Linux操作系统的X86汇编程序设计（4） PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/144/2021_2022_Linux_E6_93_8D_E4_BD_c103_144990.htm 在 Linux 下使用中断 Linux 是一个运行在保护模式下的共享库的环境, 意味着没有中断服务, Right? 错了. 我注意到在GAS 的例子源码中用了 INT 80, 注释是 "sys_write(ebx, ecx, ed x)". 这个函数是 Linux 系统调用接口的一部分, 意思是 INT 80 必须是到达系统调用服务的门户. 在 Linux 源码中到处看时(忽略从不要使用 INT 80 接口的警告, 因为函数号可能随时改变), 我发现 "系统调用号(system call numbers)" -- 就是说, 传给 INT 80 的 # 对应着一个系统调用子程序 -- 在 UNISTD.H 中. 一共有 189 个, 所以我不会在 这里列出来...但如果你在 Linux 做汇编, 给自己做个好事, 打印出来吧. 当调用 INT 80 时, eax 设为用调用的功能号. 传给系统调用则程序的参数必须按顺序 放在下列寄存器中: ebx, ecx, edx, esi, edi 这样, 第一个参数就在 ebx 里, 第二个在 ecx 里... 注意在一个系统调用程序里, 不是用栈来传递参数. 调用的返回值在 eax 里. 还有, INT 80 接口和一般的调用一样. 下面的这个程序就演示了 INT 80h 的使用。 这个程序检查并显示了它自己的 PID. 注意 使用 printf() 格式化字符串 -- 这个调用的 C 结构是: printf("%d\n", curr_PID).也要注意结束符在汇编里不一定可靠, 我常用十六进制(0Ah, 0Dh)代表 CR\LF. .pid.asmBITS 32 GLOBAL main EXTERN printf SECTION .data szText1 db Getting Current Process ID...,0Ah,0Dh,0 szDone db Done!,0Ah,0Dh,0 szError db Error in int 80!,0Ah,0Dh,0 szOutput db \%d,0Ah,0Dh,0 .printf() 的格式字符串 SECTION .text main: push dword szText1 .开始信息

call printf pop ecx GetPID: mov eax, dword 20 . getpid() 系统调用 int 80h . 系统调用中断 cmp eax, 0 . 没有 PID 0 ! :) jb Error push eax . 把返回值传递给 printf push dword szOutput . 把格式字符串传递给 printf call printf pop ecx . 清除栈 pop ecx push dword szDone . 结束信息 call printf pop ecx jmp Exit Error: push dword szError call printf pop ecx Exit: ret . EOF最后的话 大多数的麻烦来自对 Nasm 的习惯上. 而 nasm 带有手册, 但缺省是不安装的, 所以你必须把它从 /user/local/bin/nasm-0.97/nasm.man移(cp 或 mv)到 /usr/local/man/man1/nasm.man.格式有点乱, 可以很简单的用 nroff 指示符来解决. 但它不会给你 Nasm 的整个文档. 要解决这个问题, 把 nasmdoc.txt 从 /usr/local/bin/nasm-0.97/doc/nasmdoc.txt拷贝到 /usr/local/man/man1/nasmdoc.man现在你可以用 man nasm, man nasmdoc 来看 nasm 的手册和文档了想得到更多的信息, 查查这里:Linux Assembly Language HOWTO (Linux 汇编语言 HOWTO) Linux I/O Port Programming Mini-HOWTO (Linux I/O 端口编程 Mini-HOWTO) Jans Linux &amp. Assembler HomePage (http://www.bewoner.dma.be/JanW/eng.html) 我也要感谢 Jeff Weeks(http://gameprog.com/codex), 在我找到 Jan 的网页之前 给了我一些 GAS 的 hello-world 代码。