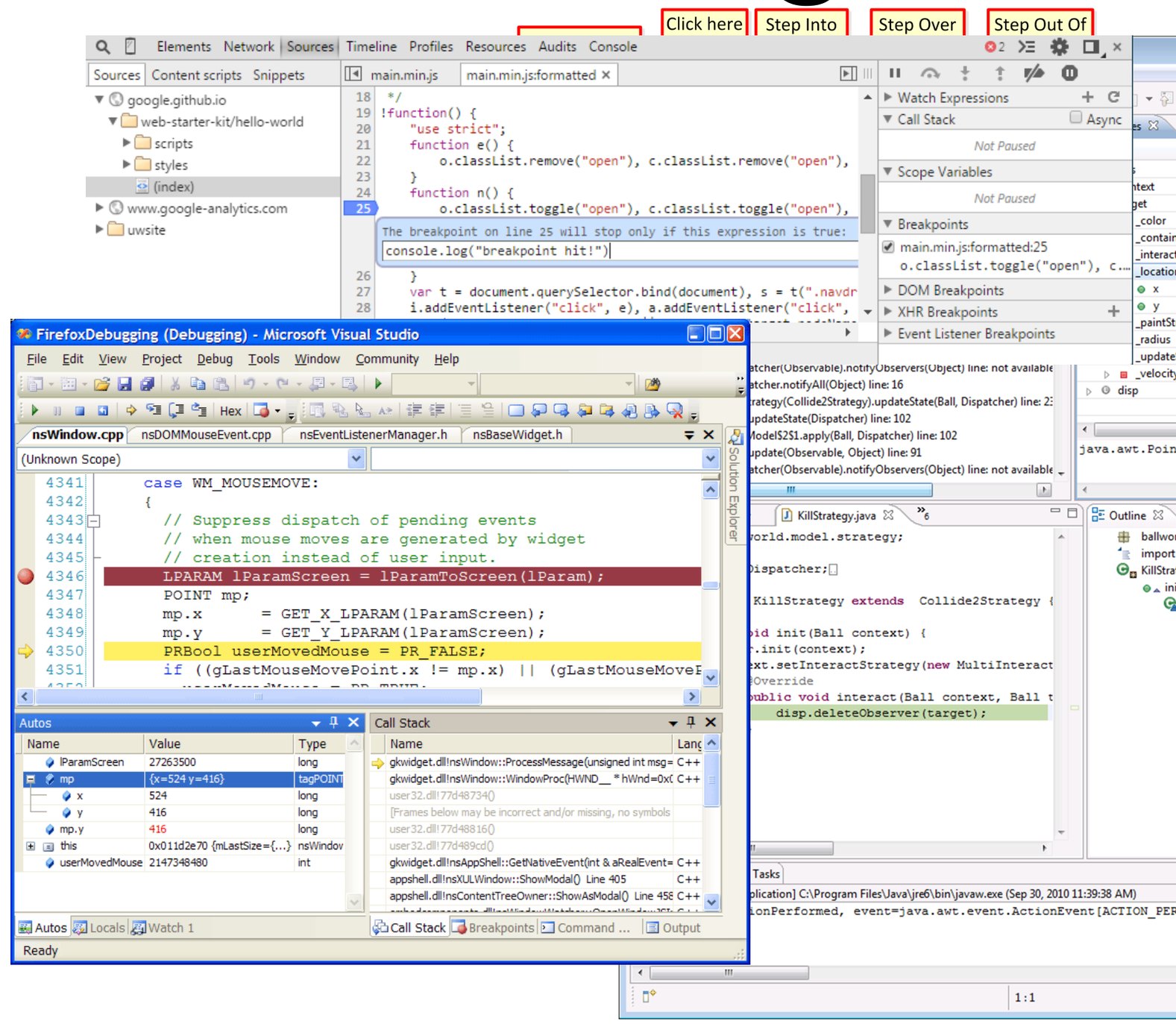
# Debugging & LLDB

Mistral Contrastin

Slides at madgen/debugging

# Why bother learning?

- It's everywhere!
- Makes you more productive
- Teaches you internals
- Fun (once you get used to it)




scs.senecac.on.ca/~david.humphrey/writing/images/full-debug-view.png
developer.chrome.com/devtools/images/js-debugging.png
www.clear.rice.edu/comp310/Eclipse/debug_screen_annotated.png

# LLDB vs printf

http://www.freestockphotos.biz/stockphoto/9343

# LLDB vs printf

https://upload.wikimedia.org/wikipedia/commons/f/fb/Improved_M249_Machine_Gun.jpg

# LLDB vs printf

https://www.flickr.com/photos/fatty/2661653476

https://www.cl.cam.ac.uk/~acr31/acr31-photo.jpg

# LLDB vs GDB?

- Stable
- Consistent
- Good thread support
- BSD-style licence
- Apple backed
- Better plugin architecture
- Less of a mess in the source

- Portable
- GPL
- Its own scripting lang.

# What won’t be covered?

# General advice

Always compile with:

`-Wall -Wextra -pedantic -std=x`

While developing/debugging:

`-O0`[1] `-g3 -Werror`

[1]If you're interested in this check out John Hennessy's "Symbolic Debugging of Optimized Code"

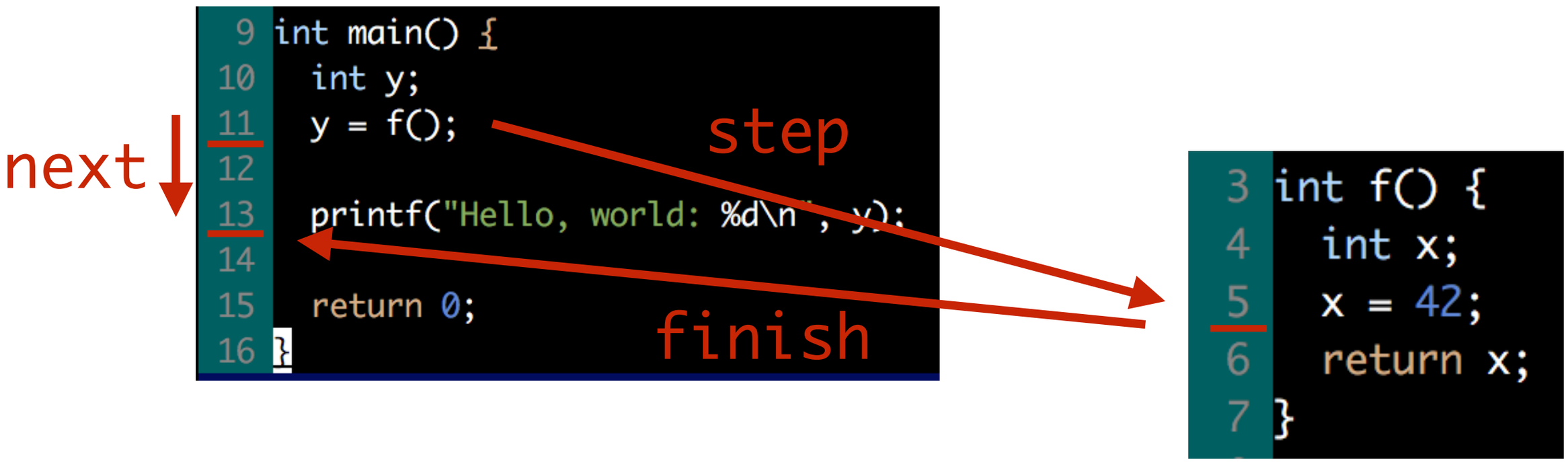
# Recap of 2nd year: Breakpoints

```
➜ recap lldb hello_world
(lldb) target create "hello_world"
Current executable set to 'hello_world' (x86_64).
(lldb) breakpoint set --name main
Breakpoint 1: where = hello_world`main + 15 at hello_world.c:11, address = 0x0000000100000f5f
(lldb) run
Process 38044 launched: '/Users/mistralcontrastin/Documents/debugging_pres/material/recap/hello_wor
ld' (x86_64)
Process 38044 stopped
* thread #1: tid = 0x417f3, 0x0000000100000f5f hello_world`main + 15 at hello_world.c:11, queue = '
com.apple.main-thread', stop reason = breakpoint 1.1
    frame #0: 0x0000000100000f5f hello_world`main + 15 at hello_world.c:11
   8
   9    int main() {
   10     int y;
-> 11     y = f();
   12     printf("Hello, world: %d\n", y);
   13
   14     return 0;
```

Alternatively,

- `breakpoint set -file hello_world.c -line 11`
- `b main`

# Recap of 2nd year: Stepping




`step`/`next`/`finish` = `thread step-in`/`step-over`/`step-out`

- `s` and `n` are short forms (`f` is not)
- There are `stepi` (`si`), `nexti` (`ni`) for instructions

# Backtrace my old friend

Search | Console | Call Hierarchy

Java Stack Trace Console

```
java.lang.NoClassDefFoundError: com/cordys/gmf/GMFTestBase$1
        at com.cordys.gmf.GMFTestBase.getDesigner(GMFTestBase.java:209)
        at com.cordys.gmf.genericTests.ZoomTest.genericTest(ZoomTest.java:19)
        at com.cordys.gmf.GMFGenericTestBase.cmTest(GMFGenericTestBase.java:44)
        at com.cordys.cm.uiunit.junit4.runners.UIUnitScreenDumperStatement.evaluate(UIUnitScreenDumperStatement.java:31)
        at com.cordys.cm.uiunit.junit4.runners.UIUnitTimeoutStatement$StatementThread.run(UIUnitTimeoutStatement.java:91)
Caused by: java.lang.ClassNotFoundException: com.cordys.gmf.GMFTestBase$1
        at java.net.URLClassLoader$1.run(URLClassLoader.java:366)
        at java.net.URLClassLoader$1.run(URLClassLoader.java:355)
        at java.security.AccessController.doPrivileged(Native Method)
        at java.net.URLClassLoader.findClass(URLClassLoader.java:354)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:423)
        at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:308)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:356)
```

wiki.jenkins-ci.org/download/attachments/72777810/eclipse_console.png

VS

```
[1]    44733 segmentation fault  ./packet
```

?????

`thread backtrace` (`bt`) to get a useful stack trace

“The unexamined life is not worth living.” — Socrates

See all variables in current stack frame:

`frame variable` (`fr v`)

Examine a particular variable:

`frame variable foo` (`fr v foo`)

Examine variable in different hex/ascii/binary/etc.

`frame variable --format hex bar`

Experiment with expressions:

`expr --format ascii -- 0x42`

# Changing context

Threads:

```
thread list
thread select x
thread …
```

Stack frames:

```
frame list
frame select x
frame …
```

Global:

```
target list
target select x
target …
```

# Focusing on interesting bits

*Watchpoints* to observe variables

`watchpoint set variable foo` (`w s v foo`)

*Conditional* inspection

`breakpoint set -n foo --condition ‘expr’`

# Text User Interface

```
| LLDB (F1) | Target (F2) | Process (F3) | Thread (F4) | View (F5) | Help (F6) |
<Sources>                                                  <Threads>
conditional`main                                           ♦─process 53219
14    return j * j;
15  }
16
17  int main(void) {
18    int i, acc;
19
20    for (i = 0, acc = 0; i < 200; i++)
21  ♦   acc += square(i);            <<< Thread 1: one-shot breakpoint 24
22
23    printf("\n%d\n", acc);
24
25    return 0;
26  }
<Variables>
(int) i = 23
(int) acc = 3795
Process: 53219    stopped          Thread: 0x87ee1    Frame:   0  PC = 0x0000000100000f4a
```

Using command **`gui`**

# Debugging outside the debugger

Good for perturbation problem and memory forensics[1]

[1]If you're interested in this check out "The art of memory forensics" by Ligh et al.

# Debugging outside the debugger

## Attaching processes locally

- From within:
  `process attach --pid 1234`
  `process detach`
- While starting up:
  `$ lldb -p 1234`

## Debugging remotely

- Run `lldb-server` in the server:
  `$ lldb-server platform --listen *:1234`
- Connect from within LLDB:
  `platform connect connect://host:1234`

# Automating tasks

- `command alias -- bpm b main`
- `command source file`
- `command script file`
- Bindings for Python and C++

# Traveling in Time

**GDB only (for now)**

Needed

## Slow

- Record & replay

```
record full
set record full 1234
record save/restore record.dat
```

- You can go reverse in time

```
reverse-continue (rc)
reverse-step
reverse-next
```

## Fast

- Checkout mozilla/rr

# What’s more?

- Manipulating var, environment, arguments, etc.
- Disassembly features
- Memory search
- Working with linker files
- Language specific commands
- Logging
- Customisation

- DTrace
- objdump

# Resources

- LLDB webpage
  lldb.llvm.org
- GDB manual (book really)
  sourceware.org/gdb/current/onlinedocs/gdb/

- CrackMe
  crackmes.de (temporarily down as of writing)
- CMU Bomb Lab
- Capture the flag
  overthewire.org/wargames/

Questions?