

Catching the C/C++ bugs



Peter Toft



Version2 again



- I have been writing C-code for 19 years
 - Super efficient for communication applications
 - Super easy to make errors
- I have been fighting bugs for MANY years
- I have also written about this
 - e.g. <http://www.version2.dk/artikel/7351-mit-cc-programmeringsmiljoe>



ALL YOUR BUG
ARE BELONG
TO ME !



Catch the bug

A) Catch by the compiler

- Gcc – understand the output
- but try others too like llvm

} Here you should start

B) Catch by static code analysis

- Lint products
- Coverity

} Not too much in focus today

C) Catch by dynamic code analysis

- Valgrind/Valkyrie
- Purify
- Insure++

} This is fun :-)

.. still bugs

- D) what then
 - You have to think in verification and coding style



ALL YOUR BUG
ARE BELONG
TO ME!



Code style

- Enforce good coding practices

```
N = (int)log(a); a=(int *)calloc(N*sizeof(int));
```

replaced by

```
N = (int)log(a);
```

```
if (N>0)
```

```
    a=(int *)calloc(N*sizeof(int));
```

```
else {
```

```
    printf("N is non-positive=%i at %s:%i\n",N,__FILE__,__LINE__);
```

```
    exit(1);
```

```
}
```

ALL YOUR BUG
ARE BELONG
TO ME!



Compile work

- gcc **-Wall -pedantic** -o fart1 fart1.c
- cat fart1.c

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

```
int main(void)
```

```
{
```

```
    int a;
```

```
    if (a>0)
```

```
        printf("hej med dig\n");
```

```
    else
```

```
        /* If a is positive */
```

```
        * then print in *
```

```
        * danish */
```

```
    printf("hi there\n");
```

```
    return 0;
```

```
}
```

Emac

ALL YOUR BUG
ARE BELONG
TO ME!



Color-code in your favorite editor

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

int main(void)
{
    int a;

    if (a>0) /* If a is positive *
        printf("hej med dig\n"); /* then print in *
    else /* danish */


    printf("hi there\n");

    return 0;
}
```

ALL YOUR BUG
ARE BELONG
TO ME !



Static code analysis

- Lint products
 - Splint - <http://www.splint.org> - 
 - FlexeLint - <http://www.gimpel.com/html/flex.htm>
 - Coverity (super heavyweight commercial product)

splint output

part1.c: (in function main)

part1.c:8:9: Variable a used before definition

An rvalue is used that may not be initialized to a value on some execution

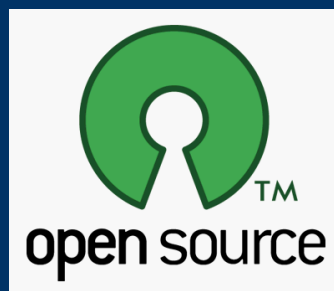
path. (Use -usedef to inhibit warning)

ALL YOUR BUG
ARE BELONG
TO ME !



Dynamic code analysis

- Valgrind/valkyrie
 - No re-compile/link
 - 10x slower
- Purity
 - 10x slower
 - Same compile / special link
- Insure++
 - 50x slower
 - Special compile and link
 - Harder to debug due to source code changed



fart2.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(void)
5  {
6      int a;
7      int b=a+1;
8      int *c;
9
10     c = (int *)malloc(b*sizeof(int));
11     c[0] = 12;
12     printf("%i %i\n",c[0],c[1]);
13     free(c);
14     printf("Done with computing\n");
15     free(c);
16
17     return 0;
18 }
```

No bug if I ask gcc -Wall



ALL YOUR BUG
ARE BELONG
TO ME !



\$ *splint* *fart2.c*

fart2.c: (in function main)

fart2.c:7:11: Variable a used before definition

An rvalue is used that may not be initialized to a value on some execution path. (Use `-usedef` to inhibit warning)

fart2.c:11:5: Index of possibly null pointer c: c

A possibly null pointer is dereferenced. Value is either the result of a function which may return null (in which case, code should check it is not null), or a global, parameter or structure field declared with the null qualifier. (Use `-nullderef` to inhibit warning)

fart2.c:10:9: Storage c may become null

fart2.c:15:10: Dead storage c passed as out parameter to free: c

Memory is used after it has been released (either by passing as an only param

or assigning to an only global). (Use `-userreleased` to inhibit warning)

fart2.c:13:10: Storage c released

ALL YOUR BUG
ARE BELONG
TO ME !



valgrind on fart2.c

```
$ gcc -g -Wall -pedantic -o fart2 fart2.c
```

```
$ ./fart2
```

Segmentation fault

```
$ valgrind ./fart2
```

```
==28329== Warning: silly arg (-67187548) to malloc()
```

```
==28329== Invalid write of size 4
```

```
==28329==   at 0x8048404: main (fart2.c:11)
```

```
==28329== Address 0x0 is not stack'd, malloc'd or (recently) free'd
```

```
==28329==
```

```
==28329== Process terminating with default action of signal 11 (SIGSEGV)
```

```
==28329== Access not within mapped region at address 0x0
```

```
==28329==   at 0x8048404: main (fart2.c:11)
```

ALL YOUR BUG
ARE BELONG
TO ME!



valkyrie on fart2.c

```
$ gcc -g -Wall -pedantic -o fart2 fart2.c
```

```
$ ./fart2
```

Segmentation fault

```
$ valkyrie ./fart2
```

```
File Tools Options Help
|| Run Valgrind Stop
|| [Icons]
Valgrind: FINISHED 'fart2' (wallclock runtime: 0.270secs)
Errors: 1 Leaked Bytes: 0 in 0 blocks
+ Memcheck output for process id ==28652== (parent pid ==28649==)
+ Preamble
+ IVW [1]: Invalid write of size 4
  stack
    0x8048404: main (fart2.c:11)
      int b=a+1;
      int *c;
      c = (int *)malloc(b*sizeof(int));
      c[0] = 12;
      printf("%i %i\n",c[0],c[1]);
      free(c);
      printf("Done with computing\n");
      free(c);
      Address 0x0 is not stack'd, malloc'd or (recently) free'd
+ Suppressed errors
```

ALL YOUR BUG
ARE BELONG
TO ME!



Purify on fart2.c

```
$ purify gcc -g -Wall -pedantic -o fart2 fart2.c  
$ ./fart2
```

```
Finished fart2 ( 3 errors, 0 leaked bytes)  
  Purify instrumented ./fart2 (pid 29732 at Sun Mar 22 18:07:28 2009)  
  Command-line: ./fart2  
  UMR: Uninitialized memory read  
  This is occurring while in thread 29732:  
    main [fart2.c:7]  
      int main(void)  
      {  
          int a;  
          int b=a+1;  
          int *c;  
  
          c = (int *)malloc(b*sizeof(int));  
      }  
    __libc_start_main [libc.so.6]  
    _start [crt1.o]  
    Reading 4 bytes from 0xbfffd3f8 on the stack of thread 29732.  
    Address 0xbfffd3f8 is local variable "a" in function main.  
  UMR: Uninitialized memory read  
  FUM: Freeing unallocated memory  
  Current file descriptors in use: 5  
  Memory leaked: 0 bytes (0%); potentially leaked: 0 bytes (0%)  
  Program exited with status code 0.
```

ALL YOUR BUG
ARE BELONG
TO ME !



Insure++ on fart2.c
\$ insure -g -Wall -pedantic -o fart2 fart2.c

The screenshot shows the Insure++ debugger window titled "Insra on comad001:3275". The interface includes a menu bar with "File", "Messages", and "Help". Below the menu is a toolbar with icons for "Previous", "Next", "Delete", "Suppress", "Sort", "Kill", and "Help". The main display area shows the following text:

```
Insure++: Instrumented "fart2.c" on comad001, pid=30840
Runtime: Executed "fart2" on comad001, pid=30954 [5 tot.]
[ ] [X] READ_UNINIT_MEM(read)          fart2.c : 7
+ {
+   int a;
>>> int b=a+1;
+   int *c;
+
[ ] [X] WRITE_NULL                    fart2.c : 11
+
+   c = (int *)malloc(b*sizeof(int));
>>> c[0] = 12;
+   printf("%i %i\n",c[0],c[1]);
+   free(c);
[ ] Insure trapped signal: 11
[ ] Problem summary
[ ] Leak summary
```

At the bottom of the window, there are two status boxes: "Connections: None" and "Messages: 5".

ALL YOUR BUG
ARE BELONG
TO ME!



Keep up the fight

Open source is good for me. I will fully embrace it.
Open source is good for me. I will fully embrace it.
Open source is good for me. I will fully embrace it.
Open source is good for me. I will fully embrace it.
Open source is good for me. I will fully embrace it.
Open source is good for me. I will fully embrace it.

