CS100
Introduction to Programming

Lecture 21 Debugging and Profiling

Today’s learning objectives

- Understanding errors
- GBD
- Time profiling
- Memory checking with valgrind

Today’s learning objectives

- Understanding errors
- GBD
- Time profiling
- Memory checking with valgrind

Understanding Errors

```
hw2.c:87:7: error: ‘foo’ undeclared
```

Understanding Errors

$$\underbrace{\texttt{hw2.c}}_{\text{file in which error occurs}}\texttt{:87:7: error: ‘foo’ undeclared}$$

Understanding Errors

```
hw2.c:87:7: error: ‘foo’ undeclared
```

$\underbrace{\texttt{hw2.c}}_{\text{file in which error occurs}}\texttt{:}\underset{\overset{\uparrow}{\text{line number}}}{\texttt{87}}\texttt{:7: error: ‘foo’ undeclared}$

Understanding Errors

$$\underbrace{\texttt{hw2.c}}_{\text{file in which error occurs}}\texttt{:}\underset{\substack{\uparrow \\ \text{line number}}}{\texttt{87}}\texttt{:}\underset{\substack{\nwarrow \\ \text{character number}}}{\texttt{7}}\texttt{: error: `foo' undeclared}$$

Understanding Errors

```
hw2.c:87:7: error: ‘foo’ undeclared
```

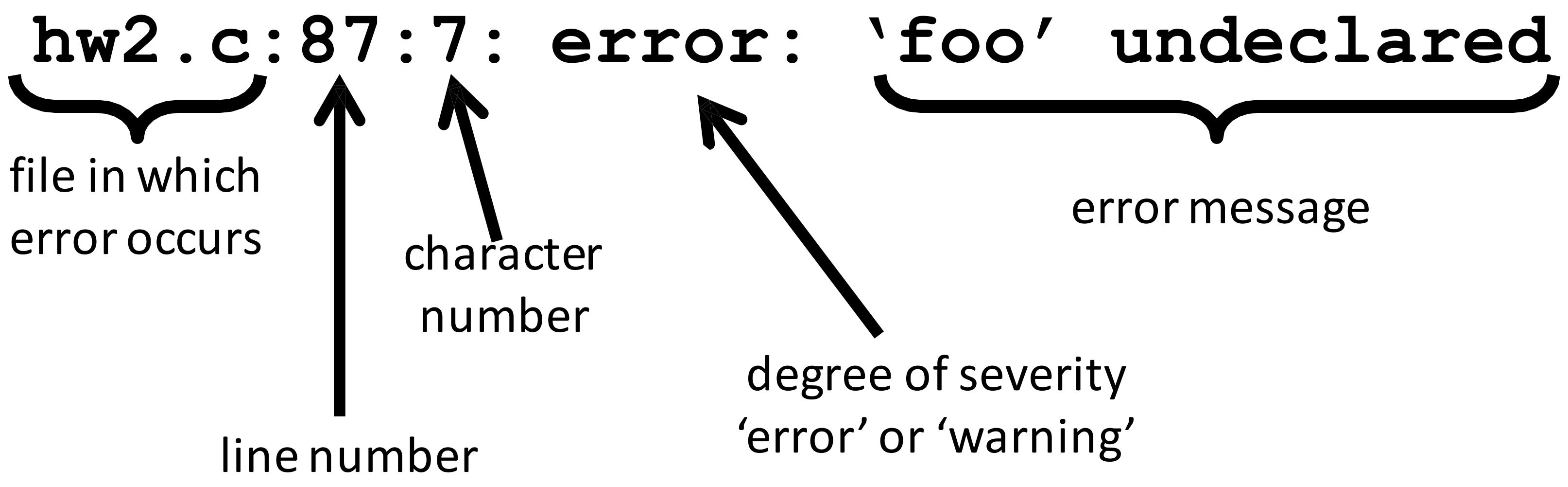
file in which error occurs

character number

degree of severity ‘error’ or ‘warning’

line number

Understanding Errors



```
hw2.c:87:7: error: ‘foo’ undeclared
```

file in which error occurs
line number
character number
degree of severity ‘error’ or ‘warning’
error message

#1 Rule of Debugging

- start with the **very first** error or warning
- recompile every time an error is fixed
  - errors will cascade
  - and de-cascade when fixed!

Cascading Errors

```
int numStudnts;
for (i = 0; i < numStudents; i++) {
  total += grades[i];
}
avg = total/numStudents;
```

Cascading Errors

```
int numStudnts;
for (i = 0; i < numStudents; i++) {
   total += grades[i];
}
avg = total/numStudents;
```

```
> gcc -Wall average.c
```

Cascading Errors

```
int numStudnts;
for (i = 0; i < numStudents; i++) {
   total += grades[i];
}
avg = total/numStudents;
```

```
> gcc -Wall average.c
```

- the `-Wall` flag shows all of warnings

Cascading Errors

```
int numStudnts;
for (i = 0; i < numStudents; i++) {
  total += grades[i];
}
avg = total/numStudents;
```

```
> gcc -Wall average.c
average.c:5:5: warning: unused variable ‘numStudnts’
average.c:22:17: error: ‘numStudents’ undeclared
average.c:25:13: error: ‘numStudents’ undeclared
```

Cascading Errors

```
int numStudnts;
for (i = 0; i < numStudents; i++) {
  total += grades[i];
}
avg = total/numStudents;
```

```
> gcc -Wall average.c
average.c:5:5: warning: unused variable 'numStudnts'
[redacted]
```

Cascading Errors

```
int numStudnts;
for (i = 0; i < numStudents; i++) {
   total += grades[i];
}
avg = total/numStudents;
```

```
> gcc -Wall average.c
average.c:5:5: warning: unused variable 'numStudnts'
average.c:22:17: error: 'numStudents' undeclared
average.c:25:13: error: 'numStudents' undeclared
```

Cascading Errors

```
int numStudents;
for (i = 0; i < numStudents; i++) {
  total += grades[i];
}
avg = total/numStudents;
```

Cascading Errors

```
int numStudents;
for (i = 0; i < numStudents; i++) {
  total += grades[i];
}
avg = total/numStudents;
```

```
> gcc -Wall average.c
```

Cascading Errors

```
int numStudents;
for (i = 0; i < numStudents; i++) {
  total += grades[i];
}
avg = total/numStudents;
```

```
> gcc -Wall average.c
```

- got rid of all 3 errors!

When Errors Occur

- compile time
  - pretty easy (normally typos or simple mistakes)
- linking
  - slightly harder (could be easy, could require rethinking how your code is laid out)
- run time
  - often difficult to pinpoint, and sometimes hard to spot at all
  - best bet is to use a debugger

Common Compiler Errors

```
hw2.c:87:7: error: ‘foo’ undeclared
```

- if `foo` is a **variable**:
  - forgot to declare
  - misspelled (on declaration or on use)
- if `foo` is a **function**:
  - forgot to `#include` file containing the prototype
  - misspelled (on declaration or on use)

Common Compiler Errors

```
hw2.c:37:6: warning: unused variable
    ‘bar’
```

- variable was declared but not used
  - normally because variable declaration has a typo
  - if you’re in the midst of writing code, this warning may be *temporarily* acceptable
    - haven’t had a chance to use the variable yet

Common Compiler Errors

```
hw2.c:54: warning: suggest
    parentheses around assignment
    used as truth value
```

- often a mistake inside a control statement
  - you meant to use **==** not **=**
  - (you want equivalency, not assignment)

Common Compiler Errors

```
hw2.c: 51: error: expected ‘;’
   before ‘for’
```

- missing semicolon on <u>previous</u> line of code
- ‘for’ is simply the word directly following the missing semicolon
  - could be ‘int’ or ‘if’ or a variable name, etc

Common Linker Errors

```
hw4.o: In function ‘main’:
hw4.c:91: undefined reference to ‘Fxn’
```

- linker can’t find code for ‘Fxn’ in any .o file
  - forgot to link `.o` file
  - misspelled named of Fxn
  - parameter list is different
    - differences between prototype/definition/call

Common Linker Errors

```
/usr/lib64/gcc/[...]/crt1.o: In function
   `_start':
/home/[...]/start.S:119: undefined
   reference to main
```

- you compiled a file that does not contain a **`main()`**
- without using the **`-c`** flag to indicate separate compilation

Error messages can be very long …

```
> gcc -Wall structs.c
In file included from /usr/include/stdio.h:33:0,
                 from structs.c:6:
/usr/lib64/gcc/x86_64-suse-linux/4.7/include/stddef.h:213:1: error:
expected '=', ',', ';', 'asm' or '__attribute__' before 'typedef'
In file included from /usr/include/stdio.h:74:0,
                 from structs.c:6:
/usr/include/libio.h:307:3: error: unknown type name 'size_t'
/usr/include/libio.h:311:67: error: 'size_t' undeclared here (not in a
function)
/usr/include/libio.h:339:62: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/libio.h:348:6: error: expected declaration specifiers or '...'
before 'size_t'
/usr/include/libio.h:470:19: error: expected '=', ',', ';', 'asm' or
'__attribute__' before '_IO_sgetn'
In file included from structs.c:6:0:
/usr/include/stdio.h:319:35: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/stdio.h:325:47: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/stdio.h:337:20: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/stdio.h:344:10: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/stdio.h:386:44: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/stdio.h:390:45: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/stdio.h:666:11: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/stdio.h:669:9: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/stdio.h:679:8: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/stdio.h:709:15: error: expected '=', ',', ';', 'asm' or
'__attribute__' before 'fread'
/usr/include/stdio.h:715:15: error: expected '=', ',', ';', 'asm' or
'__attribute__' before 'fwrite'
/usr/include/stdio.h:737:15: error: expected '=', ',', ';', 'asm' or
'__attribute__' before 'fread_unlocked'
/usr/include/stdio.h:739:15: error: expected '=', ',', ';', 'asm' or
'__attribute__' before 'fwrite_unlocked'
In file included from structs.c:9:0:
/usr/include/string.h:43:8: error: expected declaration specifiers or '...'
before 'size_t'
/usr/include/string.h:46:56: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/string.h:55:18: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/string.h:62:42: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/string.h:65:56: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/string.h:92:48: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/string.h:129:39: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/string.h:137:9: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/string.h:143:57: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/string.h:150:15: error: expected '=', ',', ';', 'asm' or
'__attribute__' before 'strxfrm'
In file included from structs.c:9:0:
/usr/include/string.h:165:15: error: expected '=', ',', ';', 'asm' or
'__attribute__' before 'strxfrm_l'
/usr/include/string.h:180:45: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/string.h:281:15: error: expected '=', ',', ';', 'asm' or
'__attribute__' before 'strcspn'
/usr/include/string.h:285:15: error: expected '=', ',', ';', 'asm' or
'__attribute__' before 'strspn'
/usr/include/string.h:395:15: error: expected '=', ',', ';', 'asm' or
'__attribute__' before 'strlen'
/usr/include/string.h:402:15: error: expected '=', ',', ';', 'asm' or
'__attribute__' before 'strnlen'
/usr/include/string.h:423:12: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/string.h:447:33: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/string.h:451:53: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/string.h:455:31: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/string.h:458:54: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/string.h:536:61: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/string.h:573:34: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/string.h:576:39: error: expected declaration specifiers or
'...' before 'size_t'
In file included from structs.c:11:0:
/usr/include/stdlib.h:139:15: error: expected '=', ',', ';', 'asm' or
'__attribute__' before '__ctype_get_mb_cur_max'
In file included from structs.c:11:0:
/usr/include/stdlib.h:331:4: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/stdlib.h:361:4: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/stdlib.h:465:22: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/stdlib.h:467:22: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/stdlib.h:467:38: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/stdlib.h:479:36: error: expected declaration specifiers or
'...' before 'size_t'
In file included from /usr/include/stdlib.h:491:0,
                 from structs.c:11:
/usr/include/alloca.h:32:22: error: expected declaration specifiers or
'...' before 'size_t'
In file included from structs.c:11:0:
/usr/include/stdlib.h:497:22: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/stdlib.h:502:45: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/stdlib.h:502:65: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/stdlib.h:755:9: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/stdlib.h:755:25: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/stdlib.h:760:34: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/stdlib.h:760:50: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/stdlib.h:839:6: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/stdlib.h:842:6: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/stdlib.h:846:31: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/stdlib.h:850:31: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/stdlib.h:859:36: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/stdlib.h:863:34: error: expected declaration specifiers or
'...' before 'size_t'
/usr/include/stdlib.h:870:15: error: expected '=', ',', ';', 'asm' or
'__attribute__' before 'mbstowcs'
/usr/include/stdlib.h:873:15: error: expected '=', ',', ';', 'asm' or
'__attribute__' before 'wcstombs'
```

… but not too hard to fix

- Follow the message til the original calling point
  - …
  - In file included from …
  - …
  - In file included from …
  - …
  - Instantiated here …
  - …
  - Instantiated here …
  - Error message

Debugging Basics

- if the error’s not clear from just looking at the code, you can try:

- inserting probe statements with printf
  - (but adding a printf might change your error!)
- rubber duck debugging
- googling the error message
- using a debugger

Today's learning objectives

- Understanding errors
- Basic use of GBD
- Time profiling
- Memory checking with valgrind

Debuggers

- see what is going on “inside” the program
  - more powerful and accurate than printf() probes

- examine individual variables (value & address)
  - can change variable’s value on the fly

- step through code line by line
  - can skip blocks of code you don’t want to see

Using GDB

- must use the '`-g`' flag when compiling
  - Done if using DEBUG mode in cmake!

- open program for testing using command line:

```
gdb hw2
```

- GDB – Gnu Project Debugger (text based)
  - "Standard debugger" on *nix systems

Using GDB

- GDB allows you to:
  - add breakpoints to stop the program at specific points (i.e. program lines)
  - use ‘print’ or ‘display’ to show values (or addresses) of variables
  - step through code line by line

GDB example

- Consider the following code to compute the factorial (test_gdb.cpp in cs100classexamples):

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

int main(){
  int i, num, j;
  std::cout << "Enter the number: ";
  std::cin >> num;
  for (i=1; i<num; i++)
    j=j*i;
  std::cout << "The factorial of " << num << " is " << j << "\n";
  return 0;
}
```

Error 1: j is not initialized

Error 2: we don’t include num in the factorial expression

Arbitrary result, i.e.

```
Enter the number: 3
The factorial of 3 is -1466591984
```

GDB example

- Start example with gdb (supposing it has been compiled in DEBUG mode):

```
gdb test_gdb
```

- Set a break point. Syntax:

```
break line_number
```

- Alternatives:

```
break [file_name]:line_number
break [file_name]:func_name
```

GDB example

- Break-points cause the program to interrupt at the specified place (line, function)
- Now run start the program:

```
run
```

- Program will interrupt with message:

```
Breakpoint 1, main () at test_gdb.cpp:10
10   j=j*i;
```

GDB example

- Analysing variable values with print instructions:

```
print {variable}
```

- Examples:

```
print i
print j
print num
```

- Alternative:

```
p {variable}
```

GDB example

- Example output:

```
(gdb) p i
$1 = 1
(gdb) p j
$2 = 3042592
(gdb) p num
$3 = 3 (gdb)
```

Wrong initial value

Continue and Stepping over/in

- `c` or `continue`: Debugger will continue executing until the next break point

- `n` or `next`: Debugger will execute the next line as single instruction

- `s` or `step`: Same as next, but does not treats function as a single instruction, instead goes into the function and executes it line by line

Further commands

- `l` or `list`: visualize the code

  l {line_number}

  l {function_name}

- `Enter`: Repeat the same command (i.e. stepping)

- `bt`: backtrace (print call-chain upon crash)

- `quit`: Quite the debugger

GDB 7.0

- Excellent debugger
- Let’s you step backwards instead of forward!
- Note:
    - Many code editors let us easily interact with GDB, and visually define break points

Today’s learning objectives

- Understanding errors
- Basic use of GBD
- Time profiling
- Memory checking with valgrind

Time profiling

- Measure the time it takes for different sections of the code
- Use:

```
#include <chrono>
```

- Example:

```
std::chrono::high_resolution_clock::time_point now =
      std::chrono::high_resolution_clock::now();
```

Time profiling

- How to measure time durations?
  - Take the difference of two time instants!

```
std::chrono::high_resolution_clock::time_point now =
      std::chrono::high_resolution_clock::now();
… //do something
std::chrono::high_resolution_clock::time_point later =
      std::chrono::high_resolution_clock::now();
double duration = (double)
      std::chrono::duration_cast<std::chrono::microseconds>(
      later-now).count();
```

LapTimer

- Consider class LapTimer provided in http://gitlab.com/laurentkneip/cs100classexamples

- Idea:
  - Configure named “laps”
  - Stop the time for each lap with using convenient interface

LapTimer

- Interface:

```
void start();
void stop( bool restart = false );
```

Basically a lap index for fast look-up.
Added class for type-safety

- Generating new, named laps, and using them

```
LapHandle addLap( const std::string & lapName );
void start( LapHandle & lap );
void stop( LapHandle & nextLap, bool restart = false );
```

LapTimer

- Reporting times:

```
void printSummary();
```

- Example output:

```
Module    | It.     | Total  [s]| / It. [s]| %
__________________________________________________
lap1      | 100     | 1.19898    | 0.01199   | 38.59%
lap2      | 100     | 0.61013    | 0.00610   | 19.64%
lap3      | 1000    | 1.29773    | 0.00130   | 41.77%

Total time consumption: 3.10685
```

Critical points when timing sections

- Beware of very short section
  - Example:

```
std::list<double> myList;
timer.start(lap);
myList.push_back(0.0);
timer.stop();
```

- Time duration smaller than resolution of clock!

Critical points when timing sections

- Use multiple iterations!

```
void stop( size_t iterations, bool restart = false );
void stop( LapHandle & nextLap,
           size_t iterations,
           bool restart = false );
```

  - Example:

```
std::list<double> myList;
timer.start(lap);
for (int i = 0; i < 1000; i++)
  myList.push_back(0.0);
timer.stop(1000);
```

Today’s learning objectives

- Understanding errors
- Basic use of GBD
- Time profiling
- Memory checking with valgrind

Valgrind

- Provides a number of debugging and profiling tools
- Most popular: mem-check
  - Checks memory-related errors such as leaks
- Usage:
  - For a program usually run by as `myprog args`, run `valgrind --leak-check=yes myprog args`
  - Runs valgrind with check for memory leaks
  - Need to have compiled with option `–g`

Valgrind example

- Take the following example:
  (test_valgrind.cpp in cs100classexamples)

```
#include <stdlib.h>

void f(void){
   int* x = (int*) malloc(10 * sizeof(int));
   x[10] = 0; // problem 1: heap block overrun
}             // problem 2: memory leak -- x not freed

int main(void){
   f();
   return 0;
}
```

Reading Valgrind output

Process ID

```
==60941== Memcheck, a memory error detector
==60941== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==60941== Using Valgrind-3.14.0 and LibVEX; rerun with -h for copyright info
==60941== Command: ./test_valgrind
==60941==
--60941-- run: /usr/bin/dsymutil "./test_valgrind"
==60941== Invalid write of size 4
==60941==    at 0x100000F4C: f() (test_valgrind.cpp:6)
==60941==    by 0x100000F73: main (test_valgrind.cpp:11)
==60941==  Address 0x100801578 is 0 bytes after a block of size 40 alloc'd
==60941==    at 0x100008041: malloc (in /usr/local/Cellar/valgrind/3.14.0/lib/valgrind/vgpreload_memc
==60941==    by 0x100000F43: f() (test_valgrind.cpp:5)
==60941==    by 0x100000F73: main (test_valgrind.cpp:11)
==60941==
==60941==
```

Upfront info about author and copyright

The command we ran

The command valgrind ran

Type of the error (access of unallocated memory)

Reading Valgrind output

```
==60941== Memcheck, a memory error detector
==60941== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==60941== Using Valgrind-3.14.0 and LibVEX; rerun with -h for copyright info
==60941== Command: ./test_valgrind
==60941==
--60941-- run: /usr/bin/dsymutil "./test_valgrind"
==60941== Invalid write of size 4
==60941==    at 0x100000F4C: f() (test_valgrind.cpp:6)
==60941==    by 0x100000F73: main (test_valgrind.cpp:11)
==60941==  Address 0x100801578 is 0 bytes after a block of size 40 alloc'd
==60941==    at 0x100008041: malloc (in /usr/local/Cellar/valgrind/3.14.0/lib/valgrind/vgpreload_memc
==60941==    by 0x100000F43: f() (test_valgrind.cpp:5)
==60941==    by 0x100000F73: main (test_valgrind.cpp:11)
==60941==
==60941==
```

Where did the error occur?
(call chain)

Reading Valgrind output

```
==60941== Memcheck, a memory error detector
==60941== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==60941== Using Valgrind-3.14.0 and LibVEX; rerun with -h for copyright info
==60941== Command: ./test_valgrind
==60941==
--60941-- run: /usr/bin/dsymutil "./test_valgrind"
==60941== Invalid write of size 4
==60941==    at 0x100000F4C: f() (test_valgrind.cpp:6)
==60941==    by 0x100000F73: main (test_valgrind.cpp:11)
==60941==  Address 0x100801578 is 0 bytes after a block of size 40 alloc'd
==60941==    at 0x100008041: malloc (in /usr/local/Cellar/valgrind/3.14.0/lib/valgrind/vgpreload_memc
==60941==    by 0x100000F43: f() (test_valgrind.cpp:5)
==60941==    by 0x100000F73: main (test_valgrind.cpp:11)
==60941==
==60941==
```

Auxiliairy information
(relative location with respect to allocated memory)

Reading Valgrind output (continued)

```
==60941== HEAP SUMMARY:
==60941==     in use at exit: 34,907 bytes in 430 blocks
==60941==   total heap usage: 509 allocs, 79 frees, 41,067 bytes allocated
==60941==
==60941== 40 bytes in 1 blocks are definitely lost in loss record 29 of 82
==60941==    at 0x100008041: malloc (in /usr/local/Cellar/valgrind/3.14.0/lib/valgrind/vgpreload_memche
==60941==    by 0x100000F43: f() (test_valgrind.cpp:5)
==60941==    by 0x100000F73: main (test_valgrind.cpp:11)
==60941==
==60941== LEAK SUMMARY:
==60941==    definitely lost: 40 bytes in 1 blocks
==60941==    indirectly lost: 0 bytes in 0 blocks
==60941==      possibly lost: 0 bytes in 0 blocks
==60941==    still reachable: 0 bytes in 0 blocks
==60941==         suppressed: 34,867 bytes in 429 blocks
==60941==
==60941== For counts of detected and suppressed errors, rerun with: -v
==60941== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 17 from 17)
```

Summary of heap state

Memory leak:
-Address of allocating statement
-Call chain

Memory leak summary