|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **武汉大学国家网络安全学院课程实验报告** | | | | | | | |
| 课程名称 | 操作系统原理 | | | | | | |
| 实验名称 | strace 工具的使用和pstrace系统调用的应用 | | | 实验序号 | 6 | 实验日期 | 20.4.29 |
| 姓 名 | 沈思源 | 学 号 |  | 专 业 | 信息安全 | 年级-班 | 2018级-2班 |
| 1. 实验目的及实验内容   Strace是Linux中一个调试和跟踪工具。它可以接管被跟踪进程执行的系统调用和收到的信号。然后把每一个执行的系统调用的名字，参数和返回值打印出来。可以通过strace找到问题出现在user层还是kernel层。strace 显示这些调用的参数并返回符号形式的值。strace 从内核接收信息，而且不需要以任何特殊的方式来构建内核。它的实现基础是ptrace系统调用。  通过学习strace工具和分析其工作原理，可以进一步的加深对OS进程管理的认识，锻炼系统开发能力。 | | | | | | | |
| 【实验内容】   1. Strace工具的使用: 2. 跟踪一个程序执行过程中发生的系统调用 3. 统计系统调用的响应时间 4. 分析strace的实现原理 5. 基于ptrace系统调用实现进程跟踪 | | | | | | | |
| 1. 实验步骤及结果分析   （本次实验所使用的软件、工具的情况；具体的实验步骤） | | | | | | | |
| 实验环境  Ubuntu 18.04.2虚拟机环境Oracle VM VirtualBox  工具: gcc7.0编译器，vim  实验步骤:   1. 编写程序，首先简单利用strace工具跟踪程序的运行       Gcc编译，Strace工具命令为 strace ./{filename}，以此跟踪程序    根据strace工具反应的信息进行如下分析：  第2行表示通过系统调用execve来建立一个进程，本例中为test1对应的进程，在控制台中执行各种命令，比如ls、cd时，都是通过系统调用execve来建立它们的进程的，通过strace可以看到程序运行的细节。  ​ 第3行brk通过传递的addr来重新设置program break，成功则返回0，否则返回-1。 brk(0)的参数是一个地址，假如已经知道了堆的起始地址，还有堆的大小，那么就可以据此修改 brk() 中的地址参数已达到调整堆的目的。以0作为参数调用brk，返回值为内存管理的起始地址(若在子进程中调用malloc，则从0x8a92000地址开始分配空间)  ​ 第5~7行表示打开动态连接库的过程，如果程序是静态连接的，这几个步骤将不需要。  ​ 第13~18行是程序的处理过程，nanosleep()使得进程进入睡眠状态,指定时候后唤醒进程,sleep()基于其实现 ，然后写入hello和world。  ​ 第19~20行表示退出进程中所有的线程。  除此之外，strace工具还提供了参数选项，加入-c选项，可以打印调用了哪些系统函数，调用多少次数，消耗了多少时间等信息 ，用于性能分析。输入命令strace -c ./test    2、利用-c选项将进程的系统调用做一个统计分析并返回。    利用strace -c ./test对进程的系统调用进行跟踪统计    test程序一共调用了5次read，1次write函数，同时根据列表其他单元格信息可以清楚看到各个系统函数调用次数是多少，消耗了多少时间。  3、挂起进程的分析  Test1.c代码使用死循环模拟用户态挂死，调用sleep模拟内核态程序挂死，然后利用strace工具分析原因    直接利用strace ./{filename}命令查询进程信息    第12行，调用read，从/lib/i386-linux-gnu/libc.so.6该libc库文件中读取832bytes，即读取ELF头信息 。  ​ 分析：用户态挂死情况下，strace在getpid()一行输出之后没有其他系统调用输出；进程在内核态挂死，最后一行的系统调用nanosleep不能完整显示，这里nanosleep没有返回值表示该调用尚未完成。  ​ 结论：使用strace跟踪挂死程序，如果最后一行系统调用显示完整，程序在逻辑代码处挂死；如果最后一行系统调用显示不完整，程序在该系统调用处挂死。 当程序挂死在系统调用处，可以查看相应系统调用的man手册，了解在什么情况下该系统调用会出现挂死情况。  4、跟踪程序的系统退出exit()调用  编写带有该调用的程序test2.c    由于 ​ -e trace=process 表示只跟踪和进程管理相关的系统调用。在strace命令中加入相应的参数    进程自己退出时（调用exit函数，或者从main函数返回）, 最终调用的是exit\_group系统调用， 并且strace会输出exited with X（X为退出码）。  ​ 因为这里的exit函数不是系统调用，而是glibc库提供的一个函数，exit函数的调用最终会转化为exit\_group系统调用，它会退出当前进程的所有线程。实际上，有一个叫做\_exit()的系统调用(注意exit前面的下划线), 线程退出时最终会调用它。  5、跟踪查看依赖库问题       1. 分析strace的原理   Linux的标准strace工具源代码文件达到了5W+行，其大致逻辑功能模块之间的联系如下图所示。    Strace工具的实现基础是pstrace系统调用，最简单的strace命令的用法就是：strace PROG，PROG是要执行的程序。strace命令执行的结果就是按照调用顺序打印出所有的系统调用，包括函数名、参数列表以及返回值。使用strace跟踪一个进程的系统调用的基本流程如下图所示。    从图中可以看出strace做了以下几件事情：  1.   设置SIGCHLD 信号的处理函数，这个处理函数只要不是SIG\_IGN即可。由于子进程停止后是通过SIGCHLD信号通知父进程的，所以这里要防止SIGCHLD信号被忽略。  2.   创建子进程，在子进程中调用ptrace(PTRACE\_TRACEME,0L, 0L, 0L)使其被父进程跟踪，并通过execv函数执行被跟踪的程序。  3.   通过wait()等待子进程停止，并获得子进程停止时的状态status。  4.   通过子进程的状态查看子进程是否已正常退出，如果是，则不再跟踪，随后调用ptrace发送PTRACE\_DETACH请求解除跟踪关系。  5.   子进程停止后，打印系统调用的函数名、参数和返回值。具体流程见图2。  6.   通过PTRACE\_SYSCALL让子进程继续运行，由于这个请求会让子进程在系统调用的入口处和系统调用完成时都会停止并通知父进程，这样，父进程就可以在系统调用开始之前获得参数，结束之后获得返回值。  在系统调用的入口和结束时子进程停止运行时，这时父进程认为子进程是因为收到SIGTRAP信号而停止的。所以父进程在wait()后可以通过SIGTRAP来与其他信号区分开。  Strace中为每个要跟踪的进程维护了一个TCB（Trace Control Block）结构，定义如下。它保存了当前发生的系统调用的信息。    子进程会在系统调用前后各停止一次，所以打印系统调用信息时分为两个阶段：在系统调用开始时可以获取系统调用号和参数，在系统调用结束时可以获取系统调用的返回结果。通过给tcb结构的flags字段清除和添加TCB\_INSYSCALL标志位来区分系统调用的开始和结束。    7、利用pstrace跟踪进程  ptrace()是一个系统调用,它允许一个进程控制另外一个进程的执行.不仅如此,我们还可以借助于ptrace修改某个进程的空间(内存或寄存器),任何传递给一个进程(即被跟踪进程)的信号(除了会直接杀死进程的SIGKILL信号)都会使得这个进程进入暂停状态,这时系统通过wait()通知跟踪进程,这样,跟踪进程就可以修改被跟踪进程的行为了.  如果跟踪进程在被跟踪进程的内存中设置了相关的事件标志位,那么运行中被跟踪进程也可能因为特殊的事件而暂停.跟踪结束后,跟踪进程甚至可以通过设置被跟踪进程的退出码(exit code)来杀死它,当然也可以让它继续运行.  Ptrace系统调用的函数原型:      .参数request：请求ptrace执行的操作  .参数pid：目标进程的ID  .参数addr：目标进程的地址值  .参数data：作用则根据request的不同而变化，如果需要向目标进程中写入数据，data存放的是需要写入的数据；如果从目标进程中读数据，data将存放返回的数据  编写ptrace测试程序      编写my\_own\_strace，  execl()函数对应的系统调用为\_\_NR\_execve，系统调用值为59。父进程通过调用fork()来创建子进程。在子进程中，先运行patrce().请求参数设为PTRACE\_TRACE,来告诉内核当前进程被父进程trace，每当有信号量传递到当前进程，该进程会暂停，提醒父进程在wait()调用处继续执行。然后再调用execl()。当execl()函数成功执行后，新程序运行之前，SIGTRAP信号量会被发送到该进程，让子进程停止，这时父进程会在wait相关调用处被通知，获取子进程的控制权，可以查看子进程内存和寄存器相关信息。  当进程进行系统调用时，int会在内核栈中依次压入用户态的寄存器SS、ESP、EFLAGS、CS、EIP.中断处理程序的SAVE\_ALL宏会将 依次将EAX、EBP、EDI、ESI、EDX、ECX、EBX寄存器值压入内核栈。调用ptrace(PTRACE\_PEEKUSER,child,8\*ORIG\_RAX,NULL) 获取USER area信息时<sys/reg.h>文件定义了与内核栈寄存器数组顺序相同的下标  WIFEXITED函数(宏)函数用来检查子进程是暂停还准备退出。  相关代码文件为my\_own\_strace,一共有3个版本，执行命令为:my\_own\_strace {filename};  程序源代码部分截图    运行测试截图 | | | | | | | |
| 1. 总结   （对实验结果进行分析，总结实验的新的体会，并提出实验的改进意见）  通过学习strace工具和分析其工作原理，进一步的加深了对OS进程管理的认识，锻炼系统开发能力。  灵活运用OS自带的系统调用是进行地层开发的有效捷径  Strace工具在之后的程序开发和调试中依然可以发挥巨大作用，值得学习 | | | | | | | |
| 。 | | | | | | | |