SCOUNIX到Linux操作系统的程序移植问题 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/267/2021_2022_SCOUNIX_E5_88_B0_c103_267101.htm 要求把一个原先运行在SCO UNIX操作系统的柜面系统移植到Red Hat Linux AS平台上。现在好像有一个趋势，金融行业的的原先运行在SCO下的系统都准备迁移到LINUX下。 下面简要整理一下，迁移过程中的一些心得体会。 一、SCO 的cc 与LINUX 的gcc的一些差别 总的来说，linux的gcc编译器相对sco下的cc要严格许多。通过这次移植发现SCO的cc对程序的要求实在不怎么严谨。举个例子，比如strcpy()函数应该是2个函数，如果你给他3个参数，编译也能通过。还有，如果一个函数的参数应该是传值，你给它传一个地址，cc也不会报错。 SCO的cc与linux的gcc在有关空指针的处理上的差别是最明显的。比如，strcpy(),strncpy(),strcmp(),strncmp(),fclose()。在SCO上，如果参数有一个是空指针，程序不会core，但在LINUX下，这些函数只要有一个是空指针，程序运行过程中就会core。在移植过程中，我们发现只要程序运行过程中出现core，十有八九是因为空指针的问题。 因此，移植的第一步，我么就对上述常见的字符串操作函数，做了一层封装，然后用封装过的函数来全局替换原来的函数。 二、gdb的使用 由于以前没在liunx下写过程序，对gdb调试工具也没有使用过。这次移植还学会了gdb的一些基本调试步骤。 gdb 可执行程序 b 设置断点 r 运行程序 c 端点后重新运行程序 n 执行下一条语句 s 进入到函数体内调试（相对于n） attach PID 调试正在运行的程序 三、关于core文件 在SCO下，一般程序core时，都会在可执行

目录下生成一个core文件，我们可以使用dbx 来查看程序的什么地方出现了core。 移植到LINUX下，一开始，程序core时，怎么都没有生成core文件。后来，才发现，需要人为设定core文件所允许的最大值。如果没有设定，默认是0，也就不会生成core文件。 设定方法如下： 执行 ulimit -c 102400，可以把这个命令放在用户的登录shell里面，这样不用每次登录时重新设置了。 使用“ gdb 可执行程序名 core文件名 ”可以查看大致在什么地方程序出现core。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com