课程实验报告

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 实验名称 | Linux安装及虚拟机的使用 | | | | | 实验序号 | 3 |
| 姓 名 | 唐承模 | 系院专业 | 信管 | 班 级 | 18 | 学 号 | 10207180586 |
| 实验日期 | 2012年12月 | | 指导教师 | 丘文峰 | | 成 绩 |  |
| **一、实验目的和要求**  1.加深对进程概念的理解，明确进程和程序的区别  2.进一步认识并发执行的实质  3分析进程争用资源的现象，学习解决进程互斥的方法  4了解Linux系统中进程通信的基本原理 | | | | | | | |
| **二、实验预习内容**  1.阅读Linux的sched.h源码文件，加深对进程管理概念的理解  2.阅读Linux的fork.c源码文件，分析进程的创建过程 | | | | | | | |
| **三、实验设备**  1. 操作系统：Linux red hat  2. 安装软件： VirtualBox和red hat | | | | | | | |
| **四、实验内容及步骤**  1.进程的创建  编写一段程序，使用系统调用fork()创建两个子进程。当此程序运行时，在系统中有一个进程 和两个子进程活动。让每一个进程在屏幕上显示一个字符:父进程显示字符"a";子进程分别显示字 符"b"和字符"c"。试观察记录屏幕上的显示结果，并分析原因。  2进程的控制  修改已编写的程序，将每个进程输出一个字符改为每个进程输出一句话，在观察程序行时屏幕上 出现的现象，并分析原因。  3.如果在程序中使用系统调用lockf()来给每一个进程加锁，可以实现进程之间的互斥，观察并分析 出现的现象  4. 软中断通信  编制一段程序，使其实现进程的软中断通信。要求：使用系统调用fork()创建两个子进程，再用系统调用signal()让父进程捕捉键盘上来的中断 信号（即按Del键）;当捕捉到中断信号后，父进程用系统调用kill()向两个子进程发出信号，子进程捕捉 到信号后分别输出下列信息后终止：  child process 1 is killed by parent!  child process 2 is killed by parent!  父进程等待两个子进程终止后，输出如下的信息后终止：  parent process is killed!  在上面的程序中增加语句signal(SIGINT, SIG\_IGN)和signal(SIGQUIT, SIG\_ING)观察行结果，并分析原因  5. 进程的管道通信  编制一段程序，实现进程的管道通信。使用系统调用pipe()建立一条管道栈；两个子进程P1和P2分别向管道各写一句话：  child 1 is sending a message!  child 2 is sending a message!  而父进程则从管道中读出来自于两个子进程的信息，显示出屏幕上。 要求父进程先接  进程P1发来的消息，然后再接收子进程P2发来的消息。 | | | | | | | |
| **五、实验结果、分析与心得**  **1.实验结果**  **1.** 进程创建  **C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\{T(O{9Z{}B1]K24$M{DJ3]R.jpg**  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\}WR1_{TC]ZGJ9[`VH}WS7YI.jpg  将while(p1=fork()==-1)语句改为while(（p1=fork()）==-1)语句后  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\@NIA@IUF}_Z~FOGUWW}1GW5.jpg  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\B__QBL3LDG1LIBXE(SM3QUJ.jpg  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\E649$5%JW2P@OU3I1_9%0$H.jpg  2.进程控制  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\5O45OLM0S]0F5TW16N~AIOL.jpg  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\LMNZ(6GS53OI[7IW2@6C~)E.jpg  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\`9N%OU[$AO99(6UAELNI79H.jpg  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\[QYA3@@JZQ[~CPNN7}QC69W.jpg  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\DJ]F(4GP}YHT]SG0[0)PW8W.jpg  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\RRHUZ5Z0]}PIGOD5RUDY%LN.jpg  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\Z3KP7B1P4~(SQ(G6AYKELNI.jpg  2.  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\I1UHJ[V$Q(5{0}M1YX~{C~N.jpg  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\LPM~3{SJX0VVB5L9{K3)5PP.jpg  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\]MX}JGCCSB732ZQR)$GAU_J.jpg  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\_YLA]UJK8DU$@~CU_51YD4K.jpg  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\ICJS5`AW0{Q3F3~BQ3UP)GH.jpg  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\OH~TCX@V8UVUSOR)7{U3)1Y.jpg  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\DNC)3JTR[$TK3L1`_0~X(C4.jpg  3.加锁后  1.  . C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\YX433HQ]`P8_D[`B75[WUZT.jpg  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\J~ZQ_XGK{NBE$5Q9C8JV6QK.jpg  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\E@{K]3T]`U{9OVYW1VFK2BK.jpg  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\PEI~2_E3X%8(SD5HN_6M67S.jpg  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\KW$77GIWPG2`0UP[UL(WHNK.jpg    C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\7VB{)T(3Z6L$EOJH$9XP554.jpg  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\9]W[S~%Y_KZRPUOBAROI8FR.jpg  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\LBO[W78N{(VF1}YO%G((3VN.jpg  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\4(QA]2J~LL5ZYJ%E{4DO8~H.jpg  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\X8XDE5SJWVDFRH`O`ORU0N3.jpg  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\CDPB}83]1W@Y893HQ7DWBUI.jpg  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\}N@RHOIB~$PJGHZ%B%$)S52.jpg  4.软中断  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\XGCM7U56HSJQ6SNVCVM8~SE.jpg  5.任务2  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\%GG@DL1AHF}LY0PB)Q0GS{X.jpg  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\HT{J]YR3`@1JC]T4[835`9K.jpg  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\11L`ERN~P%[X$M3(64HHQ9S.jpg  6.通道通信  C:\Users\abcd\AppData\Roaming\Tencent\Users\389796404\QQ\WinTemp\RichOle\N3UR[FREJQF`B_VWD}317SR.jpg  **2．分析结果**  1.由于执行第一句while(p1=fork()==-1)，大多数情况都会创建了一个子进程，然而子进程的fork（）函数返回0，父进程的fork（）函数返回子进程的id号.>0,因此经过while(p1=fork()==-1)后p1=0,则当子进程和父进程并发执行时都是只执行 if (p1==0) putchar('b')语句，输出两个b，并不会在父进程种2创建第二个子进程；有只输出一个b的情况，应该是子进程创建失败，只执行父进程，且不会再创建第二个子进程，只执行一次if (p1==0) putchar('b')语句输出一个b。如果要创建两个子进程，并输出bca，则应该在第一句while(p1=fork()==-1)语句改为while(（p1=fork()）==-1)语句。  改进之后：  输出的序列既可以是bca，bac等序列是由于三个进程间随机并发执行的原因，而出现ba，ca等序列是由于第一个子进程和第二个子进程没有被成功建立的结果。    2.输出的child、son、daughter的顺序就是随机的了，原因是输出child的子进程，daughter的子进程都已被建立，则三个进程之间是并发执行的。    3.child、son、daughter之间的程序是随机切换的，但是每当执行任何一个进程时都会一次执行到底，不会执行到一半就由于并发转去执行另外一个进程，这是由于对每个进程进行了加锁的结果。由于加锁，使得各进程间的执行时互斥的，因此才不会像进程控制那样，不但进程间的切换时随机的，进程之间的执行步骤也是交叉并被执行的。  .4  虚拟机无法接受delete，无法完全实现实验目的    5. 父进程则从管道中读出来自于两个子进程的信息，显示出屏幕上。 父进程先接收子进程P1发来的消息，然后再接收子进程P2发来的消息，于是得到上述结果      **3．心得**   1. 要学会看懂编译时给出的错误提示，然后按图索骥去纠错； 2. 程序员只管错误，不管警告；   3、遇到问题要独立思考，寻求解决办法，实在弄不懂试着找同学讨论。 | | | | | | | |
| **教师评语：**  **成绩:**  **教师签字：** | | | | | | | |