巧用Graphviz和pvtrace等工具可視化C函數調用Oct 15, 2012

引子

在分析複雜的C/C++軟件時，如果有一個工具可以便捷的生成“函數調用關係圖”，不是一件很好的事嗎？如果你慶幸是一個Javaer或鍾愛基於IDE（如Eclipse）的軟件開發，應該會經常使用類似的工具。如果，你是*Nixer（*nix用戶）呢？其實,我們一樣有工具可用（地球村那麼多hacker，你遇見的問題，多半是別人早就碰到了並給出了相應的解決方案）。

除了 使用CodeViz、egypt和ncc，你可以嘗試一下本文介紹的方案（核心的處理方式都差不多）。

實現原理

依賴於gcc的hook機制，在函數的入口及出口打上“標籤”用於獲取“調用者”函數符號地址信息（保存到文件中），然後通過addr2line（pvtrace內部實現依賴於此工具），根據給定的“地址”從可執行文件中查出對應的“函數名”。最後，生成滿足graphviz組件dot語法的文件，用dot將其轉為圖形文件即可。

具體涉及的hook，如下：

1. 函數入口及出口hook函數原型

```
1. void __cyg_profile_func_enter( void *, void * )
2.     __attribute__ (( no_instrument_function ));
3.
4. void __cyg_profile_func_exit ( void *, void * )
5.     __attribute__ (( no_instrument_function ));
```

通過實現以上原型的實例函數，完成函數調用信息採集。

2. 在調用main函數之前及其退出之後，設置特殊處理操作的hook函數原型

```
1. void main_constructor( void )
2.     __attribute__ (( no_instrument_function , constructor ));
3.
4. void main_destructor ( void )
5.     __attribute__ (( no_instrument_function , destructor ));
```

通過實現以上原型的實例函數，生成及關閉用於保存函數調用關係信息的文件（trace.txt）。

具體的實現，可參考pvtrace源代碼中的`instrument.c`文件。

更多細節，請查閱用Graphviz可視化函數調用一文。

安裝pvtrace和Graphviz

1. 安裝pvtrace

```
1. $ mkdir -p ~/project1 && cd ~/project1
2. $ wget http : //www.mtjones.com/developerworks/pvtrace.zip
3. $ unzip pvtrace . zip - d pvtrace
4. $ cd pvtrace
5. $ make
6. $ sudo make install
7.
8. # 查看pvtrace相關文件
9. $ ls - 1 pvtrace
10.   instrument . c
11.   Makefile
12.   stack . c
13.   stack . h
14.   symbols . c
15.   symbols . h
```

```
16.   trace . c
```

2. 安装graphviz

```
1. $ sudo yum install graphviz
```

測試

在完成軟件安裝之後，編寫一個測試程序（test.c），並進行測試。具體流程，如下：

1. 編輯測試文件test.c

```
1. $ cd ~/project1
2. $ cat << EOF > test . c
3. #include <stdio.h>
4. #include <stdlib.h>
5.
6. void test1 ()
7. {
8.    printf ( "in test1.\n" );
9. }
10.
11.   void test2 ()
12.   {
13.      test1 ();
14.      printf ( "in test2.\n" );
15.   }
16.
17.   void test3 ()
18.   {
19.      test1 ();
20.      test2 ();
21.
22.      printf ( "in test3.\n" );
23.   }
24.
25.   int main ( int argc , char * argv [])
```

```
26.  {
27.      printf ( "Hello wolrd.\n" );
28.
29.      test1 ();
30.      test2 ();
31.      test3 ();
32.
33.      return 0 ;
34.  }
35.  EOF
```

2. 編譯測試程序

```
1. $ gcc -g -finstrument-functions test.c
   ./pvtrace/instrument.c -o test
2.
3. 注意： 必須有`-g -finstrument-functions` 選項，否則後續就採集不到
   信息了。
```

3. 執行程序，生成信息文件trace.txt

```
1. $ ./test
```

4. 通過pvtrace、可執行文件及trace.txt，生成信息文件graph.dot

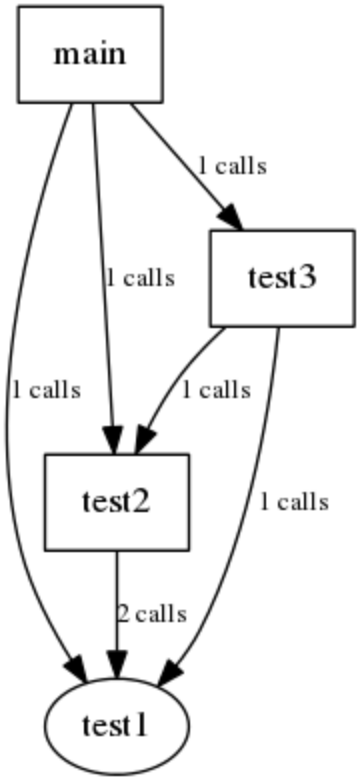
```
1. $ pvtrace test
```

5. 通過dot工具將graph.dot，轉為圖像文件graph.png

```
1. $ dot -Tpng graph.dot -o graph.png
```

6. 瀏覽生成的圖片

最終生成的圖形效果，如下：



為已有的項目生成函數調用圖

可以通過以下步驟為已有的項目生成函數調用圖：

1.將pvtrace源代碼中的`instrument.c`文件拷貝到項目中；

2. 增加對instrument.c文件的編譯

3.修改**編譯**選項：增加`-g -finstrument-functions`

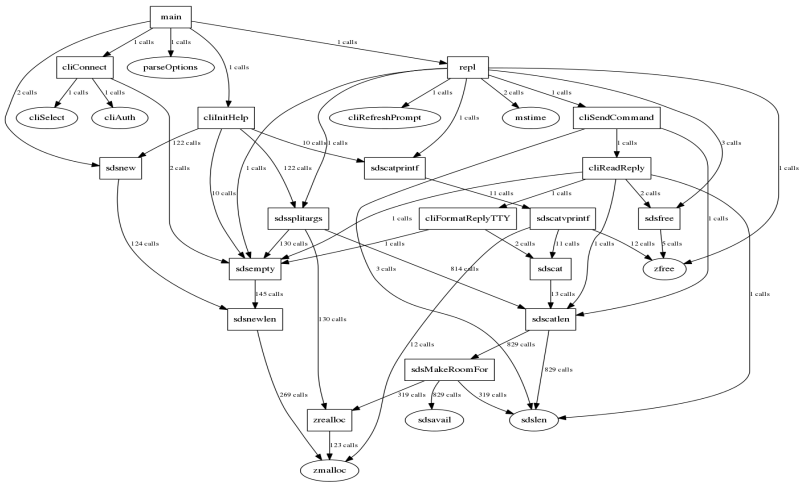
4.修改**連接**選項：將`instrument.o`連接到可執行文件中

5. 執行你的程序

6. 用pvtrace及“你的可執行文件”處理trace.txt

7. 用dot生成函數調用關係圖

以下是對redis-2.4.17版本的處理，然後生成redis-cli啟動及一個set操作對應的函數調用關係圖：



支持C++的擴展

目前pvtrace不支持C++代碼，如果有人希望改進，一種可行的改進思路，如下：

1. 修改instrument.c文件，支持C++環境的編譯；

2.通過`c++filt`工具處理解析到的函數名標籤，解析出實際的函數名：為了支持繼承、多態及函數重載等，C++編譯時對函數名進行了特殊處理；

3.採用合理的編碼方式，確保步驟2中生成的函數名滿足dot的語法（C++是用整個函數原型等信息來生成的函數簽名的，所以步驟2中用`c++filt`翻譯出來的是函數原型（包括名字空間等信息））；

4. 增加函數調用先後順序的標識。

同類工具

1. CodeViz : A CallGraph Visualiser;
2. egypt ;
3. ncc。

擴展閱讀

1. IBM developerworks上M. Tim Jones專欄及mtjones的主頁；
2. Graphviz各大組件（dot等）工具相關文檔；
3. GCC實用工具addr2line說明；
4. 陳碩的博文“用CodeViz繪製函數調用關係圖”；

參考文獻

1. M.Tim Jones的文章用Graphviz可視化函數調用。