

Why is this a memory leak? (Valgrind / Linux)

Source: <http://unix.derkeiler.com/Newsgroups/comp.unix.programmer/2004-12/0434.html>

From: John (*john_at_domain.invalid*)

Date: 12/14/04

Date: Tue, 14 Dec 2004 02:25:35 GMT

Linux (SuSE) x86 gcc (GCC) 3.3 20030226 (prerelease) (SuSE Linux)

I'm hunting down a memory leak reported by Valgrind. I've been able to write a small program that produces the valgrind report. That program is included below.

My question is based on this: If on line 45 I call `put_on_list()` instead of `call_pol()`, valgrind doesn't report any memory leaks. It only happens when I use one additional layer of function call to populate the list. See the LEAK SUMMARY section in the Valgrind output after the source listing.

Thanks for any help or clarification.

```
/* (line 1)
 * pointer_test.c
 *
 * gcc -g -o ptrt pointer_test.c
 */

#include <stdio.h>
#include <stdlib.h>

struct value_tag
{
    int v ;
    struct value_tag * next ;
};
typedef struct value_tag Value ;

struct value_list_tag
{
    Value * first ;
    Value * last ;
};
typedef struct value_list_tag ValueList ;
```

```
void
call_pol( Value *, ValueList * );

void
put_on_list( Value *, ValueList * );

void
print_list( ValueList * );

int
main(void)
{
    int i ;

    Value * val ;
    ValueList values = {0,0} ;

    for ( i = 0 ; i < 200 ; i++ )
    {
        val = calloc( 1, sizeof(Value) ) ;
        val->v = i ;
        call_pol( val, &values ) ;
    }

    print_list( &values ) ;

    return 0 ;
}

void
call_pol( Value * v, ValueList * vl )
{
    put_on_list( v, vl ) ;
}

void
put_on_list( Value * v, ValueList * vl )
{
    if ( vl->first == NULL )
    {
        vl->first = vl->last = v ;
        v->next = NULL ;
    }
    else
    {
        vl->last->next = v ;
        vl->last = v ;
    }
}
```

```
void
print_list( ValueList * vl )
{
    int i ;
    Value * val ;

    val = vl->first ;
    for ( i = 0 ; i < 200 ; i++ )
    {
        printf( "%d: %d\n", i, val->v ) ;
        val = val->next ;
    }
}
```

```
// VALGRIND OUTPUT (Trimmed Output)
```

```
john@jbox:~/misc> valgrind --tool=memcheck -v --leak-check=yes ./ptrt
==2953== Memcheck, a memory error detector for x86-linux.
==2953== Copyright (C) 2002-2004, and GNU GPL'd, by Julian Seward et
al.
==2953== Using valgrind-2.2.0, a program supervision framework for
x86-linux.
==2953== Copyright (C) 2000-2004, and GNU GPL'd, by Julian Seward et
al.
==2953== Valgrind library directory: /usr/local/lib/valgrind
==2953== Command line
==2953== ./ptrt
==2953== Startup, with flags:
==2953== --tool=memcheck
==2953== -v
==2953== --leak-check=yes
==2953== Contents of /proc/version:
==2953== Linux version 2.4.20-4GB (root@Pentium.suse.de) (gcc
version 3.3 20030226 (prerelease) (SuSE Linux)) #1 Wed Nov 17
20:28:01 UTC 2004
==2953== Reading syms from /home/john/misc/ptrt (0x8048000)
```

```
// valgrind startup and REDIRECT messages deleted
// program output deleted
```

```
==2953== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 12 from
1)
--2953--
--2953-- supp: 12 Ugly strchr error in /lib/ld-2.3.2.so
==2953== malloc/free: in use at exit: 1600 bytes in 200 blocks.
==2953== malloc/free: 200 allocs, 0 frees, 1600 bytes allocated.
==2953==
==2953== searching for pointers to 200 not-freed blocks.
==2953== checked 1408016 bytes.
==2953==
==2953== 8 bytes in 1 blocks are definitely lost in loss record 1 of
```

2

```
==2953== at 0x1B9037F9: calloc (vg_replace_malloc.c:176)
==2953== by 0x80483AF: main (pointer_test.c:43)
==2953==
==2953== LEAK SUMMARY:
==2953== definitely lost: 8 bytes in 1 blocks.
==2953== possibly lost: 0 bytes in 0 blocks.
==2953== still reachable: 1592 bytes in 199 blocks.
==2953== suppressed: 0 bytes in 0 blocks.
==2953== Reachable blocks (those to which a pointer was found) are
not shown.
==2953== To see them, rerun with: --show-reachable=yes
--2953-- TT/TC: 0 tc sectors discarded.
--2953-- 1994 tt_fast misses.
--2953-- translate: new 1872 (32779 -> 436799; ratio 133:10)
--2953-- discard 1 (32 -> 512; ratio 160:10).
--2953-- chainings: 1169 chainings, 2 unchainings.
--2953-- dispatch: 0 jumps (bb entries); of them 15402 (1540200%)
unchained.
--2953-- 426/2442 major/minor sched events.
--2953-- reg-alloc: 405 t-req-spill, 82171+2935 orig+spill uis,
--2953-- 10150 total-reg-rank
--2953-- sanity: 215 cheap, 9 expensive checks.
--2953-- ccalls: 7281 C calls, 54% saves+restores avoided (23538
bytes)
--2953-- 9587 args, avg 0.86 setup instrs each (2604
bytes)
--2953-- 0% clear the stack (21843 bytes)
--2953-- 3233 retvals, 28% of reg-reg movs avoided (1796
bytes)
john@jbox:~/misc>
```