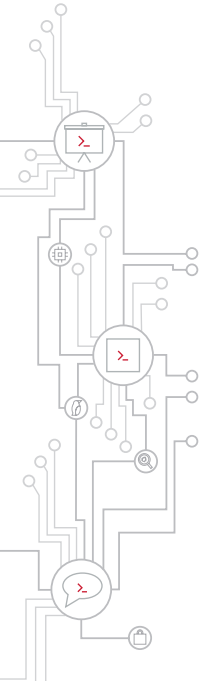

Hidden Gems in Linux's /proc File System

Richard Weinberger - sigma star gmbh

2024-01-18

Hello

Richard Weinberger

- Co-founder of sigma star gmbh
- Linux kernel developer and maintainer
- Strong focus on Linux kernel, lowlevel components, virtualization, security, code audits

sigma star gmbh

- Software Development & Security Consulting
- Main areas: Embedded Systems, Linux Kernel & Security
- Contributions to Linux Kernel and other OSS projects

procfs: Process File System

- Original goal: Expose process table as virtual files
- Usually mounted to `/proc`
- Every numeric top level directory represents a process
- Backbone for tools such as `ps(1)`, `top(1)` or `iotop(1)`
- But there is more

Kernel Stack Inspection

- Assume process 537 makes no progress and blocks, what now?
- Let's figure where it blocks inside the kernel
- procfs shows kernel stacks for processes (actually threads)
- `/proc/<pid>/stack` or `/proc/<pid>/task/<tid>/stack`

```
$ cat /proc/537/stack
[<0>] folio_wait_bit_common+0x13d/0x350
[<0>] filemap_get_pages+0x60a/0x630
[<0>] filemap_read+0xd2/0x340
[<0>] cifs_strict_readv+0x150/0x170 [cifs]
[<0>] vfs_read+0x239/0x310
[<0>] ksys_read+0x6b/0xf0
[<0>] do_syscall_64+0x58/0xc0
[<0>] entry_SYSCALL_64_after_hwframe+0x64/0
    xce
```

File Table Inspection

- Open files are modeled as symbolic links
- Symlink target is either the opened file or target inode

```
$ ls -l /proc/self/fd
total 0
lrwx------ 1 root root 64 May 31 15:49 0 -> /dev/pts/10
lrwx------ 1 root root 64 May 31 15:49 1 -> /dev/pts/10
lrwx------ 1 root root 64 May 31 15:49 2 -> /dev/pts/10
lr-x------ 1 root root 64 May 31 15:49 3 -> /proc/1387/fd

$ ls -l /proc/`pidof qemu-system-x86_64`/fd | grep disk
lrwx------ 1 rw users 64 May 31 15:49 18 -> /home/rw/linux/disk1.raw
lrwx------ 1 rw users 64 May 31 15:49 3 -> /home/rw/linux/disk2.raw
```

File Table Inspection: Find Connected FDs

- Example: `cat | less`

```
$ ls -l /proc/`pidof less`/fd
total 0
lr-x------ 1 rw users 64 May 31 15:57 0 -> pipe:[3227038]
lrwx------ 1 rw users 64 May 31 15:57 1 -> /dev/pts/12
lrwx------ 1 rw users 64 May 31 15:57 2 -> /dev/pts/12
lr-x------ 1 rw users 64 May 31 15:57 3 -> /dev/tty

$ ls -l /proc/`pidof cat`/fd
total 0
lrwx------ 1 rw users 64 May 31 15:58 0 -> /dev/pts/12
l-wx------ 1 rw users 64 May 31 15:58 1 -> pipe:[3227038]
lrwx------ 1 rw users 64 May 31 15:58 2 -> /dev/pts/12
```

- Inodes are from `pipefs`
- Inode numbers match, we know that stdin of `less` and stdout of `cat` are connected!
- Same works for sockets, namespaces, etc.

File Table Inspection: Recover Unlinked Files

- Let's do something stupid: `rm /home/rw/linux/disk1.raw`
- As long a process has a file handle to it, we can recover the file!

```
$ ls /home/rw/linux/disk1.raw
ls: cannot access '/home/rw/linux/disk1.raw': No such file or directory

$ ls -l /proc/1257/fd | grep disk
lrwx------ 1 rw users 64 May 31 14:41 18 -> /home/rw/linux/disk1.raw (deleted)
lrwx------ 1 rw users 64 May 31 14:41 3 -> /home/rw/linux/disk2.raw

$ cat /proc/1257/fd/18 > /home/rw/linux/disk1.raw
```

- Open of `/proc/1257/fd/18` does *not* open the symlink target
- procfs installs the *existing* file handle into your process!

File Table Inspection: File Status

- Beside of open flags and exact location, `/proc/<pid>/fdinfo/<fd>` tells us the current offset
- Poor man's progress counter!

```
$ sha256sum massive_file.raw &
[1] 7475

$ ls -l /proc/7475/fd | grep massive_file.raw
lr-x------ 1 rw users 64 May 31 16:55 3 -> /data1/massive_file.raw

$ grep pos: /proc/7475/fdinfo/3
pos:    5483397

...

$ grep pos: /proc/7475/fdinfo/3
pos:    1457750016
```

FIN

Thank you!

Questions, Comments?

Richard Weinberger
richard@sigma-star.at