# Valgrind使用

## 简介

Valgrind是一套Linux下，开放源代码的仿真调试工具的集合。Valgrind由内核（core）以及基于内核的其他调试工具组成。内核类似于一个框架（framework），它模拟了一个CPU环境，并提供服务给其他工具；而其他工具则类似于插件 (plug-in)，利用内核提供的服务完成各种特定的内存调试任务。

## 工具

**Valgrind**具体包括如下工具：

**Memcheck**，这是valgrind应用最广泛的工具，一个重量级的内存检查器，能够发现开发中绝大多数内存错误使用情况，比如：使用未初始化的内存，使用已经释放了的内存，内存访问越界等。这也是本文将重点介绍的部分。

**Callgrind**，它主要用来检查程序中函数调用过程中出现的问题。

**Cachegrind**，它主要用来检查程序中缓存使用出现的问题。

**Helgrind**，它主要用来检查多线程程序中出现的竞争问题。

**Massif**，它主要用来检查程序中堆栈使用中出现的问题。

**Extension**，可以利用core提供的功能，自己编写特定的内存调试工具。

Linux下内存空间布置：

由exec从源程序文件中读入

低地址

高地址

代码段

初始化的数据段(data)

未初始化的数据段(bss)

堆（Heap）

栈（Stack）

命令行参数和环境变量

由exec函数初始化为0

## memcheck

Valid-Value 表：

对于进程的整个地址空间中的每一个字节(byte)，都有与之对应的 8 个 bits；对于 CPU 的每个寄存器，也有一个与之对应的 bit 向量。这些 bits 负责记录该字节或者寄存器值是否具有有效的、已初始化的值。

Valid-Address 表

对于进程整个地址空间中的每一个字节(byte)，还有与之对应的 1 个 bit，负责记录该地址是否能够被读写。

检测原理：

当要读写内存中某个字节时，首先检查这个字节对应的 A bit。如果该A bit显示该位置是无效位置，memcheck 则报告读写错误。

内核（core）类似于一个虚拟的 CPU 环境，这样当内存中的某个字节被加载到真实的 CPU 中时，该字节对应的 V bit 也被加载到虚拟的 CPU 环境中。一旦寄存器中的值，被用来产生内存地址，或者该值能够影响程序输出，则 memcheck 会检查对应的V bits，如果该值尚未初始化，则会报告使用未初始化内存错误。

Valgrind 使用

用法: valgrind [options] prog-and-args [options]: 常用选项，适用于所有Valgrind工具

-tool=<name> 最常用的选项。运行 valgrind中名为toolname的工具。默认memcheck。

h –help 显示帮助信息。

-version 显示valgrind内核的版本，每个工具都有各自的版本。

q –quiet 安静地运行，只打印错误信息。

v –verbose 更详细的信息, 增加错误数统计。

-trace-children=no|yes 跟踪子线程? [no]

-track-fds=no|yes 跟踪打开的文件描述？[no]

-time-stamp=no|yes 增加时间戳到LOG信息? [no]

-log-fd=<number> 输出LOG到描述符文件 [2=stderr]

-log-file=<file> 将输出的信息写入到filename.PID的文件里，PID是运行程序的进行ID

-log-file-exactly=<file> 输出LOG信息到 file

-log-file-qualifier=<VAR> 取得环境变量的值来做为输出信息的文件名。 [none]

-log-socket=ipaddr:port 输出LOG到socket ，ipaddr:port

LOG信息输出

-xml=yes 将信息以xml格式输出，只有memcheck可用

-num-callers=<number> show <number> callers in stack traces [12]

-error-limit=no|yes 如果太多错误，则停止显示新错误? [yes]

-error-exitcode=<number> 如果发现错误则返回错误代码 [0=disable]

-db-attach=no|yes 当出现错误，valgrind会自动启动调试器gdb。[no]

-db-command=<command> 启动调试器的命令行选项[gdb -nw %f %p]

适用于Memcheck工具的相关选项：

-leak-check=no|summary|full 要求对leak给出详细信息? [summary]

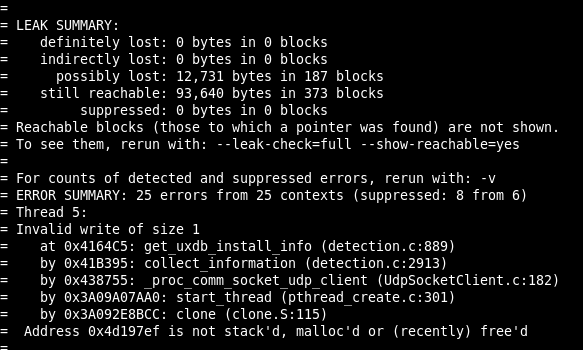
-leak-resolution=low|med|high how much bt merging in leak check [low]

-show-reachable=no|yes show reachable blocks in leak check? [no]

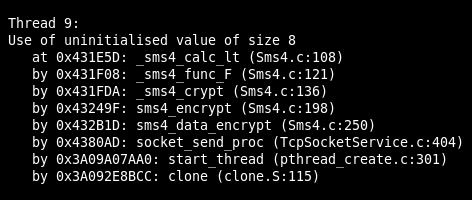
## 检测结果

检测命令：valgrind --log-file=valgrind.log --tool=memcheck --leak-check=full --show-reachable=no --workaround-gcc296-bugs=yes ./UxdbAgent

非法写操作：



使用为初始化的变量：



内存泄漏：

