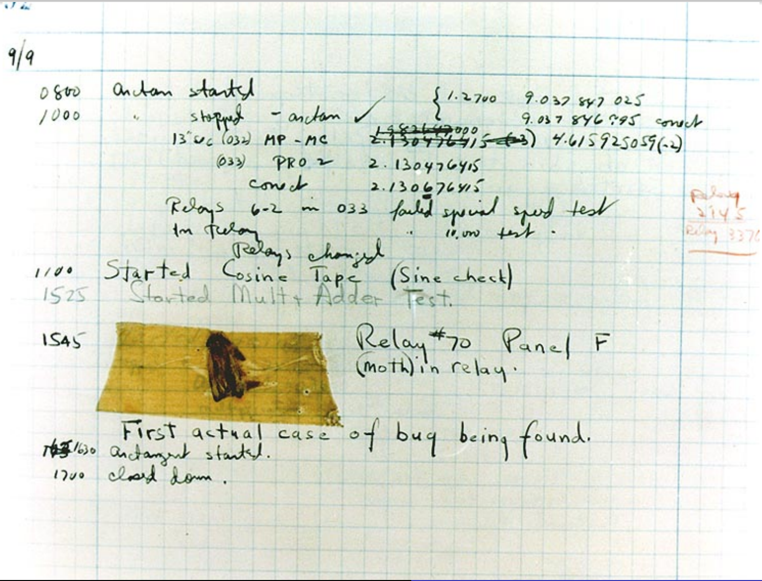
Core dumped - on debuggers and other tools

Pascal M. Vaudrevange

28.03.2008



9/9

0800 antan started
1000 " stopped - antan ✓ {1.2700 9.037 847 025
9.037 846 995 correct
13" sec (032) MP - MC ~~1.982147000~~ 2.130476415 (~~-3~~) 4.615925059(-2)
(033) PRO 2 2.130476415
correct 2.130676415
Relays 6-2 in 033 failed special speed test
In relay " 10,000 test .
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Mult + Adder Test.
1545 Relay #70 Panel F (moth) in relay.
First actual case of bug being found.
~~[illegible]~~1630 antangent started.
1700 closed down.

Relay 2145
Relay 3370

Motto

Everyone knows that debugging is twice as hard as writing a program in the first place. So if you're as clever as you can be when you write it, how will you ever debug it?

Brian Kernighan, ”The Elements of Programming Style”, 2nd edition, chapter 2

Makefile

```
CC=g++
CCOPTS=-O0 -g -W -Wall -Wunused -std=c++98 -pedantic
#CCOPTS=-O2 -W -Wall -Wunused -std=c++98
FILES=test.o

%.o: %.cpp test.h
	$(CC) $(CCOPTS) -c $@

test: $(FILES)
	$(CC) $(CCOPTS) $(FILES) -o test

clean:
	rm -f *~ *.o core
```

Makefile

```
CC=g++
CCOPTS=-O0 -g -W -Wall -Wunused -std=c++98 -pedantic
#CCOPTS=-O2 -W -Wall -Wunused -std=c++98
FILES=test.o

%.o: %.cpp test.h
	$(CC) $(CCOPTS) -c $@

test: $(FILES)
	$(CC) $(CCOPTS) $(FILES) -o test

clean:
	rm -f *~ *.o core
```

TABS!

Compiler Options

Option	Effect
-Wall	warn about questionable constructs
-Wextra	warn about even more events
	(no return value for a function etc)
-Wunused	unused variables
-pedantic -std=c++98	disallow extensions that affect portability
-g	**generate debugging information**
-O0	do not optimize
-O2	optimize

- different options for different vendors/ languages, e.g. gfortran, icc, ifort
- auto-parallelization for openmp in intel-compilers
  `-parallel -openmp`

Sample Program

```
#include<cstdlib>
#define NUM 10

int mess_things_up(int * values){
  int * tmp;
  int i;

  for(i=0; i<NUM; i++){
    tmp[i]=*values[i];
  }
}

int main(){
  int i, j;
  int * values;
  int result

  result=mess_things_up(&values);

  return 0;
}
```

Compiling

```
pascal@octonion:~/projects/talks/cerca_mar_2008/source> make
g++ -O0 -g -Wextra -Wall -Wunused -std=c++98 -pedantic -c test.cpp
test.cpp: In function int main():
test.cpp:14: warning: unused variable i
test.cpp:14: warning: unused variable j
test.cpp: In function int mess_things_up(int*):
test.cpp:11: warning: control reaches end of non-void function
g++ -O0 -g -Wextra -Wall -Wunused -std=c++98 -pedantic test.o -o test
pascal@octonion:~/projects/talks/cerca_mar_2008/source>
```

Repair warnings

```
#include<cstdlib>
#define NUM 10

int mess_things_up(int * values){
  int * tmp;
  int i;

  for(i=0; i<NUM; i++){
    tmp[i]=values[i];
  }

}

int main(){
   int i, j;
  int * values;
  int result;

  result=mess_things_up(values);

  return 0;
}
```

Repair warnings

```
#include<cstdlib>
#define NUM 10

int mess_things_up(int * values){
  int * tmp;
  int i;

  for(i=0; i<NUM; i++){
    tmp[i]=values[i];
  }
   return 0;
}

int main(){

  int * values;
  int result;

  result=mess_things_up(values);

  return 0;
}
```

Compiling

no more warnings:

```
pascal@octonion:~/projects/talks/cerca_mar_2008/source> make
g++ -O0 -g -Wextra -Wall -Wunused -std=c++98 -pedantic -c test.cpp
g++ -O0 -g -Wextra -Wall -Wunused -std=c++98 -pedantic test.o -o test
pascal@octonion:~/projects/talks/cerca_mar_2008/source>
```

but

```
pascal@octonion:~/projects/talks/cerca_mar_2008/source> ./test
Segmentation fault (core dumped)
pascal@octonion:~/projects/talks/cerca_mar_2008/source>
```

My First Debugger Session

```
pascal@octonion:~/projects/talks/cerca_mar_2008/source> gdb -q --dbx ./test
Using host libthread_db library "/lib/libthread_db.so.1".
(gdb) run
Starting program: /home/pascal/projects/talks/cerca_mar_2008/source/test

Program received signal SIGSEGV, Segmentation fault.
0x0000000000400568 in mess_things_up (values=0x7fff3770dcf0) at test.cpp:9
9           tmp[i]=values[i];
(gdb) p i
$1 = 0
(gdb) p tmp[i]
Cannot access memory at address 0x0
(gdb)
```

Missing malloc()

```
#include<cstdlib>
#define NUM 10

int mess_things_up(int * values){
  int * tmp;
  int i;


  for(i=0; i<NUM; i++){
    tmp[i]=values[i];
  }
  return 0;
}

int main(){
  int * values;
  int result;

  result=mess_things_up(values);

  return 0;
}
```

Missing malloc()

```
#include<cstdlib>
#define NUM 10

int mess_things_up(int * values){
  int * tmp;
  int i;
```

$$\boxed{\texttt{tmp = (int*) malloc(NUM*sizeof(int));}}$$

```
  for(i=0; i<NUM; i++){
    tmp[i]=values[i];
  }
  return 0;
}

int main(){
  int * values;
  int result;

  result=mess_things_up(values);

  return 0;
}
```

The world is good

```
pascal@octonion:~/projects/talks/cerca_mar_2008/source> make
g++ -O0 -g -Wextra -Wall -Wunused -std=c++98 -pedantic -c test.cpp
g++ -O0 -g -Wextra -Wall -Wunused -std=c++98 -pedantic test.o -o test
pascal@octonion:~/projects/talks/cerca_mar_2008/source> ./test
pascal@octonion:~/projects/talks/cerca_mar_2008/source>
```

More uses for gdb

```
pascal@octonion:~/projects/talks/cerca_mar_2008/source> gdb -q --dbx ./test
Using host libthread_db library "/lib/libthread_db.so.1".
(gdb) file test.cpp:main
10            tmp[i]=values[i];
11          }
12          return 0;
13        }
14
15        int main(){
16          int * values;
17          int result;
18
19          result=mess_things_up(values);
(gdb) stop in main
Breakpoint 1 at 0x4005e6: file test.cpp, line 19.
(gdb) run
Starting program: /home/pascal/projects/talks/cerca_mar_2008/source/test

Breakpoint 1, main () at test.cpp:19
19          result=mess_things_up(values);
(gdb) s
mess_things_up (values=0x7fff2051fb00) at test.cpp:8
8           tmp = (int*) malloc(NUM*sizeof(int));
(gdb) n
9           for(i=0; i<NUM; i++){
(gdb) n
10            tmp[i]=values[i];
```

```
(gdb) p i
$1 = 0
(gdb) n
9          for(i=0; i<NUM; i++){
(gdb) n
10           tmp[i]=values[i];
(gdb) p i
$2 = 1
(gdb) n
9          for(i=0; i<NUM; i++){
(gdb) set variable i = 10
(gdb) n
12         return 0;
(gdb) status
Num Type           Disp Enb Address            What
1   breakpoint     keep y   0x00000000004005e6 in main at test.cpp:19
        breakpoint already hit 1 time
(gdb) del 1
(gdb) status
No breakpoints or watchpoints.
(gdb)
```

gdb

Command	Function
file <filename:function—line>	load source file <filename>
stop in func	set breakpoint in functions
stop at <line>	set breakpoint in line <line>
bt	backtrace the call-stack
run <parameters>	
n	next instruction
s	next step
p <var>	print value of <var>
set variable <var> = <value>	set <var> to value <value>
status	show breakpoints
del <num>	delete breakpoint <num>
attach <pid>	attach to running process <pid>
watch	set watchpoint

valgrind

- detect memory management and threading bugs
- very detailed profiling to help find bottlenecks
- used by Firefox, Battlefield 1942, ROOT, R, . . .
- available modes

memcheck	fine-grained memory checker
cachegrind	annotate every line with the number of instructions executed and cache misses
callgrind	get call counts and inclusive cost for each call
helgrind	spots potential race conditions
massif	how much heap memory is used

Memory leaks

```
pascal@octonion:~/projects/talks/cerca_mar_2008/source> valgrind -q --leak-check=full ./test
==7154== Use of uninitialised value of size 8
==7154==    at 0x4005C8: mess_things_up(int*) (test.cpp:10)
==7154==    by 0x4005EE: main (test.cpp:19)
==7154==
==7154==
==7154== 40 bytes in 1 blocks are definitely lost in loss record 1 of 1
==7154==    at 0x4C21C16: malloc (vg_replace_malloc.c:149)
==7154==    by 0x40059D: mess_things_up(int*) (test.cpp:8)
==7154==    by 0x4005EE: main (test.cpp:19)
pascal@octonion:~/projects/talks/cerca_mar_2008/source>
```

More missing malloc() and free()

```
#include<cstdlib>
#define NUM 10

int mess_things_up(int * values){
  int * tmp;
  int i;

  tmp = (int*) malloc(NUM*sizeof(int));
  for(i=0; i<NUM; i++){
    tmp[i]=values[i];
  }
  free(tmp);
  return 0;
}

int main(){
  int * values;
  int result;
  values=(int*) malloc(NUM*sizeof(int));
  result=mess_things_up(values);
  free(values);
  return 0;
}
```

No more bugs!?

```
pascal@octonion:~/projects/talks/cerca_mar_2008/source> make
g++ -O0 -g -Wextra -Wall -Wunused -std=c++98 -pedantic -c test.cpp
g++ -O0 -g -Wextra -Wall -Wunused -std=c++98 -pedantic test.o -o test
pascal@octonion:~/projects/talks/cerca_mar_2008/source> valgrind -q --leak-check=full ./test
pascal@octonion:~/projects/talks/cerca_mar_2008/source>
```

```
pascal@octonion:~/projects/talks/cerca_mar_2008/source> vi Makefile
pascal@octonion:~/projects/talks/cerca_mar_2008/source> make clean && make
rm -f *~ *.o core test
g++ -O2 -W -Wextra -Wall -Wunused -std=c++98 -pedantic -c test.cpp
test.cpp: In function int main():
test.cpp:19: warning: values is used uninitialized in this function
g++ -O2 -W -Wextra -Wall -Wunused -std=c++98 -pedantic test.o -o test
pascal@octonion:~/projects/talks/cerca_mar_2008/source>
```

screen

```
                     Screen key bindings, page 1 of 2.

                     Command key:  ^A   Literal ^A:  a

break        ^B b        license      ,              removebuf    =
clear        C           lockscreen   ^X x           reset        Z
colon        :           log          H              screen       ^C c
copy         ^[ [        login        L              select       '
detach       ^D d        meta         a              silence      _
digraph      ^V          monitor      M              split        S
displays     *           next         ^@ ^N sp n     suspend      ^Z z
dumptermcap  .           number       N              time         ^T t
fit          F           only         Q              title        A
flow         ^F f        other        ^A             vbell        ^G
focus        ^I          pow_break    B              version      v
hardcopy     h           pow_detach   D              width        W
help         ?           prev         ^H ^P p ^?     windows      ^W w
history      { }         quit         \              wrap         ^R r
info         i           readbuf      <              writebuf     >
kill         K k         redisplay    ^L l           xoff         ^S s
lastmsg      ^M m        remove       X              xon          ^Q q

                [Press Space for next page; Return to end.]
```

strace

```
pascal@octonion:~/projects/talks/cerca_mar_2008/source> strace ./test 2>&1 |grep open
open("/etc/ld.so.cache", O_RDONLY)      = 3
open("/usr/lib/libstdc++.so.6", O_RDONLY) = 3
open("/lib/libm.so.6", O_RDONLY)        = 3
open("/lib/libgcc_s.so.1", O_RDONLY)    = 3
open("/lib/libc.so.6", O_RDONLY)        = 3
pascal@octonion:~/projects/talks/cerca_mar_2008/source>
```

Other thoughts

- regular expressions with grep, awk, bash ($\rightarrow$Pete)
- write **short** functions!
- vi and screen
- emacs and etags
- IDEs (Eclipse)
- versioning systems (CVS, Subversion, git)
- the right language for the job (C, Fortran, Perl, Python, Matlab)
- command line parameters
- ini-files
- libraries
- strip

Wise words from Turing award winners

The competent programmer is fully aware of the strictly limited size of his own skull; therefore he approaches the programming task in full humility, and among other things he avoids clever tricks like the plague.

Edsger Dijkstra (introduced RPN to computer science)

Wise words from Turing award winners

As soon as we started programming, we found to our surprise that it wasn't as easy to get programs right as we had thought. Debugging had to be discovered. I can remember the exact instant when I realized that a large part of my life from then on was going to be spent in finding mistakes in my own programs.

Maurice Wilkes discovers debugging, 1949