# 概述

strace命令是一个集诊断、调试、统计与一体的工具，我们可以利用strace来跟踪debug应用程序的系统调用和信号传递过程，以便我们对应用运行机制进行分析，从而解决应用程序遇到的问题。

# 使用

strace的使用相当简单，就是执行一个指定的命令比如

strace -T -tt -s 100 -o /tmp/strace.log CMD

在指定的CMD命令结束之后它也就退出。在命令执行的过程中，strace会记录和解析命令进程的所有系统调用以及这个进程所接收到的所有的信号值。

或者

strace -T -tt -s 100 -o /tmp/strace.log -p PID

对于指定进程号的跟踪，需要Crtl +C 结束调用。

strace参数

-c 统计每一系统调用的所执行的时间,次数和出错的次数等.

-d 输出strace关于标准错误的调试信息.

-f 跟踪由fork调用所产生的子进程.

-ff 如果提供-o filename,则所有进程的跟踪结果输出到相应的filename.pid中,pid是各进程的进程号.

-F 尝试跟踪vfork调用.在-f时,vfork不被跟踪.

-h 输出简要的帮助信息.

-i 输出系统调用的入口指针.

-q 禁止输出关于脱离的消息.

-r 打印出相对时间关于,,每一个系统调用.

-t 在输出中的每一行前加上时间信息.

-tt 在输出中的每一行前加上时间信息,微秒级.

-ttt 微秒级输出,以秒了表示时间.

-T 显示每一调用所耗的时间. 这个参数在排查性能问题时特别有用。

-v 输出所有的系统调用.一些调用关于环境变量,状态,输入输出等调用由于使用频繁,默认不输出.

-a column 设置返回值的输出位置.默认 为40.

-e expr 指定一个表达式,用来控制如何跟踪.格式如下:

[qualifier=][!]value1[,value2]...

qualifier只能是 trace,abbrev,verbose,raw,signal,read,write其中之一.value是用来限定的符号或数字.默认的 qualifier是 trace.感叹号是否定符号.例如: -eopen等价于 -e trace=open,表示只跟踪open调用.而-etrace!=open表示跟踪除了open以外的其他调用.有两个特殊的符号 all 和 none. 注意有些shell使用!来执行历史记录里的命令,所以要使用\.

-e trace=set 只跟踪指定的系统 调用.例如:-e trace=open,close,rean,write表示只跟踪这四个系统调用.默认的为set=all.

-e trace=file 只跟踪有关文件操作的系统调用.

-e trace=process 只跟踪有关进程控制的系统调用.

-e trace=network 跟踪与网络有关的所有系统调用.

-e strace=signal 跟踪所有与系统信号有关的 系统调用

-e trace=ipc 跟踪所有与进程通讯有关的系统调用

-e abbrev=set 设定 strace输出的系统调用的结果集.-v 等与 abbrev=none.默认为abbrev=all.

-e raw=set 将指 定的系统调用的参数以十六进制显示.

-e signal=set 指定跟踪的系统信号.默认为all.如 signal=!SIGIO(或者signal=!io),表示不跟踪SIGIO信号.

-e read=set 输出从指定文件中读出的数据.例如: -e read=3,5

-e write=set 输出写入到指定文件中的数据.

-o filename 将strace的输出写入文件filename.

-p pid 跟踪指定的进程pid.

-s strsize 指定输出的字符串的最大长度.默认为32.文件名一直全部输出.

-u username 以username 的UID和GID执行被跟踪的命令

上面解释strace的参数，其实常用的参数有-T,-tt,-c ,默认-e 为trace=all,有兴趣的朋友可以多试试 -e 参数来查看不同的参数下日志输出。

# 举例

我们可以尝试使用strace 定位 my 命令使用到的文件。

[root@qabb-r1db13 09:50:26 ~]

# strace -ttT -o /tmp/my\_3316.log -s 64 -e trace=file my 3316

(none) [RW][TEST:qa\_mytest:3316] 09:51:00 >exit

Bye

查看文件 /tmp/my\_3316.log 的内容，可以一目了然看到my命令访问(open,stat)的文件。

09:51:00.143006 execve("/opt/isys/bin/my", ["my", "3316"], [/\* 29 vars \*/]) = 0 <0.000211>

09:51:00.143454 access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory) <0.000008>

09:51:00.143509 open("/etc/ld.so.cache", O\_RDONLY) = 3 <0.000007>

09:51:00.143582 open("/lib64/libtinfo.so.5", O\_RDONLY) = 3 <0.000010>

09:51:00.143708 open("/lib64/libdl.so.2", O\_RDONLY) = 3 <0.000010>

09:51:00.143817 open("/lib64/libc.so.6", O\_RDONLY) = 3 <0.000008>

09:51:00.144196 open("/dev/tty", O\_RDWR|O\_NONBLOCK) = 3 <0.000013>

09:51:00.144357 open("/usr/lib/locale/locale-archive", O\_RDONLY) = 3 <0.000010>

09:51:00.144566 open("/proc/meminfo", O\_RDONLY|O\_CLOEXEC) = 3 <0.000015>

09:51:00.144967 stat("/root", {st\_mode=S\_IFDIR|0550, st\_size=4096, ...}) = 0 <0.000008>

09:51:00.145016 stat(".", {st\_mode=S\_IFDIR|0550, st\_size=4096, ...}) = 0 <0.000006>

09:51:00.145076 open("/usr/lib64/gconv/gconv-modules.cache", O\_RDONLY) = 3 <0.000010>

09:51:00.145501 open("/opt/isys/bin/my", O\_RDONLY) = 3 <0.000010>

09:51:00.145944 stat("/srv/my\_3316/mysqld.sock", 0x7ffc5cc12050) = -1 ENOENT (No such file or directory) <0.000008>

09:51:00.146005 stat("/srv/my3316/run/mysql.sock", {st\_mode=S\_IFSOCK|0777, st\_size=0, ...}) = 0 <0.000009>

09:51:00.147147 --- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=23258, si\_status=0, si\_utime=0, si\_stime=0} ---

09:51:00.149123 --- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=23259, si\_status=0, si\_utime=0, si\_stime=0} ---

09:51:00.151138 --- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=23263, si\_status=0, si\_utime=0, si\_stime=0} ---

09:51:00.151437 stat("/opt/mysql/bin/mysql", {st\_mode=S\_IFREG|0755, st\_size=4235424, ...}) = 0 <0.000031>

09:51:00.156526 --- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=23267, si\_status=0, si\_utime=0, si\_stime=0} ---

09:51:01.748378 --- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=23271, si\_status=0, si\_utime=0, si\_stime=0} ---

09:51:01.748665 +++ exited with 0 +++

# 小结

当应用程序进程出现异常(比如性能间歇性变慢)，我们可以使用strace来跟踪 debug其系统调用，检查程序运行时哪一步耗时比较长，进而找到系统瓶颈。

当然仅仅知道strace工具还是不够的，配合gdb/perf/SystemTap/pstack 等系统工具对应用程序进行“望闻问切”,才能得到我们要的结果。