|  |
| --- |
|  |
| 1. 实验目的及实验内容   （本次实验所涉及并要求掌握的知识；实验内容；必要的原理分析） |
| 1. 向Linux增加一个系统调用 2. 撰写一个应用测试程序调用该系统调用 3. 使用ptrace或类似的工具对该测试程序进行跟踪调试 |
| 1. 实验环境及实验步骤   （本次实验所使用的器件、仪器设备等的情况；具体的实验步骤） |
| 实验环境：  1. vmware workstation 15.0.0   1. ubuntu-18.10-desktop 2. 内核版本：linux-4.18.14   实验步骤：  一．准备环境  在终端操作：  sudo apt-get update  //更新系统软件包  sudo apt-get install vim //安装vim  sudo apt-get install libncurses5-dev libssl-dev  //下载依赖包  sudo apt-get install libelf-dev  sudo apt-get install build-essential openssl  sudo apt-get install zlibc minizip  sudo apt-get install libidn11-dev libidn11  sudo apt-get install flex bison  二．构建新内核 （下载，添加功能，编译，安装）  **1.准备：**  A.在www.kernel.org下载[linux-4.18.14.tar.gz](http://ftp.jaist.ac.jp/pub/Linux/kernel.org/linux/kernel/v4.x/linux-4.18.14.tar.gz)  B.  解压  将在官网下好的内核直接拉到ubuntu里，然后将压缩包用指令移到/usr/src目录下（过程需要root权限）。         sudo su                                            //获取管理员权限         sudo mv  压缩包位置    /usr/src         //移动文件         cd  /usr/src                                 //切换到该目录下进行操作        sudo tar -xvflinux-4.18.14.tar.gz     //解压  **2. 添加系统调用**  **注意：先将目录切换到解压后的内核里操作**     1. sudo vim  kernel/sys.c   //添加自己的函数   这里添加了sys\_helloworld     1. sudo vim arch/x86/include/asm/syscalls.h //添加声明      1. sudo vimarch/x86/entry/syscalls/syscall\_64.tbl  //添加调用号     **3.删除无用的文件**  sudo make mrproper  sudo make clean   1. **配置**   sudo make menuconfig  此处直接save，exit即可，无需修改     1. **编译新内核并安装**   A.编译  ***sudo make -j2*** //将处理机内核总数调整为4，会明显加快编译速度    （-j2为开启双线程编译）  B.安装  sudo make modules\_install  sudo make install  **C.重启**  reboot  **三.测试**  先使用vim写一个测试的cpp,命名为test2.cpp    之后gcc编译（前提是apt-get install build-essential安装gcc/g++/gdb/make 等基本编程工具。）    然后运行    demsg   查看进程信息  解释：我们之前添加的调用内部是printk。使用printk时，[Linux内核](https://www.baidu.com/s?wd=Linux%E5%86%85%E6%A0%B8&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "https://blog.csdn.net/qustdrjhj/article/details/_blank)根据日志级别，可能把消息打印到当前控制台上，这个控制台是一个字符设备。这些消息从终端输出的前提是日志输出级别小于console\_loglevel。（越小级别越高）      上图最后一行说明系统调用添加成功。  **四．使用ptrace或类似的工具对测试程序进行跟踪调试**   1. **先用strace，可以查看程序的系统调用。**   **A.**  下图中read（0, 就是程序中a的输入，这里我输入19.  而syscall\_0x150则是测试程序中的系统调用，0x150 16进制转化为10进制即为336，是我们刚刚添加的系统调用号。  write则为printf。    **B.**再用strace -c 可以查看测试程序的各类系统调用。如下图。    **2.接下来用gdb调试。gdb在ptrace的基础上得来，功能强大且简便。**    注意要用gdb调试，gcc编译.cpp文件时要加上-g。  A.我们这里仅仅简单的调试。    输入l可以查看程序源码。    输入r可以运行程序。    输入b x，可以再在x行前添加断点。    添加断点后先r，再用c可以一步步执行，每次到断点停止。    clear可以清除断点  在return前插入断点，运行，输入bt可以查看堆栈调用。 |
| 1. 实验过程分析   （实验分工，详细记录实验过程中发生的故障和问题，进行故障分析，说明故障排除的过程及方法。根据具体实验，记录、整理相应的数据表格等） |
| 遇到的问题：   1. windows下载好的kernel安装包如何移动到虚拟机里的问题。   解决：为了将下载好的kernel从主操作系统移动到虚拟机中，安装了vmware tools   1. 编译速度过慢的问题   解决：为了加快编译速度，在虚拟机的硬件设置里分配了2个处理器两个内核，并开启多线程编译。   1. 添加完系统调用后，测试程序运行失败的问题   第一次添加完并安装后编写测试程序，运行无法打印hello world。  解决：检查后发现在声明时函数名拼写错误，world拼为wrold。修改后重新编译解决。   1. 测试程序打印消息无法在终端显示的问题   解决：因为内核添加的函数用的是printk，在查阅资料后发现是否打印和输出级别有关，故用dmesg查看。   1. 在终端cd操作的问题   刚开始cd操作出现了失败  解决：查阅资料发现ubuntu终端区分大小写   1. 切换内核的问题   解决：ubuntu 重启界面长按shift可以进入高级选项界面，选择内核。   1. 遇到虚拟机硬盘空间不足的问题   解决：查询资料后扩容   1. 其他问题   在过程中初次使用vim，gdb，strace，有很多不懂的地方。  解决：查看vim，gdb，strace的使用文档   1. 编译内核过程中编译printk函数出现问题，查看发现是下载某些软件包时无法定义软件包位置   解决：在Ubuntu系统软件与更新处更换镜像网站，切换到清华镜像网站  实验记录：见二、实验步骤中的图片 |