

Advanced Topics in C/C++

Lecturer: Duc Dung Nguyen, PhD. Contact: nddung@hcmut.edu.vn

Faculty of Computer Science and Engineering Hochiminh city University of Technology

Contents



- 1. Valgrind
- 2. Logging
- 3. Profiling





• What is valgrind?



- · What is valgrind?
- Tracking memory issues
 - A tool that detects memory leaks and memory errors
 - Bugs come from mismanagement of memory: allocating the wrong size, using an uninitialized pointer, accessing memory after it was freed, overrunning a buffer, etc.
 - Debugging memory, sometimes, is not easy with a normal debugger.



- What is valgrind?
- · Tracking memory issues
 - A tool that detects memory leaks and memory errors
 - Bugs come from mismanagement of memory: allocating the wrong size, using an uninitialized pointer, accessing memory after it was freed, overrunning a buffer, etc.
 - Debugging memory, sometimes, is not easy with a normal debugger.
- Syntax:
 - \$ valgrind <your program> [program params]



- What is valgrind?
- Tracking memory issues
 - A tool that detects memory leaks and memory errors
 - Bugs come from mismanagement of memory: allocating the wrong size, using an uninitialized pointer, accessing memory after it was freed, overrunning a buffer, etc.
 - Debugging memory, sometimes, is not easy with a normal debugger.
- Syntax:
 - \$ valgrind <your program> [program params]
- Example:
 - \$ valgrind ./memoryLeak red blue

Memory Errors Vs. Memory Leaks



- Memory leaks: dynamically allocates memory and forgets to later free it.
 - Generally won't cause a program to misbehave, crash, or give wrong answers
 - · Not an urgent situation!

Memory Errors Vs. Memory Leaks



- · Memory leaks: dynamically allocates memory and forgets to later free it.
 - Generally won't cause a program to misbehave, crash, or give wrong answers
 - Not an urgent situation!
- Memory error: reading uninitialized memory, writing past the end of a piece of memory, accessing freed memory, and other memory errors can have significant consequences.
 - · Should never be treated casually or ignored
 - · Memory errors are the primary concern

Valgrind's output



• Valgrind will print a summary of its memory usage.

· Lucky you: no errors and no leaks.

Valgrind's output



- Valgrind will print a summary of its memory usage.
- If all goes well:

```
==4649== ERROR SUMMARY: 0 errors from 0 contexts

==4649== malloc/free: in use at exit: 0 bytes in 0 blocks.

==4649== malloc/free: 10 allocs, 10 frees, 2640 bytes allocated.

==4649== For counts of detected errors, rerun with: -v

==4649== All heap blocks were freed -- no leaks are possible.
```

· Lucky you: no errors and no leaks.

Valgrind's output



· Report of an error:

```
==4651== Invalid write of size 1
==4651== at 0x80486A4: main (myprogram.c:58)
==4651== Address 0x4449054 is not stack'd, malloc'd or (recently) free'd
==4651==
==4651== ERROR SUMMARY: 1 errors from 1 contexts
==4651== malloc/free: in use at exit: 0 bytes in 0 blocks.
==4651== malloc/free: 1 allocs, 1 frees, 10 bytes allocated.
==4651== For counts of detected errors, rerun with: -v
==4651== All heap blocks were freed – no leaks are possible.
```

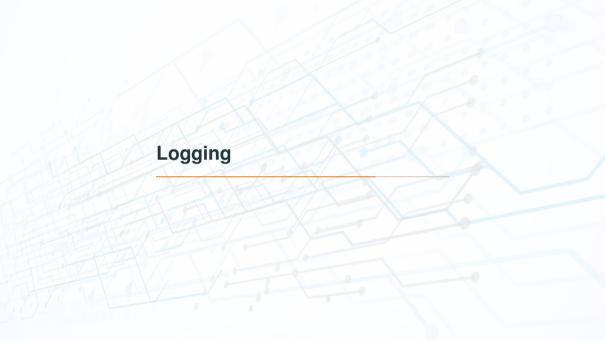
· And the code:

```
57     char *copy = malloc(strlen(buffer));
58     strcpy(copy, buffer);
```

Notes



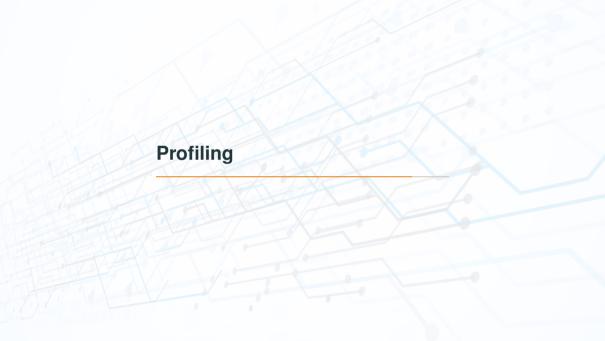
- Build valgrind from source can be very tricky on OSX
- A good tool to assist your program
- Never forget your manual: https://valgrind.org/docs/manual/manual.html



Logging



- Logging is a part of the debugging process
- You must utilize your output devices: cerr (2) for this task
- Macro is your friend
- If you can, go for Boost/log framework



Profiling



- Purpose: to measure the performance of your program, in terms of run-time, memory consumption, etc.
- · Tools are available and they depend on your task