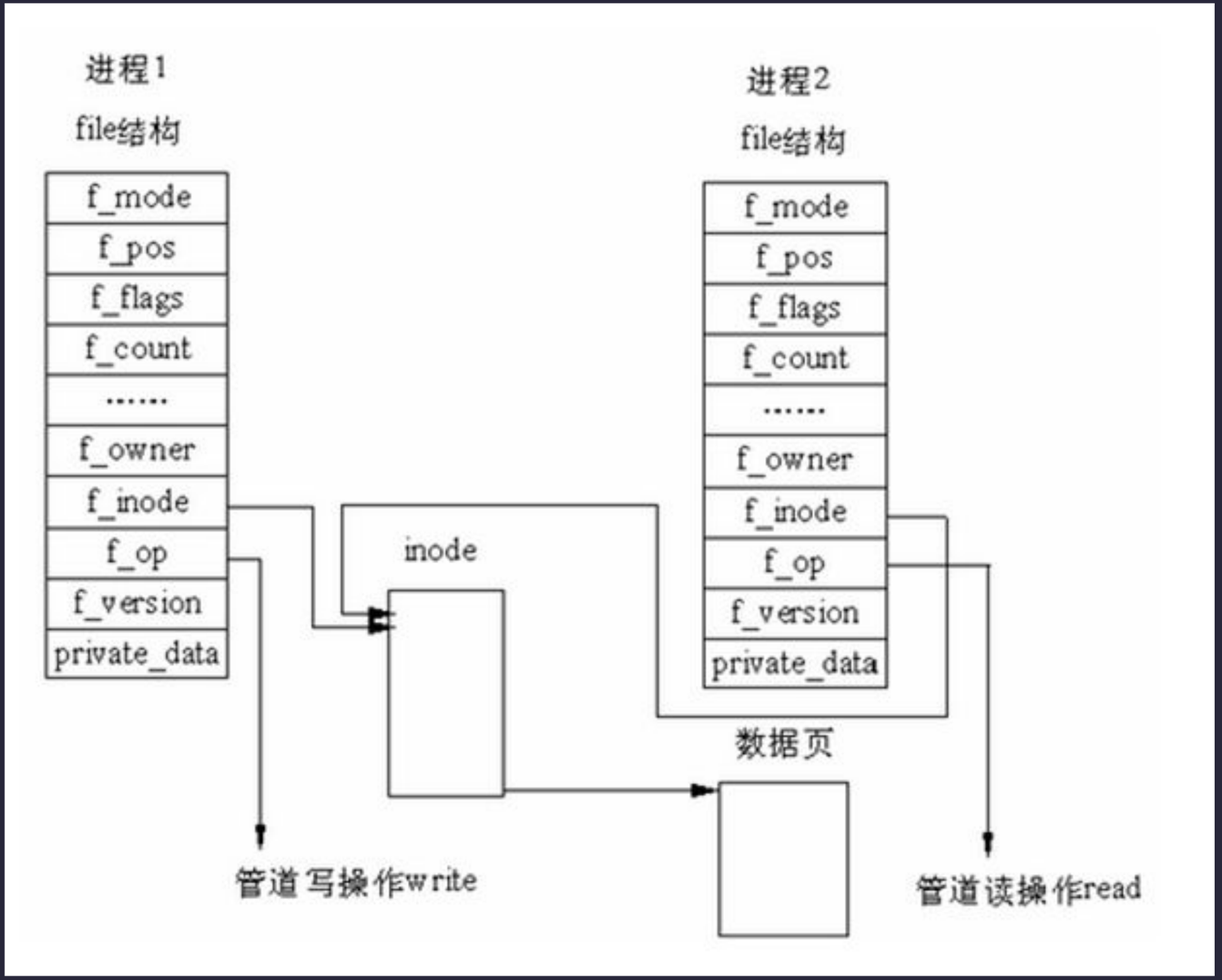
管道的特点

- 使用最多的进程间通信方式
  （其他通信机制：共享内存、消息队列、信号量、套接字等）

- 仅存在于内存，大小固定
  （一般为4K，即一个内存页的大小）

- 半双工，原子读、非原子写
  （管道内数据被读取后管道随即清空，此时读操作会被阻塞）

管道的结构


进程1
file结构
f_mode
f_pos
f_flags
f_count
.....
f_owner
f_inode
f_op
f_version
private_data
进程2
file结构
f_mode
f_pos
f_flags
f_count
.....
f_owner
f_inode
f_op
f_version
private_data
inode
数据页
管道写操作write
管道读操作read

管道的结构

- 两个file结构指向同一个VFS索引节点（inode），索引节点指向一个内存页

- 读入端（fd[0]）、写出端（fd[1]）

管道的读写

- 向管道写：将字节流数据复制到inode指向的物理内存

  conditions：

  left memery>=data size && memery unlock

  Yes：

  lock memery --> copy data --> unlock memery

  No：

  write_process sleep in inode's waiting queue until all above conditions are met

管道的读写

- 从管道读：一次性复制物理内存中的全部字节流数据

- 读进程可在管道无数据或内存被锁定时立即返回（依赖文件或管道的打开模式），或休眠在inode的等待队列，直到内存被解锁

结论

- 管道操作符“|”，将I/O流重定向到一个子进程中，子进程中的变量对于子进程之外的代码块不可见

- 父进程不能访问这些变量，也不能通过export方式将子进程变量导出到父进程

Q&A

THE END