gdb & pwndbg

vmmap:查看内存布局
n:next=f8，步过
s:step=f7，步入
c:continue,继续
b*0xaddr：breakpoint,在0xaddr处下断点
finish:执行到当前函数返回
x/[n]gx [addr] :查看64bits内存
x/[n]wx [addr] :查看32bits内存
x/[n]i [addr]：查看汇编指令

p [变量/函数名] checkpoint 快照，能实现gdb回退
set $reg = value，设置寄存器的值
set {int}addr = value,设置内存值
$reg可以索引寄存器
catch syscall exit_group/exit ...在exit下断点等。。

exec
调用exec时断点被踩到。
syscall
syscall [name | number] ...
通过系统函数的名称和系统号，来设置捕获点，当所设定的系统调用时，断点被踩到。
因为经常在linux用c语言，所以主要用到的event是最后四个，其他的没有仔细研究。
例如：
catch syscall open
catch syscall 5
这两个捕获断点一样。

16

```
(gdb) catch syscall 60
Catchpoint 3 (syscall 'exit' [60])
(gdb) catch syscall 231
Catchpoint 4 (syscall 'exit_group' [231])
```

set args 命令行参数

源码调试

gcc -g -o a a.c，其中-g参数可以添加源代码，gdb调试时，可list查看，并按源代码调试。
条件断点 b 10 if i>100 指在.c的line 10断下，如果i>100。

ignore 忽略断点n次

二，ignore

如果我们不是想根据某一条件表达式来停止，而是想断点自动忽略前面多少次的停止，从某一次开始才停止，这时ignore就很有用了。

ignore break_number count

上面的命令行表示break_number所指定的断点号将被忽略count次。

如：

ignore 1 100，表示忽略断点1的前100次停止

多进程调试

set follow-fork-mode parent/child
可以使在fork之后gdb跟从child还是parent
同时：

```
set detach-on-fork [on|off]
```

on: 断开调试follow-fork-mode指定的进程。

off: gdb将控制父进程和子进程。follow-fork-mode指定的进程将被调试，另一个进程置于暂停（suspended）状态。