# [简析Linux中 /proc/[pid] 目录的各文件](https://www.linuxprobe.com/linux-proc-pid.html)

[Linux](https://www.linuxprobe.com/) 内核提供了一种通过 proc 文件系统，在运行时访问内核内部数据结构、改变内核设置的机制。proc 文件系统是一个伪文件系统，它只存在内存当中，而不占用外存空间。它以文件系统的方式为访问系统内核数据的操作提供接口。

用户和应用程序可以通过 proc 得到系统的信息，并可以改变内核的某些参数。由于系统的信息，如进程，是动态改变的，所以用户或应用程序读取 proc 文件时，proc 文件系统是动态从系统内核读出所需信息并提交的。

下面列出的这些文件或子文件夹，并不是都是在你的系统中存在，这取决于你的内核配置和装载的模块。另外，在 proc 下还有三个很重要的目录：net，scsi 和 sys。 sys 目录是可写的，可以通过它来访问或修改内核的参数，而 net 和 scsi 则依赖于内核配置。例如，如果系统不支持 scsi，则 scsi 目录不存在。

除了以上介绍的这些，还有的是一些以数字命名的目录，它们是进程目录。系统中当前运行的每一个进程都有对应的一个目录在 proc 下，以进程的 PID 号为目录名，它们是读取进程信息的接口。而 self 目录则是读取进程本身的信息接口，是一个 link。

/proc/[pid]/auxv

/proc/[pid]/auxv 包含传递给进程的 ELF 解释器信息，格式是每一项都是一个 unsigned long长度的 ID 加上一个 unsigned long 长度的值。最后一项以连续的两个 0x00 开头。举例如下：

$ hexdump -x /proc/2406/auxv

0000000 0021 0000 0000 0000 f000 f7fa 7fff 0000

0000010 0010 0000 0000 0000 fbff 1f8b 0000 0000

0000020 0006 0000 0000 0000 1000 0000 0000 0000

0000030 0011 0000 0000 0000 0064 0000 0000 0000

0000040 0003 0000 0000 0000 0040 0040 0000 0000

0000050 0004 0000 0000 0000 0038 0000 0000 0000

0000060 0005 0000 0000 0000 0007 0000 0000 0000

0000070 0007 0000 0000 0000 0000 0000 0000 0000

0000080 0008 0000 0000 0000 0000 0000 0000 0000

0000090 0009 0000 0000 0000 55e0 0045 0000 0000

00000a0 000b 0000 0000 0000 0000 0000 0000 0000

00000b0 000c 0000 0000 0000 0000 0000 0000 0000

00000c0 000d 0000 0000 0000 0000 0000 0000 0000

00000d0 000e 0000 0000 0000 0000 0000 0000 0000

00000e0 0017 0000 0000 0000 0000 0000 0000 0000

00000f0 0019 0000 0000 0000 f079 f7f6 7fff 0000

0000100 001f 0000 0000 0000 ffea f7f6 7fff 0000

0000110 000f 0000 0000 0000 f089 f7f6 7fff 0000

0000120 0000 0000 0000 0000 0000 0000 0000 0000

0000130

/proc/[pid]/cmdline

/proc/[pid]/cmdline 是一个只读文件，包含进程的完整[命令](https://www.linuxcool.com/)行信息。如果该进程已经被交换出内存或者这个进程是 zombie 进程，则这个文件没有任何内容。该文件以空字符 null 而不是换行符作为结束标志。举例如下：

$ ps aux|grep frps

root 2406 0.1 0.1 54880 10524 ? Sl Dec11 21:30 frps -c ./frps.ini

$ cat /proc/2406/cmdline

frps-c./frps.ini

/proc/[pid]/comm

/proc/[pid]/comm 包含进程的[命令](https://www.linuxcool.com/)名。举例如下：

$ cat /proc/2406/comm

frps

/proc/[pid]/cwd

/proc/[pid]/cwd 是进程当前工作目录的符号链接。举例如下：

$ ls -lt /proc/2406/cwd

lrwxrwxrwx 1 root root 0 Dec 12 20:39 /proc/2406/cwd -> /home/mike/frp\_0.13.0\_linux\_amd64

/proc/[pid]/environ

/proc/[pid]/environ 显示进程的环境变量。举例如下：

$ strings /proc/2406/environ

SUPERVISOR\_GROUP\_NAME=ssh

TERM=linux

SUPERVISOR\_SERVER\_URL=unix:///var/run/supervisor.sock

SUPERVISOR\_PROCESS\_NAME=ssh

RUNLEVEL=2

UPSTART\_EVENTS=runlevel

PREVLEVEL=N

PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

UPSTART\_INSTANCE=

UPSTART\_JOB=rc

SUPERVISOR\_ENABLED=1

runlevel=2

PWD=/

previous=N

/proc/[pid]/exe

/proc/[pid]/exe 为实际运行程序的符号链接。举例如下：

$ ls -lt /proc/2406/exe

lrwxrwxrwx 1 root root 0 Dec 11 19:00 /proc/2406/exe -> /usr/bin/frps

/proc/[pid]/fd

/proc/[pid]/fd 是一个目录，包含进程打开文件的情况。举例如下：

$ ls -lt /proc/2406/fd

lrwx------ 1 root root 64 Dec 24 09:39 77 -> socket:[44377722]

lrwx------ 1 root root 64 Dec 17 15:07 47 -> socket:[29482617]

lr-x------ 1 root root 64 Dec 12 20:18 0 -> pipe:[13282]

l-wx------ 1 root root 64 Dec 12 20:18 1 -> pipe:[13283]

lrwx------ 1 root root 64 Dec 12 20:18 10 -> socket:[12238218]

lrwx------ 1 root root 64 Dec 12 20:18 4 -> anon\_inode:[eventpoll]

lrwx------ 1 root root 64 Dec 12 20:18 40 -> socket:[19378614]

目录中的每一项都是一个符号链接，指向打开的文件，数字则代表文件描述符。

/proc/[pid]/latency

/proc/[pid]/latency 显示哪些代码造成的延时比较大。如果要使用这个特性需要执行：

$ echo 1 > /proc/sys/kernel/latencytop

举例如下

$ cat /proc/2406/latency

Latency Top version : v0.1

30667 10650491 4891 poll\_schedule\_timeout do\_sys\_poll SyS\_poll system\_call\_fastpath 0x7f636573dc1d

8 105 44 futex\_wait\_queue\_me futex\_wait do\_futex SyS\_futex system\_call\_fastpath 0x7f6365a167bc

每一行前三个数字分别是后面代码执行的次数、总共执行延迟时间(单位是微秒)和最长执行延迟时间(单位是微秒)。后面则是代码完整的调用栈。

/proc/[pid]/maps

/proc/[pid]/maps 显示进程的内存区域映射信息。举例如下：

$ cat /proc/2406/maps

00400000-006ea000 r-xp 00000000 fd:01 1727569 /usr/bin/frps

006ea000-00a6c000 r--p 002ea000 fd:01 1727569 /usr/bin/frps

00a6c000-00ab1000 rw-p 0066c000 fd:01 1727569 /usr/bin/frps

00ab1000-00ad4000 rw-p 00000000 00:00 0

c000000000-c00000b000 rw-p 00000000 00:00 0

c41feac000-c420000000 rw-p 00000000 00:00 0

c420000000-c420400000 rw-p 00000000 00:00 0

c420400000-c420700000 rw-p 00000000 00:00 0

c420700000-c420800000 rw-p 00000000 00:00 0

c420800000-c420900000 rw-p 00000000 00:00 0

c420900000-c420a00000 rw-p 00000000 00:00 0

c420a00000-c421ea0000 rw-p 00000000 00:00 0

c421ea0000-c422a00000 rw-p 00000000 00:00 0

c422a00000-c422a60000 rw-p 00000000 00:00 0

7f0418c01000-7f0418ee1000 rw-p 00000000 00:00 0

7ffff7f4f000-7ffff7f70000 rw-p 00000000 00:00 0 [stack:5121]

7ffff7fad000-7ffff7faf000 r--p 00000000 00:00 0 [vvar]

7ffff7faf000-7ffff7fb1000 r-xp 00000000 00:00 0 [vdso]

ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0 [vsyscall]

其中注意的一点是 [stack:] 是线程的堆栈信息，对应于 /proc/[pid]/task/[tid]/ 路径。

/proc/[pid]/root

/proc/[pid]/root 是进程根目录的符号链接。举例如下：

$ ls -lt /proc/2406/root

lrwxrwxrwx 1 root root 0 Dec 12 20:39 /proc/2406/root -> /

/proc/[pid]/stack

/proc/[pid]/stack 示当前进程的内核调用栈信息，只有内核编译时打开了 CONFIG\_STACKTRACE 编译选项，才会生成这个文件。举例如下：

$ cat /proc/2406/stack

[] futex\_wait\_queue\_me+0xc6/0x130

[] futex\_wait+0x17d/0x270

[] do\_futex+0xd5/0x520

[] SyS\_futex+0x71/0x150

[] entry\_SYSCALL\_64\_fastpath+0x16/0x75

[] 0xffffffffffffffff

/proc/[pid]/statm

/proc/[pid]/statm 显示进程所占用内存大小的统计信息。包含七个值，度量单位是 page(page大小可通过 getconf PAGESIZE 得到)。举例如下：

$ cat /proc/2406/statm

13720 2617 493 746 0 12007 0

各个值含义：

a）进程占用的总的内存

b）进程当前时刻占用的物理内存

c）同其它进程共享的内存

d）进程的代码段

e）共享库(从2.6版本起，这个值为0)

f）进程的堆栈

g）dirty pages(从2.6版本起，这个值为0)

/proc/[pid]/status

/proc/[pid]/status 包含进程的状态信息。其很多内容与 /proc/[pid]/stat 和 /proc/[pid]/statm 相同，但是却是以一种更清晰地方式展现出来。举例如下：

$ cat /proc/2406/status

Name: frps

State: S (sleeping)

Tgid: 2406

Ngid: 0

Pid: 2406

PPid: 2130

TracerPid: 0

Uid: 0 0 0 0

Gid: 0 0 0 0

FDSize: 128

Groups: 0

NStgid: 2406

NSpid: 2406

NSpgid: 2406

NSsid: 2130

VmPeak: 54880 kB

VmSize: 54880 kB

VmLck: 0 kB

VmPin: 0 kB

VmHWM: 34872 kB

VmRSS: 10468 kB

VmData: 47896 kB

VmStk: 132 kB

VmExe: 2984 kB

VmLib: 0 kB

VmPTE: 68 kB

VmPMD: 20 kB

VmSwap: 0 kB

HugetlbPages: 0 kB

Threads: 11

SigQ: 0/31834

SigPnd: 0000000000000000

ShdPnd: 0000000000000000

SigBlk: 0000000000000000

SigIgn: 0000000000000000

SigCgt: fffffffe7fc1feff

CapInh: 0000000000000000

CapPrm: 0000003fffffffff

CapEff: 0000003fffffffff

CapBnd: 0000003fffffffff

CapAmb: 0000000000000000

Seccomp: 0

Cpus\_allowed: f

Cpus\_allowed\_list: 0-3

Mems\_allowed: 00000000,00000001

Mems\_allowed\_list: 0

voluntary\_ctxt\_switches: 2251028

nonvoluntary\_ctxt\_switches: 18031

关于信号（signal）的信息：SigQ 分为两部分（例如 0/31834），前面表示当前处在队列中的信号（0），后面则表示队列一共可以存储多少信号（31834）；SigPnd 表示当前线程 pending 的信号，而ShdPnd 则表示整个进程 pending 的信号；SigBlk、SigIgn 和 SigCgt 分别表示对信号的处理是阻塞，忽略，还是捕获。（关于Unix信号的相关知识，可以参考 Unix: Dealing with signals）。

/proc/[pid]/syscall

/proc/[pid]/syscall 显示当前进程正在执行的系统调用。举例如下：

$ cat /proc/2406/syscall

202 0xab3730 0x0 0x0 0x0 0x0 0x0 0x7ffff7f6ec68 0x455bb3

第一个值是系统调用号（202代表poll），后面跟着 6 个系统调用的参数值（位于寄存器中），最后两个值依次是堆栈指针和指令计数器的值。如果当前进程虽然阻塞，但阻塞函数并不是系统调用，则系统调用号的值为 -1，后面只有堆栈指针和指令计数器的值。如果进程没有阻塞，则这个文件只有一个 running 的字符串。

内核编译时打开了 CONFIG\_HAVE\_ARCH\_TRACEHOOK 编译选项，才会生成这个文件。

/proc/[pid]/wchan

/proc/[pid]/wchan 显示当进程 sleep 时，kernel 当前运行的函数。举例如下：

$ cat /proc/2406/wchan

futex\_wait\_queue\_meadmin