Lenky个人站点

色不异空 . 空不异色 . 色即是空 . 空即是色

请输入关键字...

gcc选项-g与-rdynamic的异同

2013年1月13日 lenky 发表评论 阅读评论 9,038 次浏览

gcc的-g，应该没有人不知道它是一个调试选项，因此在一般需要进行程序调试的场景下，我们都会加上该选项，并且根据调试工具的不同，还能直接选择更有针对性的说明，比如-ggdb。-g是一个编译选项，即在源代码编译的过程中起作用，让gcc把更多调试信息（也就包括符号信息）收集起来并将存放到最终的可执行文件内。
相比-g选项，-rdynamic却是一个连接选项，它将指示连接器把所有符号（而不仅仅只是程序已使用到的外部符号，但不包括静态符号，比如被static修饰的函数）都添加到动态符号表（即.dynsym表）里，以便那些通过dlopen()或backtrace()（这一系列函数使用.dynsym表内符号）这样的函数使用。

看示例：

```
1  [root@www c]# cat t.c
2  #include <stdio.h>
3  void bar() {}
4  void baz() {}
5  void foo() {}
6  int main() { foo(); printf("test"); return 0; }
```

对于上面的示例代码，普通和加-g编译：

```
 1  [root@www c]# uname -a
 2  Linux www.t1.com 2.6.38.8 #2 SMP Wed Nov 2 07:52:53 CST 201
 3  [root@www c]# gcc -O0 -o t t.c
 4  [root@www c]# gcc -O0 -g -o t.g t.c
 5  [root@www c]# readelf -a t > t.elf
 6  [root@www c]# readelf -a t.g > t.g.elf
 7  [root@www c]# ls -lh *.elf t t.g
 8  -rwxr-xr-x. 1 root root 6.6K Jul 24 06:50 t
 9  -rw-r--r--. 1 root root  15K Jul 24 06:51 t.elf
10  -rwxr-xr-x. 1 root root 7.9K Jul 24 06:50 t.g
```

```
-rw-r--r--. 1 root root  16K Jul 24 06:51 t.g.elf
```

加-g编译后，因为包含了debug信息，因此生成的可执行文件偏大（程序本身非常小，所以增加的调试信息不多）。
看-g编译的符号表：

```
[root@www c]# readelf -s t

Symbol table '.dynsym' contains 4 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx N
     0: 0000000000000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND
     2: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND
     3: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND

Symbol table '.symtab' contains 67 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx N
...
    48: 00000000004003e0     0 FUNC    GLOBAL DEFAULT   13
    49: 00000000004004c4     6 FUNC    GLOBAL DEFAULT   13
...
    53: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND
    54: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND
    55: 00000000004005e8     4 OBJECT  GLOBAL DEFAULT   15
    56: 00000000004004d0     6 FUNC    GLOBAL DEFAULT   13
...
    64: 00000000004004d6    31 FUNC    GLOBAL DEFAULT   13
    65: 0000000000400390     0 FUNC    GLOBAL DEFAULT   11
    66: 00000000004004ca     6 FUNC    GLOBAL DEFAULT   13
```

注意.dynsym表，只有该程序用到的几个外部动态符号存在。
加-rdynamic选项编译，readelf查看：

```
[root@www c]# gcc -O0 -rdynamic -o t.rd t.c
[root@www c]# readelf -s t.rd

Symbol table '.dynsym' contains 20 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx N
     0: 0000000000000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND
     2: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND
     3: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND
     4: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND
     5: 0000000000400724     6 FUNC    GLOBAL DEFAULT   13
     6: 0000000000400730     6 FUNC    GLOBAL DEFAULT   13
     7: 0000000000600b68     0 NOTYPE  GLOBAL DEFAULT   24
     8: 0000000000600b80     0 NOTYPE  GLOBAL DEFAULT  ABS
     9: 0000000000600b6c     0 NOTYPE  GLOBAL DEFAULT  ABS
    10: 0000000000600b68     0 NOTYPE  WEAK   DEFAULT   24
    11: 0000000000400640     0 FUNC    GLOBAL DEFAULT   13
    12: 0000000000400848     4 OBJECT  GLOBAL DEFAULT   15
    13: 0000000000400770   137 FUNC    GLOBAL DEFAULT   13
    14: 0000000000600b6c     0 NOTYPE  GLOBAL DEFAULT  ABS
```

```
    15: 0000000000400736    39 FUNC    GLOBAL DEFAULT   13 r
    16: 00000000004005f0     0 FUNC    GLOBAL DEFAULT   11
    17: 0000000000400760     2 FUNC    GLOBAL DEFAULT   13
    18: 0000000000400838     0 FUNC    GLOBAL DEFAULT   14
    19: 000000000040072a     6 FUNC    GLOBAL DEFAULT   13 i

Symbol table '.symtab' contains 67 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx N
...
    50: 0000000000400640     0 FUNC    GLOBAL DEFAULT   13
    51: 0000000000400724     6 FUNC    GLOBAL DEFAULT   13 i
...
    55: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND
    56: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND
    57: 0000000000400848     4 OBJECT  GLOBAL DEFAULT   15
    58: 0000000000400730     6 FUNC    GLOBAL DEFAULT   13
...
    64: 0000000000400736    31 FUNC    GLOBAL DEFAULT   13 r
    65: 00000000004005f0     0 FUNC    GLOBAL DEFAULT   11
    66: 000000000040072a     6 FUNC    GLOBAL DEFAULT   13 i
[root@www c]#
```

可以看到添加-rdynamic选项后，.dynsym表就包含了所有的符号，不仅是已使用到的外部动态符号，还包括本程序内定义的符号，比如bar、foo、baz等。
.dynsym表里的数据并不能被strip掉：

```
[root@www c]# strip t.rd
[root@www c]# readelf -s t.rd

Symbol table '.dynsym' contains 20 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx N
     0: 0000000000000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND
     2: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND
     3: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND
     4: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND
     5: 0000000000400724     6 FUNC    GLOBAL DEFAULT   13 i
     6: 0000000000400730     6 FUNC    GLOBAL DEFAULT   13
     7: 0000000000600b68     0 NOTYPE  GLOBAL DEFAULT   24
     8: 0000000000600b80     0 NOTYPE  GLOBAL DEFAULT  ABS
     9: 0000000000600b6c     0 NOTYPE  GLOBAL DEFAULT  ABS
    10: 0000000000600b68     0 NOTYPE  WEAK   DEFAULT   24
    11: 0000000000400640     0 FUNC    GLOBAL DEFAULT   13
    12: 0000000000400848     4 OBJECT  GLOBAL DEFAULT   15
    13: 0000000000400770   137 FUNC    GLOBAL DEFAULT   13
    14: 0000000000600b6c     0 NOTYPE  GLOBAL DEFAULT  ABS
    15: 0000000000400736    39 FUNC    GLOBAL DEFAULT   13 r
    16: 00000000004005f0     0 FUNC    GLOBAL DEFAULT   11
    17: 0000000000400760     2 FUNC    GLOBAL DEFAULT   13
    18: 0000000000400838     0 FUNC    GLOBAL DEFAULT   14
    19: 000000000040072a     6 FUNC    GLOBAL DEFAULT   13 i
```

简单总结一下-g选项与-rdynamic选项的差别：

1，-g选项新添加的是调试信息（一系列.debug_xxx段），被相关调试工具，比如gdb使用，可以被strip掉。

2，-rdynamic选项新添加的是动态连接符号信息，用于动态连接功能，比如dlopen()系列函数、backtrace()系列函数使用，不能被strip掉，即强制strip将导致程序无法执行：

```
[root@www c]# ./t.rd
test[root@www c]# strip -R .dynsym t.rd
[root@www c]# ./t.rd
./t.rd: relocation error: ./t.rd: symbol , version GLIBC_2.2
[root@www c]#
```

3，.symtab表在程序加载时会被加载器丢弃，gdb等调试工具由于可以直接访问到磁盘上的二进制程序文件：

```
[root@www c]# gdb t.g -q
Reading symbols from /home/work/dladdr/c/t.g...done.
(gdb)
```

因此可以使用所有的调试信息，这包括.symtab表；而backtrace()系列函数作为程序执行的逻辑功能，无法去读取磁盘上的二进制程序文件，因此只能使用.dynsym表。
其它几个工具可以动态指定查看，比如nm、objdump：

```
[root@www c]# nm t.rd
nm: t.rd: no symbols
[root@www c]# nm -D t.rd
0000000000400848 R _IO_stdin_used
                 w _Jv_RegisterClasses
0000000000600b6c A __bss_start
0000000000600b68 D __data_start
                 w __gmon_start__
0000000000400760 T __libc_csu_fini
0000000000400770 T __libc_csu_init
                 U __libc_start_main
0000000000600b6c A _edata
0000000000600b80 A _end
0000000000400838 T _fini
00000000004005f0 T _init
0000000000400640 T _start
0000000000400724 T bar
000000000040072a T baz
0000000000600b68 W data_start
0000000000400730 T foo
0000000000400736 T main
                 U printf
[root@www c]#
[root@www c]# objdump -T t.rd

t.rd:     file format elf64-x86-64

DYNAMIC SYMBOL TABLE:
0000000000000000      DF *UND*  0000000000000000  GLIBC_2.2
0000000000000000  w   D  *UND*  0000000000000000
```

```
31 | 0000000000000000  w   D  *UND*  0000000000000000
32 | 0000000000000000      DF *UND*  0000000000000000  GLIBC_2.2
33 | 0000000000400724 g    DF .text  0000000000000006  Base
34 | 0000000000400730 g    DF .text  0000000000000006  Base
35 | 0000000000600b68 g    D  .data  0000000000000000  Base
36 | 0000000000600b80 g    D  *ABS*  0000000000000000  Base
37 | 0000000000600b6c g    D  *ABS*  0000000000000000  Base
38 | 0000000000600b68  w   D  .data  0000000000000000  Base
39 | 0000000000400640 g    DF .text  0000000000000000  Base
40 | 0000000000400848 g    DO .rodata    0000000000000004  Base
41 | 0000000000400770 g    DF .text  0000000000000089  Base
42 | 0000000000600b6c g    D  *ABS*  0000000000000000  Base
43 | 0000000000400736 g    DF .text  0000000000000027  Base
44 | 00000000004005f0 g    DF .init  0000000000000000  Base
45 | 0000000000400760 g    DF .text  0000000000000002  Base
46 | 0000000000400838 g    DF .fini  0000000000000000  Base
47 | 000000000040072a g    DF .text  0000000000000006  Base
```

4，-rdynamic选项不产生任何调试信息，因此在一般情况下，新增的附加信息比-g选项要少得多。除非是完全的静态连接，否则即便是没有加-rdynamic选项，程序使用到的外部动态符号，比如前面示例里的printf，也会被自动加入到.dynsym表。

完全参考：
http://stackoverflow.com/questions/8623884/gcc-debug-symbols-g-flag-vs-linkers-rdynamic-option

转载请保留地址：http://www.lenky.info/archives/2013/01/2190 或 http://lenky.info/?p=2190

分类: *nix技术, 跟踪调试 标签: -g, -rdynamic, gcc

类似文章

- 2013 年 1 月 23 日 gcc的编译过程及相关简介 (0)
- 2013 年 2 月 6 日 Linux glibc源码升级 (1)
- 2012 年 8 月 18 日 gcc到底帮我打开了哪些具体优化选项 (0)
- 2013 年 1 月 20 日 gcc中的weak和alias (0)
- 2013 年 3 月 16 日 如何对仅在指定调用路径下的函数断下断点 (0)

评论 (0) Trackbacks (0) 发表评论 Trackback

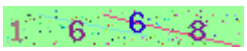
1. 本文目前尚无任何评论.

1. 本文目前尚无任何 trackbacks 和 pingbacks.

昵称 (必填)

电子邮箱 (我们会为您保密) (必填)

网址



Enter the abo

订阅评论

提交评论

NOTICE: You should type some Chinese word (like “你好”) in your comment to pass the spam-check, thanks for your patience!

gcc中的weak和alias nginx与正向代理

订阅

Sina

Recent Posts

- 一篇好文章：《为什么好产品会失败》
- VPP安装
- 伪装蜜罐
- 云舒创业公司默安科技的幻盾产品是做什么的？【标题党，哈哈哈】
- Linux上应用程序进行系统调用陷入内核后执行的cpu会发生变化吗？

Categories

- *nix应用编程
  - Android
  - GTK/QT编程
  - UbuntuKylin
  - 输入法
- *nix技术
  - Shell命令
  - 仿真虚拟
  - 内存管理
  - 内核技术
  - 多机集群
  - 多核优化
  - 应用程序
  - 文件系统
  - 跟踪调试
- Google开源
  - 浏览器
- 产品规划
- 佛文梵音
- 其它杂项
- 密码保护
- 数据结构
- 沙龙研讨
- 源码分析
  - lighttpd
  - nginx
- 生活点滴
  - MV&音乐
  - 人生感悟
  - 户外活动
  - 求职招聘
  - 漫画动漫
  - 计划总结
  - 读书后感
- 硬件设备
  - X86芯片
- 社区经验
- 系统管理
- 网络协议
  - http&spdy
  - TCP/IP
- 网络安全
  - 数据泄露
  - 网络攻防
- 英文翻译

Blogroll

- agentzh
- ant
- binutils
- chinaunix
- chromium
- cinderellasue
- cpu-world
- enterprise…
- geek
- glibc-man
- gmane
- gnu
- h-online
- hpca
- infoq
- kernel
- kernel-doc
- ligavin
- linuxforums
- linuxfoundation
- linuxfromscratch
- linuxjournal
- linuxquestions
- lwn.net
- marc
- net-security
- oftc
- open-open
- opera
- optimize
- osdev
- paste
- pdos
- redhat
- redhat
- scale
- solidot
- tektalk
- ubuntu-apps
- wangcong
- wikipedia
- xen
- yufeng

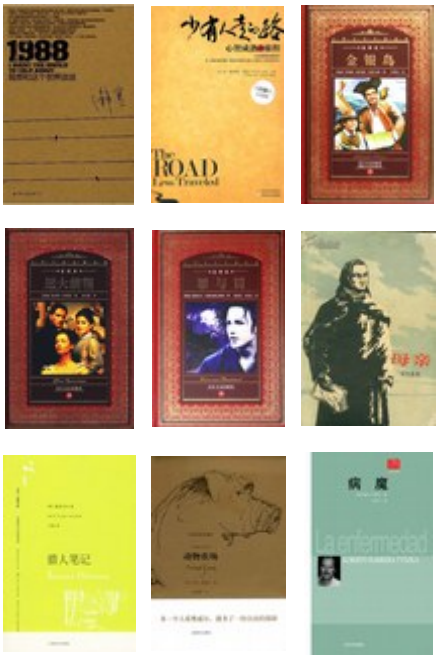
Archives

- 2016年十二月
- 2016年十一月
- 2016年十月
- 2016年六月

Recent Comments

- 爱国者 在 linux环境下protobuf-c的编译使用
  楼主，大sangfor在长沙有研发分部了，湖南的几乎都回去了，QQ
- pingback: Linux环境下protobuf-c的编译使用 | 神刀安全网
- swire 在 linux环境下protobuf-c的编译使用
  我去，多久没更新……我都快要忘记了。
- pingback: linux环境下protobuf-c的编译使用 - IT大道
- 123 在 Xfs文件系统磁盘布局之十三：普通文件数据结构（续）
  recs[1-63] = [startino,freecount,free] 1:[0,0,0xd6...
- 下一页 »

Dou Ban



我的豆瓣主页

Meta

- 注册
- 登录