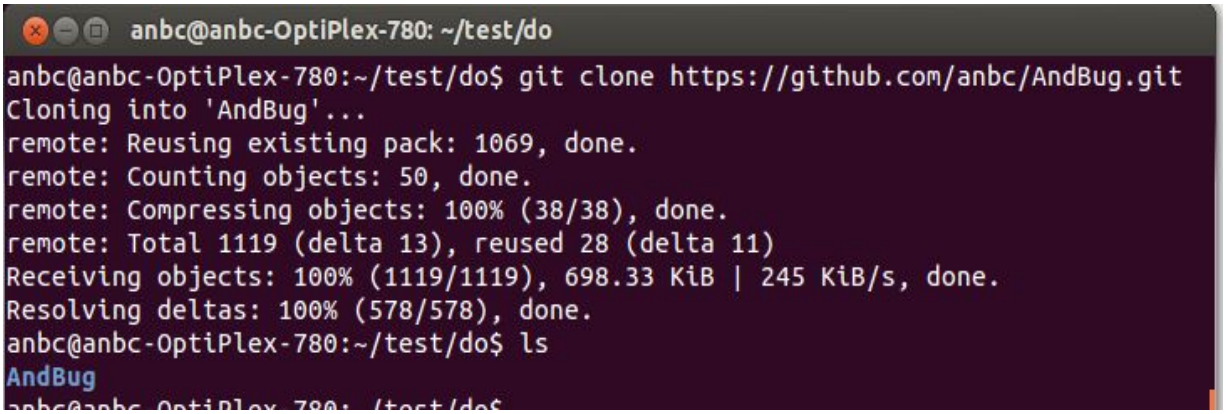
Android 动态逆向分析工具（一）

——Andbug 的基本操作

anbingchun@163.com

1、下载 andbug

修改版本：
git clone https://github.com/anbc/AndBug.git
作者原版本：
git clone https://github.com/swdunlop/AndBug



2、对 andbug 的部分模块进行编译

在 Andbug 文件夹中使用 make 命令进行编译

```
anbc@anbc-Aspire-M3660: ~/test/test/AndBug

anbc@anbc-Aspire-M3660:~/test/test/AndBug$ make
python setup.py build_ext -i
running build_ext
building 'andbug.jdwp' extension
creating build
creating build/temp.linux-i686-2.7
creating build/temp.linux-i686-2.7/lib
creating build/temp.linux-i686-2.7/lib/jdwp
gcc -pthread -fno-strict-aliasing -DNDEBUG -g -fwrapv -O2 -Wall -Wstrict-prototy
pes -fPIC -I/usr/include/python2.7 -c lib/jdwp/jdwp.c -o build/temp.linux-i686-2
.7/lib/jdwp/jdwp.o
lib/jdwp/jdwp.c: 在函数‘__pyx_f_4jdwp_10JdwpBuffer_ipack’中:
lib/jdwp/jdwp.c:1405:3: 警告: 隐式声明函数‘jdwp_expand’ [-Wimplicit-function-de
claration]
gcc -pthread -fno-strict-aliasing -DNDEBUG -g -fwrapv -O2 -Wall -Wstrict-prototy
pes -fPIC -I/usr/include/python2.7 -c lib/jdwp/wire.c -o build/temp.linux-i686-2
.7/lib/jdwp/wire.o
gcc -pthread -shared -Wl,-O1 -Wl,-Bsymbolic-functions -Wl,-Bsymbolic-functions -
Wl,-z,relro build/temp.linux-i686-2.7/lib/jdwp/jdwp.o build/temp.linux-i686-2.7/
lib/jdwp/wire.o -o /home/anbc/test/test/AndBug/lib/andbug/jdwp.so
PYTHONPATH=lib python2 setup.py test
running test
<<< 1 META
```

3、对文件路径进行设置

在 Andbug 文件夹中的 andbug 文件中，以下修改，修改 sys.path.append(“”)中的路径，比如作者将 Andbug 下载到了“/home/anbc/test/test/Andbug/”路径中，将 sys.path.append(“/home/anbc/test/test/Andbug/lib”)

```
anbc@anbc-Aspire-M3660: ~/test/test/AndBug
## more details.
##
## You should have received a copy of the GNU Lesser General Public License
## along with AndBug.  If not, see <http://www.gnu.org/licenses/>.

'this script executes command modules found in andbug.commands'
import os, os.path, sys, traceback, atexit
sys.path.append("/home/anbc/test/test/AndBug/lib")
#sys.path.append("/home/anbc/work_folder/andbug_work/andbug/lib")
def panic(why, exit=True, exc=False):
    sys.stderr.write("!! %s\n" % (why,))
    sys.stderr.flush()
    if exc:
        traceback.print_exc()
    if exit:
        sys.exit(-1)

def main(args):
    import andbug, andbug.cmd, andbug.command

    atexit.register(sys.stdout.write, '\x1B[0m\n')
    andbug.command.load_commands()
                                                          34,0-1        23%
```

4、启动虚拟机

emulator -avd Android_3

```
anbc@anbc-OptiPlex-780:~$ emulator -avd Android_3
Failed to load libGL.so
error libGL.so: cannot open shared object file: No such file or directory
Failed to load libGL.so
error libGL.so: cannot open shared object file: No such file or directory
emulator: emulator window was out of view and was recentered
```

5、查看当前的进行信息

```
anbc@anbc-OptiPlex-780: ~/test/do/AndBug

anbc@anbc-OptiPlex-780:~/test/do/AndBug$ adb shell ps
USER     PID   PPID  VSIZE  RSS     WCHAN    PC         NAME
root      1     0     296    208    c0098770 0000e840 S /init
root      2     0     0      0      c005048c 00000000 S kthreadd
root      3     2     0      0      c0042268 00000000 S ksoftirqd/0
root      4     2     0      0      c004ce30 00000000 S events/0
root      5     2     0      0      c004ce30 00000000 S khelper
root      6     2     0      0      c004ce30 00000000 S suspend
root      7     2     0      0      c004ce30 00000000 S kblockd/0
root      8     2     0      0      c004ce30 00000000 S cqueue
root      9     2     0      0      c016f7c4 00000000 S kseriod
root      10    2     0      0      c004ce30 00000000 S kmmcd
root      11    2     0      0      c006f36c 00000000 S pdflush
root      12    2     0      0      c006f36c 00000000 S pdflush
root      13    2     0      0      c007340c 00000000 S kswapd0
root      14    2     0      0      c004ce30 00000000 S aio/0
root      25    2     0      0      c016d0f8 00000000 S mtdblockd
root      26    2     0      0      c004ce30 00000000 S kstriped
root      27    2     0      0      c004ce30 00000000 S hid_compat
root      28    2     0      0      c004ce30 00000000 S rpciod/0
root      29    1     276    156    c0098770 0000e840 S /sbin/ueventd
system    30    1     836    344    c0195c08 40036fc0 S /system/bin/servicemanage
r
```

```
anbc@anbc-OptiPlex-780: ~/test/do/AndBug
u0_a23    388   37    185312 38200 ffffffff 40037ebc S com.android.systemui
u0_a24    414   37    177720 20848 ffffffff 40037ebc S com.android.inputmethod.l
atin
radio     436   37    196972 26048 ffffffff 40037ebc S com.android.phone
system    449   37    183816 19280 ffffffff 40037ebc S com.android.settings
u0_a16    461   37    175308 16352 ffffffff 40037ebc S com.android.location.fuse
d
u0_a4     505   37    188696 22220 ffffffff 40037ebc S android.process.acore
u0_a32    511   37    176204 17876 ffffffff 40037ebc S com.android.music
u0_a5     539   37    190680 40572 ffffffff 40037ebc S com.android.launcher
u0_a10    558   37    182772 21528 ffffffff 40037ebc S android.process.media
u0_a1     584   37    177700 17472 ffffffff 40037ebc S com.android.quicksearchbo
x
u0_a6     606   37    178692 19952 ffffffff 40037ebc S com.android.deskclock
u0_a4     626   37    183928 20952 ffffffff 40037ebc S com.android.contacts
u0_a3     642   37    180948 20460 ffffffff 40037ebc S com.android.mms
u0_a28    696   37    183548 18084 ffffffff 40037ebc S com.android.exchange
u0_a33    716   37    179868 19292 ffffffff 40037ebc S com.android.providers.cal
endar
u0_a26    734   37    186104 20036 ffffffff 40037ebc S com.android.calendar
u0_a13    920   37    237648 61084 ffffffff 40037ebc S com.android.browser
root      1229  46    752    428   c002a7a0 4003294c S /system/bin/sh
root      1231  1229  1092   432   00000000 40036d50 R ps
anbc@anbc-OptiPlex-780:~/test/do/AndBug$
```

可以看到浏览器应用的进程 id 是 920，包名是：com.android.browser

6、启动 andbug

./andbug shell -p com.android.browser

Andbug 有两种启动方式，一种是以进程 id 方式启动，一种是以包名方式启动
通过进程 id 或者包名指定对某个 apk 进行调试。

如图：以包名方式启动，进入 andbug 的 shell 中

```
anbc@anbc-OptiPlex-780: ~/test/do/AndBug
anbc@anbc-OptiPlex-780:~/test/do/AndBug$ ./andbug shell -p com.android.browser

## AndBug (C) 2011 Scott W. Dunlop <swdunlop@gmail.com>
>>
```

7、列举 apk 中的类信息

命令：classes
作用：列举出指定 apk 中使用的所有的 Class 信息，包括 apk 中自己实现的 Class 以及所调用的系统 class 信息

```
anbc-OptiPlex-780: ~/test/do/AndBug
    -- android.util.Pair
    -- android.graphics.Region$1
    -- org.apache.http.message.ParserCursor
    -- android.database.sqlite.SQLiteStatementInfo
    -- com.android.org.bouncycastle.asn1.DERObjectIdentifier
    -- org.apache.http.impl.cookie.RFC2965SpecFactory
    -- android.widget.CursorAdapter$ChangeObserver
    -- android.os.Bundle$1
    -- libcore.net.url.JarURLConnectionImpl$JarURLConnectionInputStream
    -- android.app.NotificationManager
    -- com.android.browser.Browser
    -- android.graphics.Matrix$1
    -- android.webkit.WebSettings$RenderPriority
    -- android.app.SharedPreferencesImpl$EditorImpl$1
    -- java.security.Permission
    -- android.content.pm.PathPermission$1
    -- android.net.Uri
    -- java.lang.ClassNotFoundException
    -- android.graphics.drawable.ClipDrawable
    -- android.app.SharedPreferencesImpl$EditorImpl$2
    -- org.apache.http.impl.client.AbstractHttpClient
    -- org.apache.harmony.security.x509.SubjectPublicKeyInfo
    -- org.apache.harmony.xnet.provider.jsse.OpenSSLSignature
    -- libcore.io.Base64
    -- android.location.ILocationManager$Stub$Proxy
    -- libcore.util.CollectionUtils
    -- org.apache.http.message.BasicRequestLine
    -- java.util.concurrent.ExecutorService
    -- com.android.org.bouncycastle.asn1.x509.PolicyInformation
    -- android.database.IContentObserver
    -- android.media.MediaCrypto
```

另外 classes 命令后边可以跟 Class 名称的信息，通过添加名称信息，可以列举出符合条件的 class 的信息。
如：classes app

如图，列举出所有类路径中包含了 app 字符串的类信息

```
>> classes app
## Loaded Classes
   -- com.android.internal.app.AlertController$ButtonHandler
   -- android.app.backup.BackupDataInput
   -- android.app.ActionBar
   -- android.app.DialogFragment
   -- android.app.ActivityThread$ContextCleanupInfo
   -- android.app.ActivityManagerNative$1
   -- android.app.ActivityThread$DropBoxReporter
   -- android.app.Dialog$1
   -- android.app.PendingIntent
   -- android.app.ApplicationPackageManager
   -- android.app.ResultInfo$1
   -- android.app.LoadedApk$ServiceDispatcher$InnerConnection
   -- com.android.internal.appwidget.IAppWidgetService$Stub$Proxy
   -- android.app.ContextImpl
   -- android.app.INotificationManager
   -- android.app.ActivityThread$EventLoggingReporter
   -- android.app.INotificationManager$Stub
   -- android.app.backup.BackupHelperDispatcher
   -- android.app.AppGlobals
   -- android.app.IActivityManager$ContentProviderHolder
   -- org.apache.harmony.security.utils.AlgNameMapper
   -- com.android.org.bouncycastle.jcajce.provider.symmetric.DES$Mappings
   -- com.android.internal.app.AlertController$1
   -- android.app.ListFragment
   -- android.database.CrossProcessCursorWrapper
   -- android.app.ResultInfo
   -- android.app.KeyguardManager
   -- android.app.IAlarmManager
   -- android.app.ActivityThread$ProviderRefCount
   -- android.app.ContextImpl$StaticServiceFetcher
```

8、列举指定类的方法信息

命令：methods android.app.NotificationManager
功能：列举出 android.app.NotificationManager 类中的所有方法信息

```
>> methods android.app.NotificationManager
## Methods Landroid/app/NotificationManager;
   -- android.app.NotificationManager.<clinit>()V
   -- android.app.NotificationManager.<init>(Landroid/content/Context;Landroid/o
      s/Handler;)V
   -- android.app.NotificationManager.from(Landroid/content/Context;)Landroid/ap
      p/NotificationManager;
   --
      android.app.NotificationManager.getService()Landroid/app/INotificationMana
      ger;
   -- android.app.NotificationManager.cancel(I)V
   -- android.app.NotificationManager.cancel(Ljava/lang/String;I)V
   -- android.app.NotificationManager.cancelAll()V
   -- android.app.NotificationManager.cancelAsUser(Ljava/lang/String;ILandroid/o
      s/UserHandle;)V
   -- android.app.NotificationManager.notify(ILandroid/app/Notification;)V
   -- android.app.NotificationManager.notify(Ljava/lang/String;ILandroid/app/Not
      ification;)V
   -- android.app.NotificationManager.notifyAsUser(Ljava/lang/String;ILandroid/a
      pp/Notification;Landroid/os/UserHandle;)V
>>
```

其中以 **android.app.NotificationManager.from(Landroid/content/Context;)Landroid/ap p/NotificationManager;** 为例。

android.app.NotificationManager——为类名
from——函数名
Landroid/content/Context;——from 函数的参数类型
Landroid/app/NotificationManager;——from 函数的返回值类型

9、断点操作

9.1 对类设置断点，所有设计该类的操作都会终止下来

break java.io.File

```
>> break java.io.File
## Setting Hooks
   -- Hooked <536870912> java.io.File <class 'andbug.vm.Class'>
```

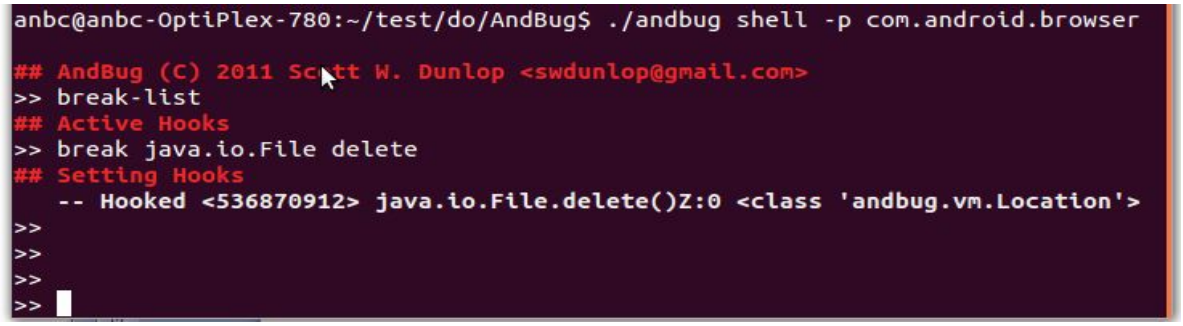
可以通过 break-list 命令查看断点设置情况

```
>> break-list
## Active Hooks
   -- Hook <536870912> java.io.File <class 'andbug.vm.Class'>
>>
```

9.2 对方法进行断点设置

通过下面命令对函数进行中断

>> break java.io.File delete



9.3 断点触发的情况

设置断点后，操作 apk 程序，一旦程序出发了某个设置的断点，整个程序就会挂起，并显示相关信息。通过输出的信息可以知道，apk 由于调用了 java.io.File.exists()Z:0 函数被中断下来。其他信息还包括整个调用 java.io.File.exists()Z:0 函数的堆栈信息。

```
anbc@anbc-OptiPlex-780: ~/test/do/AndBug
>> break java.io.File
## Setting Hooks
   -- Hooked <536870914> java.io.File <class 'andbug.vm.Class'>
>> break-list
## Active Hooks
   -- Hook <536870912> java.io.File.delete()Z:0 <class 'andbug.vm.Location'>
   -- Hook <536870913> java.net.Socket <class 'andbug.vm.Class'>
   -- Hook <536870914> java.io.File <class 'andbug.vm.Class'>
>> ## Breakpoint hit in thread <1> main    (running suspended), process suspende
d.
   -- java.io.File.exists()Z:0
   -- android.app.ContextImpl.getCacheDir()Ljava/io/File;:22
   -- android.content.ContextWrapper.getCacheDir()Ljava/io/File;:2
   --
      com.android.browser.CrashRecoveryHandler.writeState(Landroid/os/Bundle;)V:
      13
   -- com.android.browser.Controller.onSaveInstanceState(Landroid/os/Bundle;)V:6
   --
      com.android.browser.BrowserActivity.onSaveInstanceState(Landroid/os/Bundle
      ;)V:2
   -- android.app.Activity.performSaveInstanceState(Landroid/os/Bundle;)V:0
   -- android.app.Instrumentation.callActivityOnSaveInstanceState(Landroid/app/A
      ctivity;Landroid/os/Bundle;)V:0
   -- android.app.ActivityThread.performStopActivityInner(Landroid/app/ActivityT
      hread$ActivityClientRecord;Landroid/app/ActivityThread$StopInfo;ZZ)V:98
   -- android.app.ActivityThread.handleStopActivity(Landroid/os/IBinder;ZI)V:22
   -- android.app.ActivityThread.access$900(Landroid/app/ActivityThread;Landroid
      /os/IBinder;ZI)V:0
   -- android.app.ActivityThread$H.handleMessage(Landroid/os/Message;)V:148
   -- android.os.Handler.dispatchMessage(Landroid/os/Message;)V:20
   -- android.os.Looper.loop()V:84
   -- android.app.ActivityThread.main([Ljava/lang/String;)V:48
   -- java.lang.reflect.Method.invokeNative(Ljava/lang/Object;[Ljava/lang/Object
      ;Ljava/lang/Class;[Ljava/lang/Class;Ljava/lang/Class;IZ)Ljava/lang/Object;
      <native>
   -- java.lang.reflect.Method.invoke(Ljava/lang/Object;[Ljava/lang/Object;)Ljav
      a/lang/Object;:17
   -- com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run()V:11
   -- com.android.internal.os.ZygoteInit.main([Ljava/lang/String;)V:66
   -- dalvik.system.NativeStart.main([Ljava/lang/String;)V <native>
>>
```

9.4 删除已设置断点

通过 break-remove 536870916
将之前设置的中断删除掉

```
>> break java.io.File
## Setting Hooks
   -- Hooked <536870916> java.io.File <class 'andbug.vm.Class'>
>> break-list
## Active Hooks
   -- Hook <536870916> java.io.File <class 'andbug.vm.Class'>
>> break-remove 536870916
## Hook <536870916> removed
>> break-list
## Active Hooks
>>
```

另外：break-remove all 表示删除所有当前设置的断点。

10、恢复运行

命令：resume
功能：触发断点整个进程暂停后，通过调用 resume 命令，继续运行该 apk
注：由于代码中 bug 的原因，需要连续两次调用 resume 命令才能恢复 apk 的运行

```
>>
>> resume
## Process Resumed
>> resume
## Process Resumed
>> ## Breakpoint hit in thread <1> main    (running suspended), process suspende
d.
   -- java.io.File.<init>(Ljava/io/File;Ljava/lang/String;)V:0
   --
      com.android.browser.CrashRecoveryHandler.writeState(Landroid/os/Bundle;)V:
      19
   -- com.android.browser.Controller.onSaveInstanceState(Landroid/os/Bundle;)V:6
   --
      com.android.browser.BrowserActivity.onSaveInstanceState(Landroid/os/Bundle
      ;)V:2
   -- android.app.Activity.performSaveInstanceState(Landroid/os/Bundle;)V:0
   -- android.app.Instrumentation.callActivityOnSaveInstanceState(Landroid/app/A
      ctivity;Landroid/os/Bundle;)V:0
   -- android.app.ActivityThread.performStopActivityInner(Landroid/app/ActivityT
      hread$ActivityClientRecord;Landroid/app/ActivityThread$StopInfo;ZZ)V:98
   -- android.app.ActivityThread.handleStopActivity(Landroid/os/IBinder;ZI)V:22
   -- android.app.ActivityThread.access$900(Landroid/app/ActivityThread;Landroid
      /os/IBinder;ZI)V:0
   -- android.app.ActivityThread$H.handleMessage(Landroid/os/Message;)V:148
   -- android.os.Handler.dispatchMessage(Landroid/os/Message;)V:20
   -- android.os.Looper.loop()V:84
   -- android.app.ActivityThread.main([Ljava/lang/String;)V:48
   -- java.lang.reflect.Method.invokeNative(Ljava/lang/Object;[Ljava/lang/Object
      ;Ljava/lang/Class;[Ljava/lang/Class;Ljava/lang/Class;IZ)Ljava/lang/Object;
      <native>
   -- java.lang.reflect.Method.invoke(Ljava/lang/Object;[Ljava/lang/Object;)Ljav
      a/lang/Object;:17
   -- com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run()V:11
   -- com.android.internal.os.ZygoteInit.main([Ljava/lang/String;)V:66
   -- dalvik.system.NativeStart.main([Ljava/lang/String;)V <native>
```

如上图,执行两次 resume 命令后，恢复 apk 的运行，由于对整个 java.io.File 类设置了断点，应用又中断在 java.io.File.<init>上。

11、进程暂停命令

命令：suspend
功能：暂停当前 apk 进程

由于还没有与该命令配合使用的命令，实际使用中没有太大作用

```
anbc@anbc-OptiPlex-780:~/test/do/AndBug$ ./andbug shell -p com.android.browser

## AndBug (C) 2011 Scott W. Dunlop <swdunlop@gmail.com>
>> suspend
## Process Suspended
>>
```

12、帮助命令 help

```
anbc@anbc-OptiPlex-780: ~/test/do/AndBug
>> help
## AndBug (C) 2011 Scott W. Dunlop <swdunlop@gmail.com>
   The AndBug shell is a simple interactive console shell that reduces typing
   and overhead involved in setting up a debugging session.  Commands entered at
   the prompt will be evaluated using the current device and process as a
   context.  Where possible, AndBug uses readline; if your Python install lacks
   readline, this shell will be more difficult to use due to the poor console
   I/O functionality in vanilla Python.  (The "rlwrap"  utility may help.)

   AndBug is NOT intended for a piracy tool, or other illegal purposes, but  as
   a tool for researchers and developers to gain insight into the
   implementation of Android applications.  Use of AndBug is at your own risk,
   like most open source tools, and no guarantee of fitness or safety is made or
   implied.
## Commands:
   -- break | b <class> [<method>] [show/lineNo]
      set breakpoint
   -- break-list
      list active breakpoints/hooks
   -- break-remove <eid/all>
      remove hook/breakpoint
   -- class-trace | ct | ctrace <class-path>
      reports calls to dalvik methods associated with a class
   -- classes [<partial class name>]
      lists loaded classes. if no partial class name supplied, list all classes.
   -- dump <class-path> [<method-query>]
      dumps methods using original sources or apktool sources
   -- exit
      terminates andbug with prejudice
   -- help [<command>]
      information about how to use andbug
   -- inspect <object-id>
      inspect an object
   -- method-trace | mt | mtrace <method>
      reports calls to specific dalvik method
   -- methods <class-path> [<method-query>]
      lists the methods of a class
   -- navi [allowRemote=<False or anychar>] [port=<8080>]
      starts an http server for browsing process state
   -- resume [<name>]
      resumes threads in the process
```

13、class-trace 类跟踪命令

命令：class-trace java.io.File
功能：对 java.io.File 类的调用情况进行跟踪
可以看到“thread <15> BackgroundHandler”线程，调用了“java.io.File.renameTo”，调用的参数是：

-- this=Ljava/io/File; <830018592096>

-- newPath=Ljava/io/File; <830016257352>

并且可以看到整个的堆栈调用的情况。

```
anbc@anbc-OptiPlex-780: ~/test/do/AndBug
      -- this=Lcom/android/browser/CrashRecoveryHandler; <830016122944>
      -- state=Landroid/os/Bundle; <830018577184>
      -- stateJournal=Ljava/io/File; <830018592096>
      -- fout=Ljava/io/FileOutputStream; <830018856152>
   -- com.android.browser.CrashRecoveryHandler$1.handleMessage(Landroid/os/Messa
      ge;)V:12
      -- this=Lcom/android/browser/CrashRecoveryHandler$1; <830016131072>
      -- msg=Landroid/os/Message; <830016183656>
      -- saveState=Landroid/os/Bundle; <830018577184>
   -- android.os.Handler.dispatchMessage(Landroid/os/Message;)V:20
      -- this=Lcom/android/browser/CrashRecoveryHandler$1; <830016131072>
      -- msg=Landroid/os/Message; <830016183656>
   -- android.os.Looper.loop()V:84
      -- me=Landroid/os/Looper; <830016061000>
      -- msg=Landroid/os/Message; <830016183656>
      -- ident=43005507536826
      -- logging=None
      -- queue=Landroid/os/MessageQueue; <830016061032>
   -- android.os.HandlerThread.run()V:28
      -- this=Landroid/os/HandlerThread; <830016009832>
## trace thread <15> BackgroundHandler  (running suspended)
   -- java.io.File.renameTo(Ljava/io/File;)Z:0
      -- this=Ljava/io/File; <830018592096>
      -- newPath=Ljava/io/File; <830016257352>
   --
      com.android.browser.CrashRecoveryHandler.writeState(Landroid/os/Bundle;)V:
      50
      -- stateFile=Ljava/io/File; <830016257352>
      -- this=Lcom/android/browser/CrashRecoveryHandler; <830016122944>
      -- state=Landroid/os/Bundle; <830018577184>
      -- stateJournal=Ljava/io/File; <830018592096>
      -- fout=Ljava/io/FileOutputStream; <830018856152>
   -- com.android.browser.CrashRecoveryHandler$1.handleMessage(Landroid/os/Messa
      ge;)V:12
      -- this=Lcom/android/browser/CrashRecoveryHandler$1; <830016131072>
      -- msg=Landroid/os/Message; <830016183656>
      -- saveState=Landroid/os/Bundle; <830018577184>
   -- android.os.Handler.dispatchMessage(Landroid/os/Message;)V:20
      -- this=Lcom/android/browser/CrashRecoveryHandler$1; <830016131072>
      -- msg=Landroid/os/Message; <830016183656>
   -- android.os.Looper.loop()V:84
```

取消跟踪也可以使用 break-remove 命令实现。

14、method-trace 方法跟踪命令

命令：method-trace java.io.File renameTo

功能：对 java.io.File renameTo 函数进行跟踪，跟踪与直接设置断点的差别是，使用跟踪不中断目标进程的执行，只对相应函数的调用信息进行输出。

```
anbc@anbc-OptiPlex-780:~/test/do/AndBug$ ./andbug shell -p com.android.browser

## AndBug (C) 2011 Scott W. Dunlop <swdunlop@gmail.com>
>> method-trace java.io.File renameTo
!! too many arguments for command "method-trace."
>> method-trace java.io.File.renameTo
## Setting Hooks
   -- Hooked java.io.File.renameTo(Ljava/io/File;)Z:0
>> break-list
## Active Hooks
   -- Hook <536870918> java.io.File.renameTo(Ljava/io/File;)Z:0 <class
      'andbug.vm.Location'>
>>
```

获取函数调用的跟踪信息

```
!! too many arguments for command "method-trace."
>> method-trace java.io.File.renameTo
## Setting Hooks
   -- Hooked java.io.File.renameTo(Ljava/io/File;)Z:0
>> break-list
## Active Hooks
   -- Hook <536870918> java.io.File.renameTo(Ljava/io/File;)Z:0 <class
      'andbug.vm.Location'>
>> ## trace thread <15> BackgroundHandler  (running suspended)
   -- java.io.File.renameTo(Ljava/io/File;)Z:0
      -- this=Ljava/io/File; <830017306728>
      -- newPath=Ljava/io/File; <830017307536>
## trace thread <15> BackgroundHandler  (running suspended)
   -- java.io.File.renameTo(Ljava/io/File;)Z:0
      -- this=Ljava/io/File; <830018653256>
      -- newPath=Ljava/io/File; <830018741584>
```

15、列举当前线程信息

命令：threads

```
>> threads
## thread <1> main       (running suspended)
## thread <2> GC         (waiting suspended)
## thread <3> Signal Catcher    (waiting suspended)
## thread <5> Compiler  (waiting suspended)
## thread <6> ReferenceQueueDaemon      (waiting suspended)
## thread <7> FinalizerDaemon   (waiting suspended)
## thread <8> FinalizerWatchdogDaemon   (waiting suspended)
## thread <9> Binder_1  (running suspended)
## thread <10> Binder_2 (running suspended)
## thread <11> AsyncTask #1     (waiting suspended)
## thread <12> AsyncTask #2     (waiting suspended)
>>
```

16、对线程进行跟踪

命令：thread-trace

将 main 线程设置为跟踪县城，所有 main 县城相关的调用都会被跟踪下来，如图所示：

```
anbc@anbc-OptiPlex-780: ~/test/do/AndBug
      -- this=Ljava/lang/reflect/Method; <830015921152>
      -- args=([],)
      -- receiver=None
   -- com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run()V:11
      -- this=Lcom/android/internal/os/ZygoteInit$MethodAndArgsCaller;
         <830015921304>
   -- com.android.internal.os.ZygoteInit.main([Ljava/lang/String;)V:66
      -- caller=Lcom/android/internal/os/ZygoteInit$MethodAndArgsCaller;
         <830015921304>
      -- argv=(#'com.android.internal.os.ZygoteInit', #'start-system-server')
   -- dalvik.system.NativeStart.main([Ljava/lang/String;)V <native>
## trace thread <1> main        (running suspended)
   -- android.os.MessageQueue.nativeWake(I)V <native>
   -- android.os.MessageQueue.removeSyncBarrier(I)V:46
      -- this=Landroid/os/MessageQueue; <830015924616>
      -- p=Landroid/os/Message; <830016027880>
      -- token=3211
      -- prev=None
   -- android.os.Looper.removeSyncBarrier(I)V:2
      -- this=Landroid/os/Looper; <830015924584>
      -- token=3211
   -- android.view.ViewRootImpl.doTraversal()V:17
      -- this=Landroid/view/ViewRootImpl; <830015905712>
   -- android.view.ViewRootImpl$TraversalRunnable.run()V:2
      -- this=Landroid/view/ViewRootImpl$TraversalRunnable; <830015958488>
   -- android.view.Choreographer$CallbackRecord.run(J)V:20
      -- this=Landroid/view/Choreographer$CallbackRecord; <830016027760>
      -- frameTimeNanos=9943394854190
   -- android.view.Choreographer.doCallbacks(IJ)V:26
      -- this=Landroid/view/Choreographer; <830015924104>
      -- callbackType=2
      -- frameTimeNanos=9943394854190
   -- android.view.Choreographer.doFrame(JI)V:103
      -- jitterNanos=2208395221
      -- this=Landroid/view/Choreographer; <830015924104>
      -- frame=29401
      -- startNanos=9943395243661
      -- frameTimeNanos=9943394854190
   -- android.view.Choreographer$FrameDisplayEventReceiver.run()V:9
```

会列出函数调用的情况，参数，以及堆栈情况。

17、显示指定类中的静态变量的信息

命令：statics com.android.internal.view.menu.MenuBuilder

```
>> statics com.android.internal.view.menu.MenuBuilder
## Static Fields, com.android.internal.view.menu.MenuBuilder

-- PRESENTER_KEY = android:menu:presenters
-- TAG = MenuBuilder
-- sCategoryToOrder = (1, 4, 5, 3, 2, 0)
-- ACTION_VIEW_STATES_KEY = android:menu:actionviewstates
-- EXPANDED_ACTION_VIEW_ID = android:menu:expandedactionview
>>
```

18、查看对象信息

通过 class-trace 命令可以跟踪到目标函数中对象的 Id 信息，

```
## trace thread <1> main        (running suspended)
   -- java.io.File.getPath()Ljava/lang/String;:0
      -- this=Ljava/io/File; <830015928824>
   -- java.io.File.<init>(Ljava/io/File;Ljava/lang/String;)V:7
      -- this=Ljava/io/File; <830019498488>
      -- name=app_databases
      -- dir=Ljava/io/File; <830015928824>
   -- android.app.ContextImpl.makeFilename(Ljava/io/File;Ljava/lang/String;)Ljav
      a/io/File;:10
      -- this=Landroid/app/ContextImpl; <830015939944>
      -- base=Ljava/io/File; <830015928824>
      -- name=app_databases
   -- android.app.ContextImpl.getDir(Ljava/lang/String;I)Ljava/io/File;:23
      -- this=Landroid/app/ContextImpl; <830015939944>
      -- name=app_databases
      -- mode=0
   -- android.content.ContextWrapper.getDir(Ljava/lang/String;I)Ljava/io/File;:2
      -- this=Lcom/android/browser/Browser; <830015940904>
      -- name=databases
      -- mode=0
   -- com.android.browser.BrowserSettings.syncStaticSettings(Landroid/webkit/Web
      SettingsClassic;)V:72
      -- this=Lcom/android/browser/BrowserSettings; <830016000912>
```

通过对象的 Id 使用 inspect 命令，可以查处该队形的详细信息。

```
>>
>> inspect 830016120776
## object <830016120776> Lcom/android/browser/TabControl; in thread <1> main
   (running suspended)
   -- mOnThumbnailUpdatedListener=None <NoneType>
   -- mController=Lcom/android/browser/Controller; <830016119696> <Object>
   -- mTabs=Ljava/util/ArrayList; <830016120816> <Object>
   -- mMaxTabs=16 <int>
   -- mCurrentTab=0 <int>
   -- mTabQueue=Ljava/util/ArrayList; <830016120928> <Object>
>> inspect 830015928824
## object <830015928824> Ljava/io/File; in thread <1> main      (running
   suspended)
   -- path=/data/data/com.android.browser <String>
>>
```

由于在 break 命令设置断点后，触发断点时反馈的信息，没有包含 Object Id 的信息，导致 inspect 命令用起来不是很方便。

19、源码关联命令

命令：source 与源代码关联起来，可以是 smali 代码。
命令：dump 展示指定方法的代码。

20、Web 输出命令

命令：navi
注：为了支持 navi 命令，需要安装 bottle 库。

需要 bottle 库。应该是一个 web 展示的页面
需要安装 bottle 库，来实现。

21、退出命令

命令：exit

```
>> exit
anbc@anbc-OptiPlex-780:~/test/do/AndBug$
```