GCC编译的四个阶段Linux认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/644/2021_2022_GCC_E7_BC_96_E8_AF_91_E7_c103_644768.htm 在使用GCC编译程序时,编译过程可以被细分为四个阶段: 预处理（Pre-Processing）编译（Compiling）汇编（Assembling）链接（Linking）。 例如: #include gt. int main(void) { printf ("Hello world, Linux programming!\n"). return 0. } 然后执行下面的命令编译和运行这段程序： # gcc hello.c -o hello # ./hello Hello world, Linux programming! GCC需要调用预处理程序cpp,由它负责展开在源文件中定义的宏,并向其中插入“ #include ”语句所包含的内容.接着,GCC会调用ccl和as将处理后的源代码编译成目标代码.最后,GCC会调用链接程序ld,把生成的目标代码链接成一个可执行程序。 第一步是进行预编译,使用-E参数可以让GCC在预处理结束后停止编译过程： # gcc -E hello.c -o hello.i 此时若查看hello.cpp文件中的内容,会发现stdio.h的内容确实都插到文件里去了,而其它应当被预处理的宏定义也都做了相应的处理。下一步是将hello.i编译为目标代码,这可以通过使用-c参数来完成： # gcc -c hello.i -o hello.o GCC默认将.i文件看成是预处理后的C语言源代码,因此上述命令将自动跳过预处理步骤而开始执行编译过程,也可以使用-x参数让GCC从指定的步骤开始编译。最后一步是将生成的目标文件链接成可执行文件： # gcc hello.o -o hello 在采用模块化的设计思想进行软件开发时,通常整个程序是由多个源文件组成的,相应地也就形成了多个编译单元,使用GCC能够很好地管理这些编译单元。假设有一个由foo1.c和foo2.c两个源文件组成的程序,为了对它们进行编

译,并最终生成可执行程序foo,可以使用下面这条命令：# gcc foo1.c foo2.c -o foo 如果同时处理的文件不止一个,GCC仍然会按照预处理、编译和链接的过程依次进行。如果深究起来,上面这条命令大致相当于依次执行如下三条命令：# gcc -c foo1.c -o foo1.o # gcc -c foo2.c -o foo2.o # gcc foo1.o foo2.o -o foo 在项目文件代码较多时,还要借助像Make这样的工具。 GDB调试： 列文件清单 List (gdb) list line1,line2 显示数据 print 检查各个变量的值(gdb) print p (p为变量名) whatis 显示某个变量的类型 (gdb) whatis p 断点(breakpoint) break line-number 使程序恰好在执行给定行之前停止。 break function-name 使程序恰好在进入指定的函数之前停止。 break line-or-function if condition 如果condition（条件）是真，程序到达指定行或函数时停止。 break routine-name 在指定例程的入口处设置断点 (gdb) break filename:line-number 如果该程序是由很多原文件构成的，在各个原文件中设置断点 (gdb) break filename:function-name 要想设置一个条件断点，可以利用break if命令，如下所示：(gdb) break line-or-function if expr 例：(gdb) break 46 if testsize==100 countinue 命令断点继续运行 显示当前gdb的断点信息：(gdb) info break 会以如下的形式显示所有的断点信息：Num Type Disp Enb Address What 1 breakpoint keep y 0x000028bc in init_random at qsort2.c:155 2 breakpoint keep y 0x0000291c in init_organ at qsort2.c:168 (gdb) 删除指定的某个断点：(gdb) 0delete breakpoint 1 该命令将会删除编号为1的断点，如果不带编号参数，将删除所有的断点 (gdb) 0delete breakpoint 禁止使用某个断点 (gdb) disable breakpoint 1 该命令将禁止断点 1,同时断点信息的 (Enb)域将变为 n 允许使用某个

断点 (gdb) enable breakpoint 1 该命令将允许断点 1,同时断点信息的 (Enb)域将变为 y 清除原文件中某一代码行上的所有断点 (gdb)clean number 注：number 为原文件的某个代码行的行号 信号 gdb 通常可以捕捉到发送给它的大多数信号，通过捕捉信号，例如，按CTRL-C将中断信号发送给gdb，通常就会终止gdb。但是你或许不想中断gdb，真正的目的是要中断gdb正在运行的程序，因此，gdb要抓住该信号并停止它正在运行的程序，这样就可以执行某些调试操作。 Handle命令可控制信号的处理，他有两个参数，一个是信号名，另一个是接受到信号时该作什么。 