何时放入常规bin？

/*
Place the chunk in unsorted chunk list. Chunks are not placed into regular bins until after they have been given one chance to be used in malloc. */

- 释放时先放到无序bin
- 下次执行malloc时，放入所属的常规bin

```
/* place chunk in bin */

if (in_smallbin_range(size)) {
  victim_index = smallbin_index(size);
  bck = bin_at(av, victim_index);
  fwd = bck->fd;
}
else {
  victim_index = largebin_index(size);
  bck = bin_at(av, victim_index);
  fwd = bck->fd;

  if (fwd != bck) {
    /* if smaller than smallest, place first */
assert((bck->bk->size & NON_MAIN_ARENA) == 0);
    if ((unsigned long)(size) < (unsigned long)(bck->bk->size)) {
      fwd = bck;
      bck = bck->bk;
    }
    else if ((unsigned long)(size) >=
             (unsigned long)(FIRST_SORTED_BIN_SIZE)) {

      /* maintain large bins in sorted order */
      size |= PREV_INUSE; /* Or with inuse bit to speed comparisons */
  assert((fwd->size & NON_MAIN_ARENA) == 0);
      while ((unsigned long)(size) < (unsigned long)(fwd->size)) {
        fwd = fwd->fd;
    assert((fwd->size & NON_MAIN_ARENA) == 0);
  }
      bck = fwd->bk;
    }
  }
}

mark_bin(av, victim_index);
victim->bk = bck;
victim->fd = fwd;
fwd->bk = victim;
bck->fd = victim;
}
```

0804c000-0806d000 rw-p 00000000 00:00 0 [heap]
0806d000-08090000 rw-p 00000000 00:00 0 [heap]
08090000-080b2000 rw-p 00000000 00:00 0 [heap]

#0 __GI___sbrk (increment=-4096) at sbrk.c:35
#1 0xb79c4f4f in __GI___default_morecore (increment=-4096) at morecore.c:49
#2 0xb79beedd in sYSTRIm (pad=<optimized out>, av=<optimized out>) at malloc.c:2810
#3 0xb79bfcb5 in _int_free (av=0xb7af2440, p=0x808fff8, have_lock=1) at malloc.c:4196

0804c000-0806d000 rw-p 00000000 00:00 0 [heap]
0806d000-08090000 rw-p 00000000 00:00 0 [heap]
08090000-080b1000 rw-p 00000000 00:00 0 [heap]

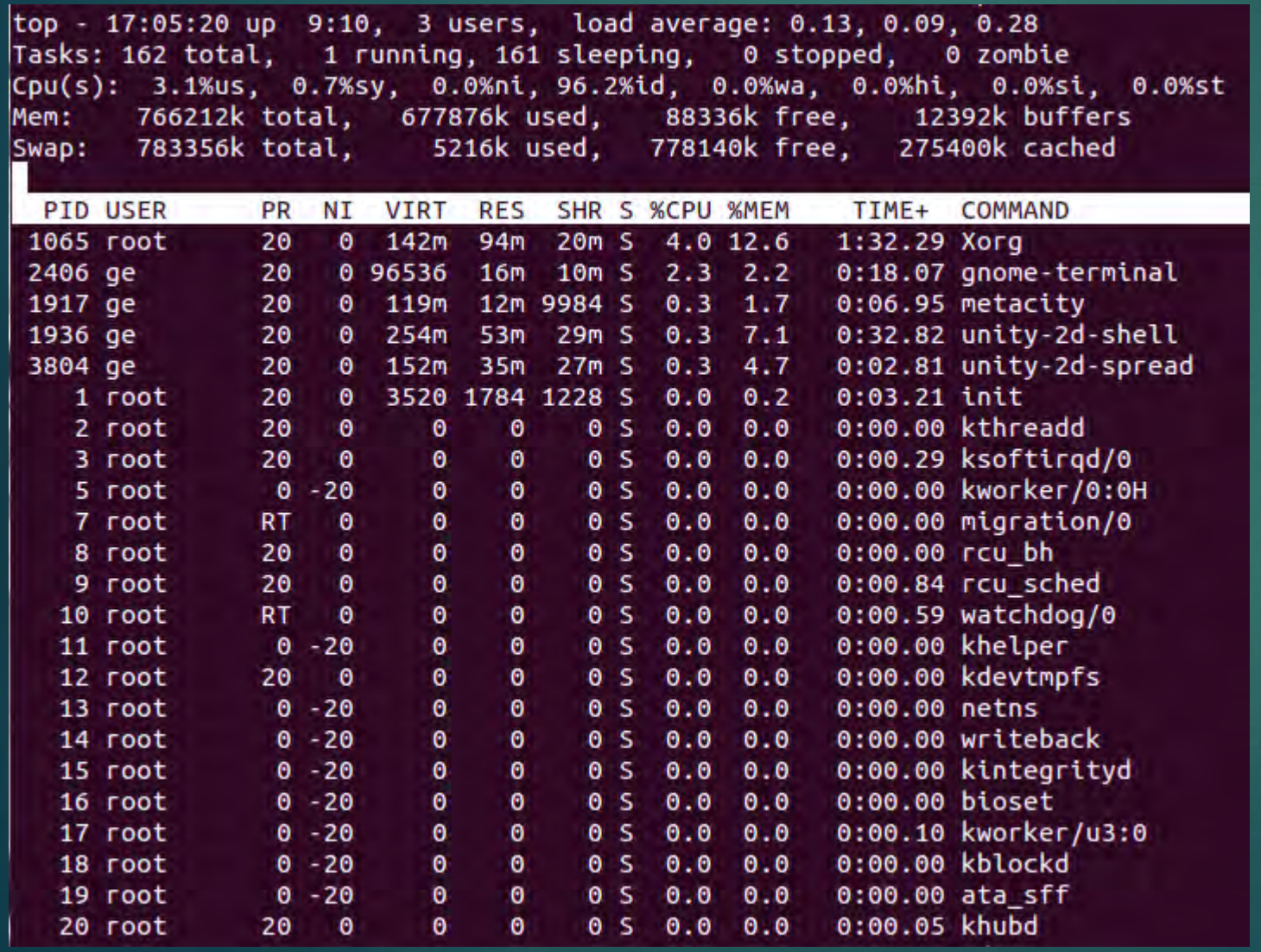

概览
Arena
批发
分配和释放
Valgrind

```
ge@gewubox:~/work/heap$ cat /proc/meminfo
MemTotal:         766212 kB
MemFree:           90056 kB
Buffers:           12236 kB
Cached:           275296 kB
SwapCached:          348 kB
Active:           306888 kB
Inactive:         304720 kB
Active(anon):     156212 kB
Inactive(anon):   182932 kB
Active(file):     150676 kB
Inactive(file):   121788 kB
Unevictable:           0 kB
Mlocked:               0 kB
HighTotal:             0 kB
HighFree:              0 kB
LowTotal:         766212 kB
LowFree:           90056 kB
SwapTotal:        783356 kB
SwapFree:         778140 kB
Dirty:                28 kB
Writeback:             0 kB
AnonPages:        323768 kB
Mapped:            93660 kB
Shmem:             15068 kB
Slab:              44596 kB
SReclaimable:      32576 kB
SUnreclaim:        12020 kB
KernelStack:        3008 kB
PageTables:         7428 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:     1166460 kB
Committed_AS:    2400960 kB
VmallocTotal:     249912 kB
VmallocUsed:       20264 kB
VmallocChunk:     221168 kB
HardwareCorrupted:     0 kB
AnonHugePages:         0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
DirectMap4k:       34752 kB
DirectMap2M:      751616 kB
```

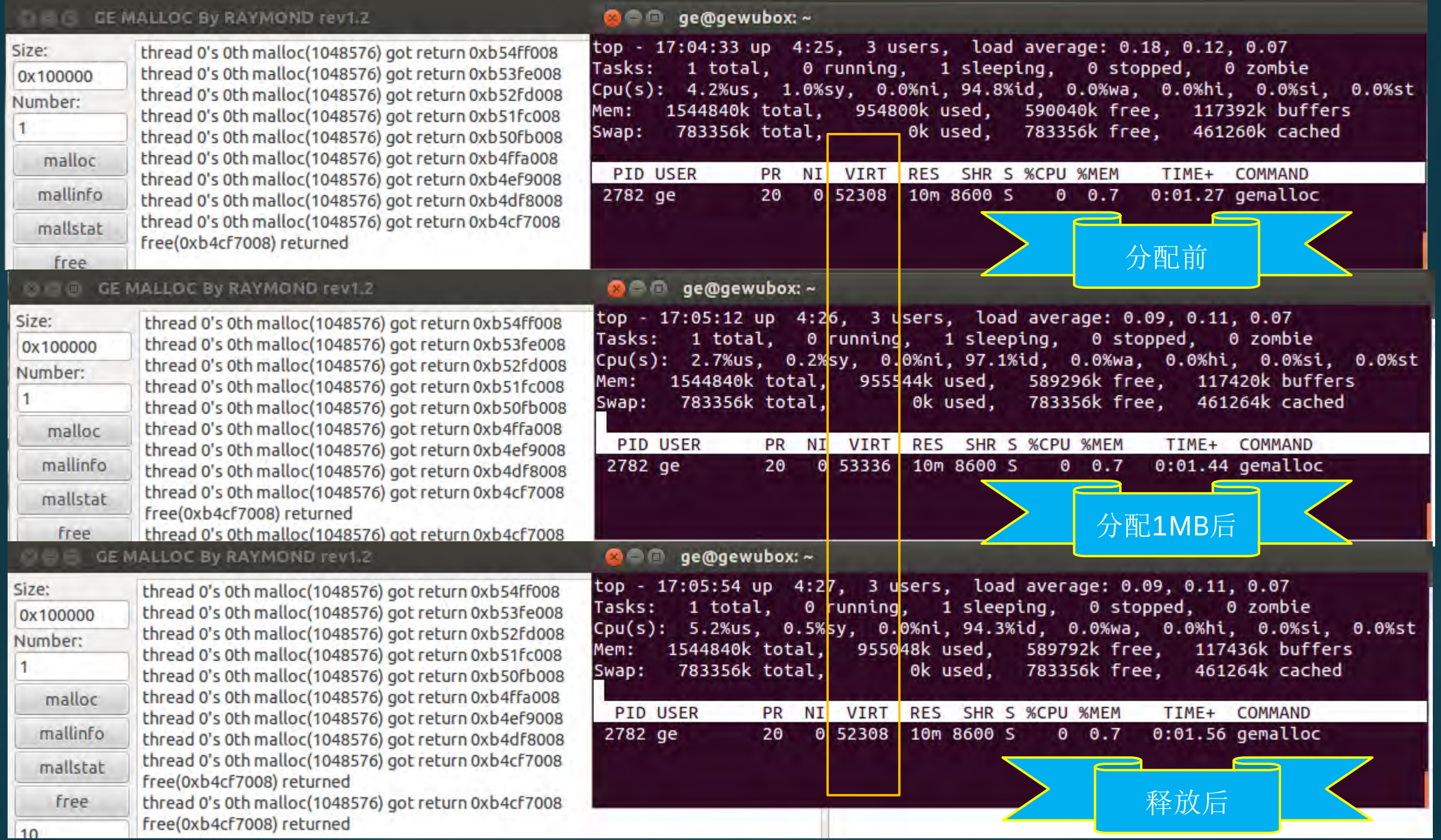
free

```
ge@gewubox:~/work/heap$ free -l -t
             total       used       free     shared    buffers     cached
Mem:        766212     676192      90020          0      12348     275304
Low:        766212     676192      90020
High:            0          0          0
-/+ buffers/cache:     388540     377672
Swap:       783356       5216     778140
Total:     1549568     681408     868160
```

top



```
top - 17:05:20 up  9:10,  3 users,  load average: 0.13, 0.09, 0.28
Tasks: 162 total,   1 running, 161 sleeping,   0 stopped,   0 zombie
Cpu(s):  3.1%us,  0.7%sy,  0.0%ni, 96.2%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:    766212k total,   677876k used,    88336k free,    12392k buffers
Swap:   783356k total,     5216k used,   778140k free,   275400k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1065 root      20   0  142m  94m  20m S  4.0 12.6   1:32.29 Xorg
 2406 ge        20   0 96536  16m  10m S  2.3  2.2   0:18.07 gnome-terminal
 1917 ge        20   0  119m  12m 9984 S  0.3  1.7   0:06.95 metacity
 1936 ge        20   0  254m  53m  29m S  0.3  7.1   0:32.82 unity-2d-shell
 3804 ge        20   0  152m  35m  27m S  0.3  4.7   0:02.81 unity-2d-spread
    1 root      20   0  3520 1784 1228 S  0.0  0.2   0:03.21 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kthreadd
    3 root      20   0     0    0    0 S  0.0  0.0   0:00.29 ksoftirqd/0
    5 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 kworker/0:0H
    7 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    8 root      20   0     0    0    0 S  0.0  0.0   0:00.00 rcu_bh
    9 root      20   0     0    0    0 S  0.0  0.0   0:00.84 rcu_sched
   10 root      RT   0     0    0    0 S  0.0  0.0   0:00.59 watchdog/0
   11 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 khelper
   12 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kdevtmpfs
   13 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 netns
   14 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 writeback
   15 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 kintegrityd
   16 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 bioset
   17 root       0 -20     0    0    0 S  0.0  0.0   0:00.10 kworker/u3:0
   18 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 kblockd
   19 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 ata_sff
   20 root      20   0     0    0    0 S  0.0  0.0   0:00.05 khubd
```


GE MALLOC By RAYMOND rev1.2
Size:
0x100000
Number:
1
malloc
mallinfo
mallstat
free
thread 0's 0th malloc(1048576) got return 0xb54ff008
thread 0's 0th malloc(1048576) got return 0xb53fe008
thread 0's 0th malloc(1048576) got return 0xb52fd008
thread 0's 0th malloc(1048576) got return 0xb51fc008
thread 0's 0th malloc(1048576) got return 0xb50fb008
thread 0's 0th malloc(1048576) got return 0xb4ffa008
thread 0's 0th malloc(1048576) got return 0xb4ef9008
thread 0's 0th malloc(1048576) got return 0xb4df8008
thread 0's 0th malloc(1048576) got return 0xb4cf7008
free(0xb4cf7008) returned
ge@gewubox: ~
top - 17:04:33 up 4:25, 3 users, load average: 0.18, 0.12, 0.07
Tasks: 1 total, 0 running, 1 sleeping, 0 stopped, 0 zombie
Cpu(s): 4.2%us, 1.0%sy, 0.0%ni, 94.8%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 1544840k total, 954800k used, 590040k free, 117392k buffers
Swap: 783356k total, 0k used, 783356k free, 461260k cached
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
2782 ge 20 0 52308 10m 8600 S 0 0.7 0:01.27 gemalloc
分配前
GE MALLOC By RAYMOND rev1.2
Size:
0x100000
Number:
1
malloc
mallinfo
mallstat
free
thread 0's 0th malloc(1048576) got return 0xb54ff008
thread 0's 0th malloc(1048576) got return 0xb53fe008
thread 0's 0th malloc(1048576) got return 0xb52fd008
thread 0's 0th malloc(1048576) got return 0xb51fc008
thread 0's 0th malloc(1048576) got return 0xb50fb008
thread 0's 0th malloc(1048576) got return 0xb4ffa008
thread 0's 0th malloc(1048576) got return 0xb4ef9008
thread 0's 0th malloc(1048576) got return 0xb4df8008
thread 0's 0th malloc(1048576) got return 0xb4cf7008
free(0xb4cf7008) returned
ge@gewubox: ~
top - 17:05:12 up 4:26, 3 users, load average: 0.09, 0.11, 0.07
Tasks: 1 total, 0 running, 1 sleeping, 0 stopped, 0 zombie
Cpu(s): 2.7%us, 0.2%sy, 0.0%ni, 97.1%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 1544840k total, 955544k used, 589296k free, 117420k buffers
Swap: 783356k total, 0k used, 783356k free, 461264k cached
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
2782 ge 20 0 53336 10m 8600 S 0 0.7 0:01.44 gemalloc
分配1MB后
GE MALLOC By RAYMOND rev1.2
Size:
0x100000
Number:
1
malloc
mallinfo
mallstat
free
10
thread 0's 0th malloc(1048576) got return 0xb54ff008
thread 0's 0th malloc(1048576) got return 0xb53fe008
thread 0's 0th malloc(1048576) got return 0xb52fd008
thread 0's 0th malloc(1048576) got return 0xb51fc008
thread 0's 0th malloc(1048576) got return 0xb50fb008
thread 0's 0th malloc(1048576) got return 0xb4ffa008
thread 0's 0th malloc(1048576) got return 0xb4ef9008
thread 0's 0th malloc(1048576) got return 0xb4df8008
thread 0's 0th malloc(1048576) got return 0xb4cf7008
free(0xb4cf7008) returned
thread 0's 0th malloc(1048576) got return 0xb4cf7008
free(0xb4cf7008) returned
ge@gewubox: ~
top - 17:05:54 up 4:27, 3 users, load average: 0.09, 0.11, 0.07
Tasks: 1 total, 0 running, 1 sleeping, 0 stopped, 0 zombie
Cpu(s): 5.2%us, 0.5%sy, 0.0%ni, 94.3%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 1544840k total, 955048k used, 589792k free, 117436k buffers
Swap: 783356k total, 0k used, 783356k free, 461264k cached
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
2782 ge 20 0 52308 10m 8600 S 0 0.7 0:01.56 gemalloc
释放后

堆陷阱

- 多次释放
- 使用释放后的块
- 释放野指针
- 溢出

- 刀刀致命

IOActive™
COMPREHENSIVE COMPUTER SECURITY SERVICES

Understanding the heap by breaking it

A case study of the heap as a persistent data structure through non-traditional exploitation techniques

Justin N. Ferguson // BH2007

Understanding the heap by breaking it

A case study of the heap as a persistent data structure through non-traditional exploitation techniques

Abstract:

Traditional exploitation techniques of overwriting heap metadata has been discussed ad-nauseum, however due to this common perspective the flexibility in abuse of the heap is commonly overlooked. This paper examines a flaw that was found in several popular implementations of the GSS-API as a method for elaborating upon the true beauty of data structure exploitation. This paper focuses on the dynamic memory management implementation provided by the GNU C library, particularly ptmalloc2 and presents methods for evading certain sanity checks in the library along with previously unpublished methods for obtaining control.

Outline:

Double Free

```
*** glibc detected *** ./geheapd: double free or corruption (!prev): 0x099ba008 ***
======= Backtrace: =========
/lib/i386-linux-gnu/libc.so.6(+0x74fd2)[0xb75f1fd2]
./geheapd[0x80487ba]
/lib/i386-linux-gnu/libc.so.6(__libc_start_main+0xf3)[0xb7596533]
./geheapd[0x8048471]
======= Memory map: ========
08048000-08049000 r-xp 00000000 08:01 19         /home/ge/work/heap/geheapd
08049000-0804a000 r--p 00000000 08:01 19         /home/ge/work/heap/geheapd
0804a000-0804b000 rw-p 00001000 08:01 19         /home/ge/work/heap/geheapd
099ba000-099db000 rw-p 00000000 00:00 0          [heap]
b754e000-b756a000 r-xp 00000000 08:01 132094     /lib/i386-linux-gnu/libgcc_s.so.1
b756a000-b756b000 r--p 0001b000 08:01 132094     /lib/i386-linux-gnu/libgcc_s.so.1
b756b000-b756c000 rw-p 0001c000 08:01 132094     /lib/i386-linux-gnu/libgcc_s.so.1
b757c000-b757d000 rw-p 00000000 00:00 0
b757d000-b7720000 r-xp 00000000 08:01 154504     /lib/i386-linux-gnu/libc-2.15.so
b7720000-b7722000 r--p 001a3000 08:01 154504     /lib/i386-linux-gnu/libc-2.15.so
b7722000-b7723000 rw-p 001a5000 08:01 154504     /lib/i386-linux-gnu/libc-2.15.so
b7723000-b7726000 rw-p 00000000 00:00 0
b7734000-b7735000 rw-p 00000000 00:00 0
b7735000-b7736000 rw-p 00000000 00:00 0
b7736000-b7738000 rw-p 00000000 00:00 0
b7738000-b7739000 r-xp 00000000 00:00 0          [vdso]
b7739000-b7759000 r-xp 00000000 08:01 154510     /lib/i386-linux-gnu/ld-2.15.so
b7759000-b775a000 r--p 0001f000 08:01 154510     /lib/i386-linux-gnu/ld-2.15.so
b775a000-b775b000 rw-p 00020000 08:01 154510     /lib/i386-linux-gnu/ld-2.15.so
bf77c000-bff20000 rw-p 00000000 00:00 0          [stack]
Aborted (core dumped)
```

```
for (j = freeBegin; j < freeEnd; j += freeStep)
    free(alloc[j]);

if(argv[3] != NULL && strcmp(argv[3],"df")==0 )
{
    printf("doing double free now\n");
    free(alloc[0]);
}
```

第二次释放80字节的块

```
Program received signal SIGSEGV, Segmentation fault.
0x0804dc8e in _int_malloc (av=0x80521a0, bytes=46) at malloc.c:3876
warning: Source file is more recent than executable.
3876            bck->fd = bin;
(gdb) bt
#0  0x0804dc8e in _int_malloc (av=0x80521a0, bytes=46) at malloc.c:3876
#1  0x0804d734 in calloc (n=1, elem_size=46) at malloc.c:3633
#2  0xb79b3753 in ?? () from /lib/i386-linux-gnu/libglib-2.0.so.0
#3  0xb79b3e6b in g_malloc0 () from /lib/i386-linux-gnu/libglib-2.0.so.0
#4  0xb73a7cae in ?? () from /usr/lib/i386-linux-gnu/libpango-1.0.so.0
...
#15 0x0804987a in append_list (szMsg=0xbfffe17c) at gemalloc.c:27
#16 0x080498e0 in d4d (format=0x804faa2 "free(%p) returned") at gemalloc.c:40
#17 0x08049a94 in button_free_clicked (data=0x0) at gemalloc.c:101
```

- 释放返回后
- 再分配时非法访问

二次释放8字节的块

```
#0  0x0804e749 in malloc_consolidate (av=0xb6000010) at malloc.c:4415
#1  0x0804dd48 in _int_malloc (av=0xb6000010, bytes=513) at malloc.c:3900
#2  0x0804ea68 in _int_realloc (av=0xb6000010, oldmem=0xb60102b8, bytes=512) at malloc.c:4541
#3  0x0804d2b2 in realloc (oldmem=0xb60102b8, bytes=512) at malloc.c:3489
...
#20 0xb7a8a2cc in g_signal_emit_valist () from /usr/lib/i386-linux-gnu/libgobject-2.0.so.0
#21 0xb7e3dcaa in gtk_signal_emit_by_name () from /usr/lib/i386-linux-gnu/libgtk-x11-2.0.so.0
#22 0xb7e082d9 in ?? () from /usr/lib/i386-linux-gnu/libgtk-x11-2.0.so.0
#23 0xb7e13e71 in ?? () from /usr/lib/i386-linux-gnu/libgtk-x11-2.0.so.0
#24 0xb7e0efa0 in gtk_clist_append () from /usr/lib/i386-linux-gnu/libgtk-x11-2.0.so.0
#25 0x0804987a in append_list (szMsg=0xbfffe17c) at gemalloc.c:27
#26 0x080498e0 in d4d (format=0x804faa2 "free(%p) returned") at gemalloc.c:40
#27 0x08049a94 in button_free_clicked (data=0x0) at gemalloc.c:101
```

- 释放返回
- 再分配时合并块时死循环

```
Program received signal SIGSEGV, Segmentation fault.
0x0804d049 in free (mem=0xb4cf7008) at malloc.c:3398
3398      if (chunk_is_mmapped(p))                      /* release mmapped memory. */
(gdb) bt
#0  0x0804d049 in free (mem=0xb4cf7008) at malloc.c:3398
#1  0x08049a7f in button_free_clicked (data=0x0) at gemalloc.c:99
```

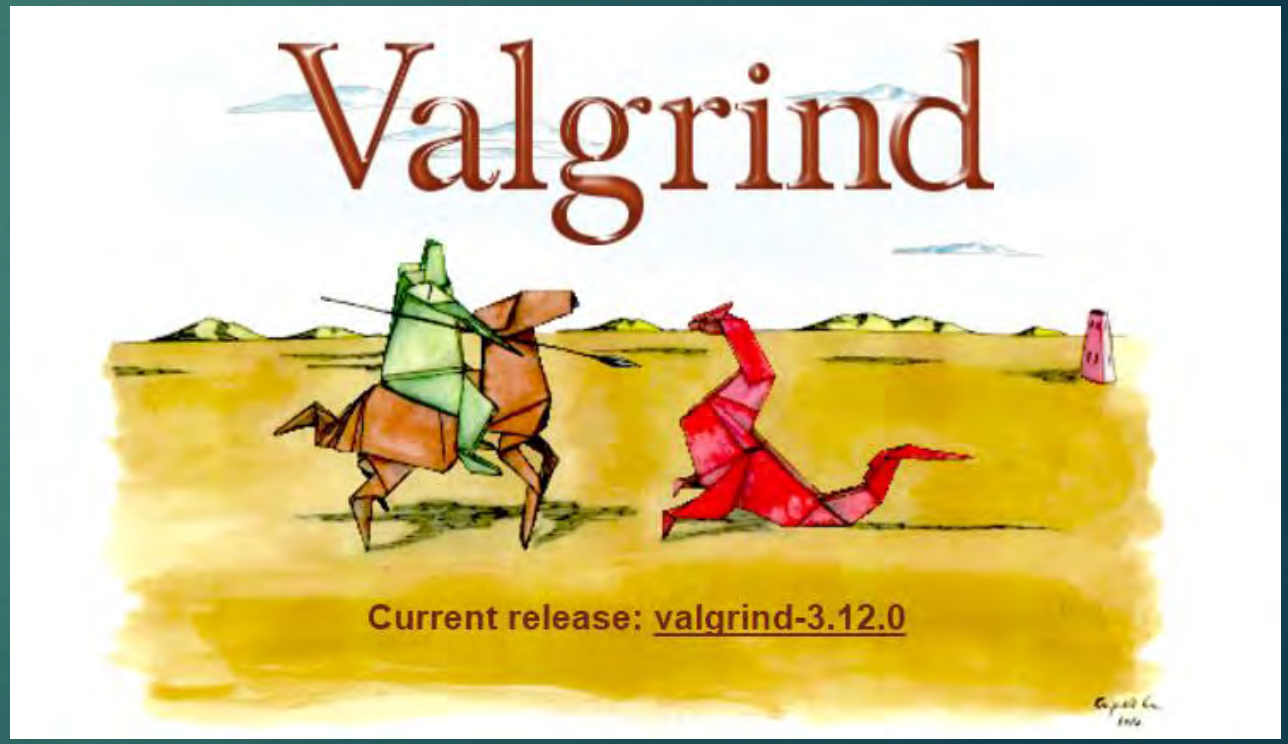
Double free超大块

```
/*
  Debugging:

  Because freed chunks may be overwritten with bookkeeping fields, this
  malloc will often die when freed memory is overwritten by user
  programs.  This can be very effective (albeit in an annoying way)
  in helping track down dangling pointers.

  If you compile with -DMALLOC_DEBUG, a number of assertion checks are
  enabled that will catch more memory errors. You probably won't be
  able to make much sense of the actual assertion errors, but they
  should help you locate incorrectly overwritten memory.  The checking
  is fairly extensive, and will slow down execution
  noticeably. Calling malloc_stats or mallinfo with MALLOC_DEBUG set
  will attempt to check every non-mmapped allocated and free chunk in
  the course of computing the summmaries. (By nature, mmapped regions
  cannot be checked very much automatically.)

  Setting MALLOC_DEBUG may also be helpful if you are trying to modify
  this code. The assertions in the check routines spell out in more
  detail the assumptions and invariants underlying the algorithms.

  Setting MALLOC_DEBUG does NOT provide an automated mechanism for
  checking that all accesses to malloced memory stay within their
  bounds. However, there are several add-ons and adaptations of this
  or other mallocs available that do this.
*/
```

```
Continuing.
*** glibc detected *** /home/ge/work/geheap/gemalloc: corrupted double-linked list: 0xb58037c0 ***
======= Backtrace: =========
/lib/i386-linux-gnu/libc.so.6(+0x75002)[0xb79bf002]
/lib/i386-linux-gnu/libc.so.6(+0x76050)[0xb79c0050]
/lib/i386-linux-gnu/libglib-2.0.so.0(+0x4cccb)[0xb7397ccb]
/lib/i386-linux-gnu/libglib-2.0.so.0(g_free+0x20)[0xb7397f50]
/usr/lib/i386-linux-gnu/libgdk-x11-2.0.so.0(gdk_region_destroy+0x30)[0xb7794a20]
/usr/lib/i386-linux-gnu/libgdk-x11-2.0.so.0(+0x3debc)[0xb77a5ebc]
/usr/lib/i386-linux-gnu/libgdk-x11-2.0.so.0(+0x3dfd7)[0xb77a5fd7]
/usr/lib/i386-linux-gnu/libgdk-x11-2.0.so.0(+0x3e1b8)[0xb77a61b8]
/usr/lib/i386-linux-gnu/libgdk-x11-2.0.so.0(gdk_window_hide+0xc9)[0xb77a8869]
/usr/lib/liboverlay-scrollbar-0.2.so.0(+0x412c)[0xb6e1212c]
/usr/lib/liboverlay-scrollbar-0.2.so.0(+0x80f1)[0xb6e160f1]
/usr/lib/i386-linux-gnu/libgobject-2.0.so.0(g_cclosure_marshal_VOID__VOID+0x8c)[0xb7b211ec]
/usr/lib/i386-linux-gnu/libgobject-2.0.so.0(g_closure_invoke+0x184)[0xb7b1f484]
/usr/lib/i386-linux-gnu/libgobject-2.0.so.0(+0x1f0d9)[0xb7b310d9]
/usr/lib/i386-linux-gnu/libgobject-2.0.so.0(g_signal_emit_valist+0xcfc)[0xb7b392cc]
/usr/lib/i386-linux-gnu/libgtk-x11-2.0.so.0(gtk_signal_emit_by_name+0xca)[0xb7e3dcaa]
/usr/lib/i386-linux-gnu/libgtk-x11-2.0.so.0(+0x2a7065)[0xb7e08065]
/usr/lib/i386-linux-gnu/libgtk-x11-2.0.so.0(+0x2b2e71)[0xb7e13e71]
/usr/lib/i386-linux-gnu/libgtk-x11-2.0.so.0(gtk_clist_append+0xa0)[0xb7e0efa0]
/home/ge/work/geheap/gemalloc[0x8048ffe]
/home/ge/work/geheap/gemalloc[0x8049058]
/home/ge/work/geheap/gemalloc[0x8049199]
/lib/i386-linux-gnu/libpthread.so.0(+0x6d4c)[0xb7afcd4c]
/lib/i386-linux-gnu/libc.so.6(clone+0x5e)[0xb7a3b87e]
======= Memory map: ========
08048000-0804a000 r-xp 00000000 08:01 1835228    /home/ge/work/geheap/gemalloc
0804a000-0804b000 r--p 00001000 08:01 1835228    /home/ge/work/geheap/gemalloc
0804b000-0804c000 rw-p 00002000 08:01 1835228    /home/ge/work/geheap/gemalloc
0804c000-0806d000 rw-p 00000000 00:00 0          [heap]
0806d000-08090000 rw-p 00000000 00:00 0          [heap]
08090000-080b1000 rw-p 00000000 00:00 0          [heap]
080b1000-080d2000 rw-p 00000000 00:00 0          [heap]
```

Valgrind

- an instrumentation framework for building dynamic analysis tools
- There are Valgrind tools that can automatically detect many memory management and threading bugs, and profile your programs in detail
- You can also use Valgrind to build new tools
- 安装：sudo apt-get install valgrind



基本原理

- A program running under Valgrind is not executed directly by the CPU. Instead it runs on a synthetic CPU provided by Valgrind.
- Your program is then run on a synthetic CPU provided by the Valgrind core. As new code is executed for the first time, the core hands the code to the selected tool. The tool adds its own instrumentation code to this and hands the result back to the core, which coordinates the continued execution of this instrumented code.
- This is why a debugger cannot debug your program when it runs on Valgrind.

包一层

```
int foo ( int x, int y ) { return x + y; }
```

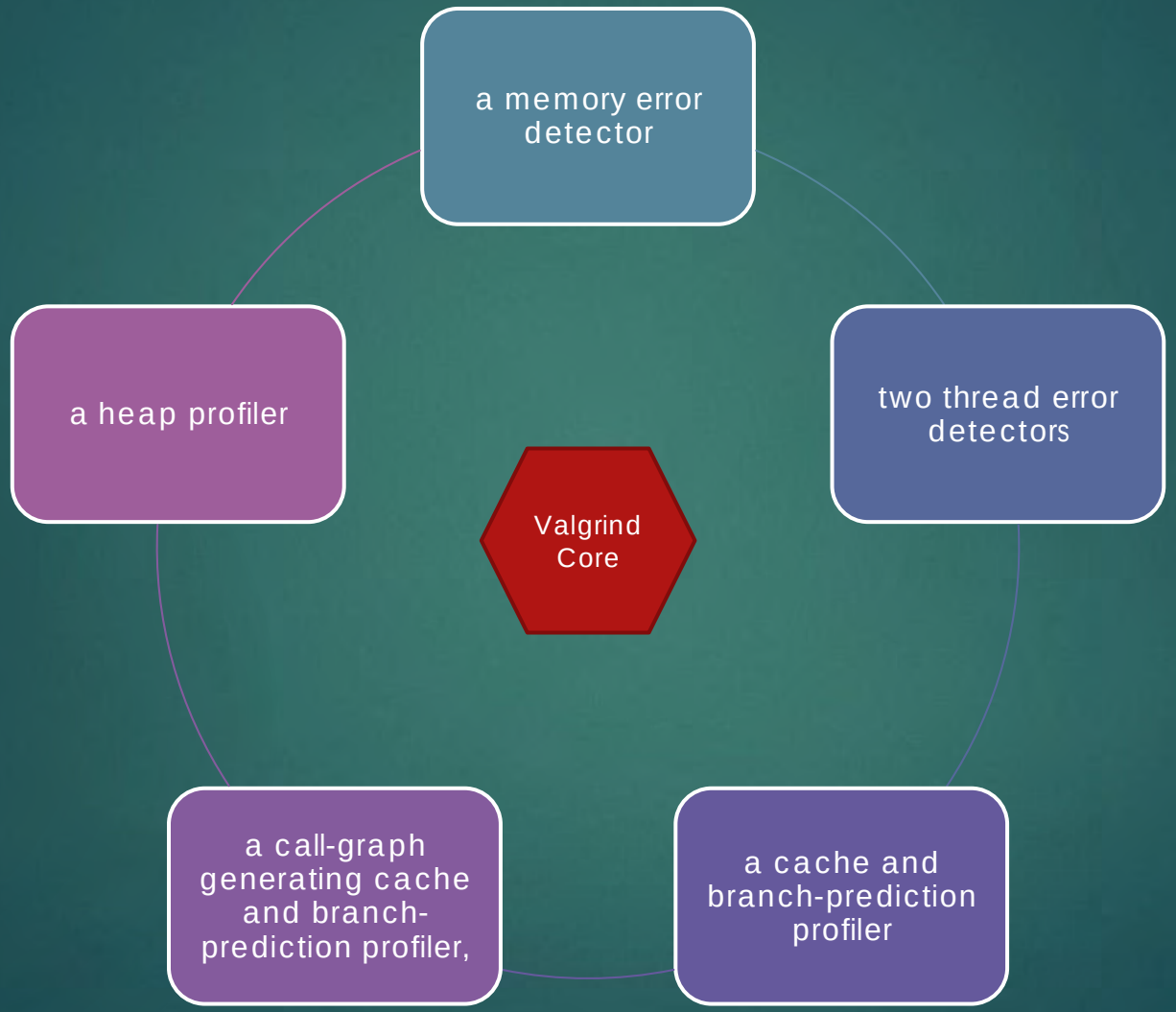
```
#include <stdio.h>
#include "valgrind.h"
int I_WRAP_SONAME_FNNAME_ZU(NONE,foo)( int x, int y )
{
   int    result;
   OrigFn fn;
   VALGRIND_GET_ORIG_FN(fn);
   printf("foo's wrapper: args %d %d \n", x, y);
   CALL_FN_W_WW(result, fn, x,y);
   printf("foo's wrapper: result %d \n", result);
   return result;
}
```

- http://valgrind.org/docs/manual/manual-core-adv.html#manual-core-adv.wrapping

兼容gdb

- valgrind --vgdb=yes --vgdb-error=0 prog
- gdb prog
- (gdb) target remote | vgdb

六大工具


a memory error detector
two thread error detectors
a cache and branch-prediction profiler
a call-graph generating cache and branch-prediction profiler,
a heap profiler
Valgrind Core

Memcheck: a memory error detector

- Accessing memory you shouldn't, e.g. overrunning and underrunning heap blocks, overrunning the top of the stack, and accessing memory after it has been freed.
- Using undefined values, i.e. values that have not been initialised, or that have been derived from other undefined values.
- Incorrect freeing of heap memory, such as double-freeing heap blocks, or mismatched use of malloc/new/new[] versus free/delete/delete[]
- Overlapping src and dst pointers in memcpy and related functions.
- Passing a fishy (presumably negative) value to the size parameter of a memory allocation function.
- Memory leaks.

检测泄露

```
==4724== Warning: client switching stacks?  SP change: 0xbe5fa040 --> 0xbed9b270
==4724==          to suppress, use: --max-stackframe=8000048 or greater
==4724==
==4724== HEAP SUMMARY:
==4724==     in use at exit: 2,000 bytes in 2 blocks
==4724==   total heap usage: 2 allocs, 0 frees, 2,000 bytes allocated
==4724==
==4724== 2,000 bytes in 2 blocks are definitely lost in loss record 1 of 1
==4724==    at 0x402BE68: malloc (in /usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==4724==    by 0x804872F: main (geheap.c:54)
==4724==
==4724== LEAK SUMMARY:
==4724==    definitely lost: 2,000 bytes in 2 blocks
==4724==    indirectly lost: 0 bytes in 0 blocks
==4724==      possibly lost: 0 bytes in 0 blocks
==4724==    still reachable: 0 bytes in 0 blocks
==4724==         suppressed: 0 bytes in 0 blocks
==4724==
==4724== For counts of detected and suppressed errors, rerun with: -v
==4724== ERROR SUMMARY: 51 errors from 32 contexts (suppressed: 0 from 0)
```

- valgrind --tool=memcheck --leak-check=yes ./geheapd 2 1000 leak

泄露报告

```
==3145== 720 bytes in 9 blocks are definitely lost in loss record 2,943 of 3,217
==3145==    at 0x402BE68: malloc (in /usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==3145==    by 0x8049343: do_malloc (gemalloc.c:57)
==3145==    by 0x8049434: button_malloc_clicked (gemalloc.c:78)
==3145==    by 0x456C242: g_cclosure_marshal_VOID__VOIDv (in /usr/lib/i386-linux-gnu/libgobject-2.0.so
==3145==    by 0x456A726: ??? (in /usr/lib/i386-linux-gnu/libgobject-2.0.so.0.3200.4)
==3145==    by 0x4583A18: g_signal_emit_valist (in /usr/lib/i386-linux-gnu/libgobject-2.0.so.0.3200.4)
==3145==    by 0x4584442: g_signal_emit (in /usr/lib/i386-linux-gnu/libgobject-2.0.so.0.3200.4)
==3145==    by 0x40BF289: gtk_button_clicked (in /usr/lib/i386-linux-gnu/libgtk-x11-2.0.so.0.2400.10)
==3145==    by 0x40C069F: ??? (in /usr/lib/i386-linux-gnu/libgtk-x11-2.0.so.0.2400.10)
==3145==    by 0x456C1EB: g_cclosure_marshal_VOID__VOID (in /usr/lib/i386-linux-gnu/libgobject-2.0.so.0
==3145==    by 0x45692FC: ??? (in /usr/lib/i386-linux-gnu/libgobject-2.0.so.0.3200.4)
```

多次释放

```
==4727==
==4727== Invalid free() / delete / delete[] / realloc()
==4727==    at 0x402B06C: free (in /usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==4727==    by 0x8048800: main (geheap.c:69)
==4727==  Address 0x41ef028 is 0 bytes inside a block of size 1,000 free'd
==4727==    at 0x402B06C: free (in /usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==4727==    by 0x8048800: main (geheap.c:69)
==4727==
==4727==
==4727== More than 10000000 total errors detected.  I'm not reporting any more.
==4727== Final error counts will be inaccurate.  Go fix your program!
==4727== Rerun with --error-limit=no to disable this cutoff.  Note
==4727== that errors may occur in your program without prior warning from
==4727== Valgrind, because errors are no longer being displayed.
```

- valgrind --tool=memcheck --leak-check=yes ./geheapd 2 1000 df

```
==3164== Invalid free() / delete / delete[] / realloc()
==3164==    at 0x402B06C: free (in /usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==3164==    by 0x80494AE: button_free_clicked (gemalloc.c:107)
==3164==    by 0x456C242: g_cclosure_marshal_VOID__VOIDv (in /usr/lib/i386-linux-gnu/libgobject-2.0.so.0.32
==3164==    by 0x456A726: ??? (in /usr/lib/i386-linux-gnu/libgobject-2.0.so.0.3200.4)
==3164==    by 0x4583A18: g_signal_emit_valist (in /usr/lib/i386-linux-gnu/libgobject-2.0.so.0.3200.4)
==3164==    by 0x4584442: g_signal_emit (in /usr/lib/i386-linux-gnu/libgobject-2.0.so.0.3200.4)
==3164==    by 0x40BF289: gtk_button_clicked (in /usr/lib/i386-linux-gnu/libgtk-x11-2.0.so.0.2400.10)
==3164==    by 0x40C069F: ??? (in /usr/lib/i386-linux-gnu/libgtk-x11-2.0.so.0.2400.10)
==3164==    by 0x456C1EB: g_cclosure_marshal_VOID__VOID (in /usr/lib/i386-linux-gnu/libgobject-2.0.so.0.320
==3164==    by 0x45692FC: ??? (in /usr/lib/i386-linux-gnu/libgobject-2.0.so.0.3200.4)
==3164==  Address 0x6f68e00 is 0 bytes inside a block of size 80 free'd
==3164==    at 0x402B06C: free (in /usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==3164==    by 0x80494AE: button_free_clicked (gemalloc.c:107)
==3164==    by 0x456C242: g_cclosure_marshal_VOID__VOIDv (in /usr/lib/i386-linux-gnu/libgobject-2.0.so.0.32
==3164==    by 0x456A726: ??? (in /usr/lib/i386-linux-gnu/libgobject-2.0.so.0.3200.4)
==3164==    by 0x4583A18: g_signal_emit_valist (in /usr/lib/i386-linux-gnu/libgobject-2.0.so.0.3200.4)
==3164==    by 0x4584442: g_signal_emit (in /usr/lib/i386-linux-gnu/libgobject-2.0.so.0.3200.4)
==3164==    by 0x40BF289: gtk_button_clicked (in /usr/lib/i386-linux-gnu/libgtk-x11-2.0.so.0.2400.10)
==3164==    by 0x40C069F: ??? (in /usr/lib/i386-linux-gnu/libgtk-x11-2.0.so.0.2400.10)
==3164==    by 0x456C1EB: g_cclosure_marshal_VOID__VOID (in /usr/lib/i386-linux-gnu/libgobject-2.0.so.0.320
==3164==    by 0x45692FC: ??? (in /usr/lib/i386-linux-gnu/libgobject-2.0.so.0.3200.4)
==3164==
```

切问而近思

2017 CPP-Summit

C++17 was not that great. How can C++20, 23 do better?

Michael Wong, Codeplay Software,
VP of Research and Development
Chair of SYCL Heterogeneous Programming Language
ISOCPP.org Director, VP http://isocpp.org/wiki/faq/wg21#michael-wong
Head of Delegation for C++ Standard for Canada
Chair of Programming Languages for Standards Council of Canada

Chair of WG21 SG5 Transactional Memory
Chair of WG21 SG14 Games Dev/Low Latency/Financial Trading/Embedded
Editor: C++ SG5 Transactional Memory Technical Specification
Editor: C++ SG1 Concurrency Technical Specification
http:://wongmichael.com/about

Acknowledgement Disclaimer

Numerous people internal and external to the original C++/Khronos group, in industry and academia, have made contributions, influenced ideas, written part of this presentations, and offered feedbacks to form part of this talk.
I even lifted this acknowledgement and disclaimer from some of them.
But I claim all credit for errors, and stupid mistakes. **These are mine, all mine!**

Legal Disclaimer

This work represents the view of the author and does not necessarily represent the view of Codeplay.

Other company, product, and service names may be trademarks or service marks of others.