

进程追踪

手稿

封面

进程追踪是调试器的核心组成部分，在 Linux 下可用系统调用 ptrace 实现。

概述

首先，需要了解以下两个概念：

1. 追踪者 -- 观察和控制其他进程的进程；
2. 被追踪者 -- 被调试程序所在进程；

系统调用 `ptrace` 提供了让追踪者能够观察和控制被追踪者的一种方式，并可以检查并修改被追踪者的内存和寄存器，主要用于实现断点调试和系统调用的追踪。

追踪对象

若要追踪进程，被追踪者必须首先与追踪者相关联。

关联以及后续命令都是针对线程的：在多线程（用 clone、CLONE_THREAD 标志创建的线程）进程中，每个线程可以与不同的追踪者单独关联，或者不关联（不调试）。因此，被追踪者总是意味着“线程”，绝不是“进程”（但可能是多线程）。

ptrace 使用下面的方式向特定被追踪者发送命令：

```
ptrace(PTRACE_foo, pid, ...)
```

这里的 pid 指的是相应 Linux 线程的线程 ID。

工作原理

如果进程要发起追踪，可以调用 `fork`，然后在子进程中以 `PTRACE_TRACEME` 调用 `ptrace`，最后（通常是）调用 `execve`。或者，进程以 `PTRACE_ATTACH` 或 `PTRACE_SEIZE` 调用 `ptrace` 来追踪其他进程。

当进程被追踪时，被追踪者在收到信号后总是暂停执行，即使信号被忽略。（`SIGKILL` 除外，它仍旧按原方式工作，即终止进程。）

追踪者在调用 `waitpid`（或其他 `wait` 系列函数）后将得到通知，在 `status` 中将包含导致这次暂停的信息。当被追踪者暂停后，追踪者可使用各种 `ptrace` 请求检查并修改被追踪者。追踪者可以让被追踪者继续，并可选择是否忽略收到的信号（甚至用其他信号代替这次的信号）。

函数原型与返回值

ptrace 的函数原型如下：

```
long ptrace(enum __ptrace_request request, pid_t pid,  
             void *addr, void *data);
```

如前所述，pid 指定目标对象的 ID，request 决定着要采取的动作（PTRACE_PEEK* 请求用于获取数据），addr 和 data 用于在追踪者和被追踪者之间传递数据。

当调用成功时，PTRACE_PEEK* 请求返回要求的数据，其他请求返回 0。当发生错误时，所有请求返回 -1，并设置 errno。

注意：由于成功的 PTRACE_PEEK* 请求也可能会返回 -1（获取到的数据），因此调用者必须先将 errno 清零，事后再检查它的值来确定是否发生了错误。

追踪的开始与结束

所有被追踪者成功的 `execve` 调用都将导致它被发送一个 `SIGTRAP` 信号（如果没有指定 `PTRACE_O_TRACEEXEC` 选项的话），让父进程有机会在新程序开始运行之前就获得对它的控制权。

当追踪者结束追踪后，它可以让被追踪者正常运行，即通过 `PTRACE_DETACH` 请求使其退出追踪模式。

追踪请求

下面是 ptrace 可以使用的追踪请求列表：

追踪请求	功能	备注
PTRACE_TRACEME	表示该进程将被它的父进程追踪。	pid、addr、data 被忽略； 只用于被追踪者，其余均用于追踪者。
PTRACE_PEEKTEXT, PTRACE_PEEKDATA	从被追踪者的内存地址 addr 处读取一个字，作为 ptrace 调用的返回值。	Linux 没有独立的 text 和 data 地址空间，因此这两个请求是一样的。
PTRACE_PEEKUSER	从被追踪者的USER区偏移地址 addr 处读取一个字，其中包含寄存器和有关进程的其他信息。这个字作为 ptrace 调用的返回值。	通常偏移地址需要字对齐，每个架构可能不同。
PTRACE_POKETEXT, PTRACE_POKEDATA	将data拷贝至被追踪者内存地址 addr 处。	如同 PTRACE_PEEKTEXT 和PTRACE_PEEKDATA， 它们是一样的。
PTRACE_POKEUSER	将字 data 拷贝至被追踪者 USER 区偏移地址 addr 处。	如同 PTRACE_PEEKUSER，偏移地址需字对齐。 为了维护内核完整性，有些修改是不允许的。
PTRACE_GETREGS, PTRACE_GETFPREGS	将被追踪者的通用寄存器或浮点寄存器拷贝至追踪者相应地址 data 处。	SPARC 系统以相反方式使用 data 和 addr，即忽略 data，拷贝至 addr。 并非所有架构都支持这两个请求。
PTRACE_GETREGSET	读取被追踪者的寄存器。addr 指定寄存器类型（架构相关）。 data 指向 struct iovec，描述目标缓冲区的位置和长度。	

追踪请求（续 1）

ptrace 可以使用的追踪请求列表（续 1）：

追踪请求	功能	备注
PTRACE_SETREGS, PTRACE_SETFPREGS	根据追踪者地址 data 修改被追踪者的通用寄存器或浮点寄存器。	如同 PTRACE_POKEUSER，有些修改不允许。 SPARC 使用 addr 而不是 data。 并非所有架构都支持这两个请求。
PTRACE_SETREGSET	修改被追踪者的寄存器。addr 和 data 的含义与 PTRACE_GETREGSET 类似	自 Linux 2.6.34
PTRACE_GETSIGINFO	获取导致本次暂停的有关信息。从被追踪者拷贝 siginfo_t 结构至追踪者地址 data 处。（忽略 addr）	自 Linux 2.3.99-pre6
PTRACE_SETSIGINFO	从追踪者地址 data 处拷贝 siginfo_t 结构至被追踪者。	自 Linux 2.3.99-pre6
PTRACE_PEEKSIGINFO	获取 siginfo_t 结构，但不从队列中删除信号。addr 指定拷贝信号的起始位置和拷贝数量。data 指定缓冲区。	自 Linux 3.10
PTRACE_GETSIGMASK	在指向 siginfo_t 类型缓冲区的指针 data 中放置阻塞信号掩码的拷贝，addr 是缓冲区的大小。	自 Linux 3.11
PTRACE_SETSIGMASK	用指向 siginfo_t 类型缓冲区的指针 data 中的值修改阻塞信号掩码，addr 是缓冲区的大小。	自 Linux 3.11

追踪请求（续 2）

ptrace 可以使用的追踪请求列表（续 2）：

追踪请求	功能	备注
PTRACE_SETOPTIONS	用 data 设置 ptrace 选项，data 被解释为选项的位掩码。	自 Linux 2.4.6
PTRACE_GETEVENTMSG	获取刚刚发生的 ptrace 事件有关消息（unsigned long），放置在追踪者地址 data 处。	自 Linux 2.5.46
PTRACE_CONT	恢复已暂停的被追踪进程。data 如果不为 0，则表示要传递给它的信号数量。（忽略 addr）	
PTRACE_SYSCALL, PTRACE_SINGLESTEP	恢复已暂停的被追踪进程，但是相应地，在下一个系统调用的入口或出口、单步执行后暂停。data 和 addr 同 PTRACE_CONT。	
PTRACE_SYSEMU, PTRACE_SYSEMU_SINGLESTEP	对于 PTRACE_SYSEMU，继续并在下一个系统调用入口处暂停，但不执行。PTRACE_SYSEMU_SINGLESTEP 一样，但如果不是系统调用依旧单步执行。data 和 addr 同 PTRACE_CONT。	自 Linux 2.6.14 仅 x86 支持
PTRACE_LISTEN	恢复已暂停的被追踪者，但是阻止它运行。效果如同被 SIGSTOP 暂停。	自 Linux 3.4 必须用 PTRACE_SEIZE 关联
PTRACE_KILL	向被追踪者发送 SIGKILL 去终止它。	已废弃 使用 kill 和 tkill

追踪请求（续 3）

ptrace 可以使用的追踪请求列表（续 3）：

追踪请求	功能	备注
PTRACE_INTERRUPT	暂停被追踪者。	自 Linux 3.4
PTRACE_ATTACH	与 pid 指定的进程关联，让它成为调用进程的追踪目标。被追踪者被发送 SIGSTOP，但在该调用完成之前，不一定会暂停，可以用 waitpid 等待它的暂停。	
PTRACE_SEIZE	与 pid 指定的进程关联，让它成为调用进程的追踪目标。与 PTRACE_ATTACH 不同的是，它不会暂停进程。addr 必须是 0，data 中是立即生效的 ptrace 选项位掩码。	自 Linux 3.4 execve 不会收到额外的 SIGTRAP
PTRACE_DETACH	就像 PTRACE_CONT，恢复已暂停的被追踪进程，但要先与之取消关联。（忽略 addr）	在 Linux 下，无论采用哪种方式发起追踪，均可用它来取消关联。

追踪选项

下面是 ptrace 可以使用的追踪选项列表：

追踪选项	含义	备注
PTRACE_O_EXITKILL	如果追踪者退出，向每个被追踪者发送 SIGKILL。	自 Linux 3.8
PTRACE_O_TRACECLONE	被追踪者在下一个 clone 处暂停，自动追踪新进程。	自 Linux 2.5.46
PTRACE_O_TRACEEXEC	被追踪者在下一个 execve 处暂停。	自 Linux 2.5.46
PTRACE_O_TRACEEXIT	当被追踪者退出时暂停。	自 Linux 2.5.60
PTRACE_O_TRACEFORK	被追踪者在下一个 fork 处暂停，自动追踪新进程。	自 Linux 2.5.46
PTRACE_O_TRACESYSGOOD	当传递系统调用陷阱时，设置信号值的 D7（即传递 SIGTRAP 0x80）。	自 Linux 2.4.6
PTRACE_O_TRACEVFORK	被追踪者在下一个 vfork 处暂停，自动追踪新进程。	自 Linux 2.5.46
PTRACE_O_TRACEVFORKDONE	被追踪者在下一个 vfork 完成后暂停，自动追踪新进程。	自 Linux 2.5.60
PTRACE_O_TRACESECCOMP	当 seccomp SECCOMP_RET_TRACE 被触发时，暂停被追踪者。	自 Linux 3.5
PTRACE_O_SUSPEND_SECCOMP	挂起被追踪者的 seccomp 保护。	自 Linux 4.3