

# A Sampling of Other Things



# Today

- Profiling
- Static Analysis
- Developer Surveys

# Profiling

What is it?

What's it good for?

When will you use it

(other than 281?)

What are its limitations

# Getting started with `perf`

- Install the tool

```
sudo apt install linux-tools-common linux-tools-`uname -r`
```

# Getting started with `perf`

- Install the tool

```
sudo apt install linux-tools-common linux-tools-`uname -r`
```

- Write a simple program

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

- And profile it

```
$ make main  
cc    main.c  -o main  
$ perf record ./main  
$ ls  
main main.c perf.data  
$ perf report
```

# Getting something useful from `perf`

- Need a program that takes some time

```
void child() {
    int i;
    for (i=0; i < 0xFFFFFFFF; i++) { // 7 F's
        asm("nop;");
    }
}

int main() {
    int i;
    for (i=0; i < 0xFFFFFFFF; i++) { // 7 F's
        asm("nop;");
    }
    child();
}
```

```
$ make main
cc    main.c  -o main
$ perf record ./main
$ perf report
```

# Understanding a little how `perf` works

```
$ perf record -F1 ./main  
$ perf report
```

```
$ perf record -F100000 ./main  
$ perf report
```

- What does `-F` do?
  - (Try `man perf-report`, you can use `/` to search in `man`)

# Can we profile library code?

- Let's write a lot of 0's

```
#include <string.h>
...
for (i=0; i < 0xFFFFF; i++) { // 7 F's -> 5 F's
...
char buf[0xFFFF];
...
// asm("nop");
memset(buf, 0, 0xFFFF);
```



# Some libraries are uglier :(

- Add a `printf`

```
#include <stdio.h>
...
for (i=0; i < 0xFFFFF; i++) { // 7 F's -> 5 F's
...
// asm("nop");
printf("%d\n", i);
```

## This can make profiling real code hard

- Don't go down blind alleys (e.g. `perf annotate --stdio`)

## (5-10 min) Try it out

Pick any prior code you've written and try profiling it

```
$ perf record ./your_program  
$ perf report
```

> The bigger the better

Are the results what you expect?

# Closing thoughts on profiling

When should you profile your code?

How often should you profile your code?

# Closing thoughts on profiling

When should you profile your code?

How often should you profile your code?

---

Other questions, thoughts about profiling?

# Static Analysis

What is it?

Why is it useful?

When should you run it?

# Just a little history first

Linting - the original static analysis

# Just a little history first

## Linting - the original static analysis

- The point: The line between style and correctness is blurry

# Just a little history first

## Linting - the original static analysis

- The point: The line between style and correctness is blurry

Today, the lines between compilers, linters, and static analyzers are blurring



# Static Analysis in action: cppcheck

```
sudo apt install cppcheck
```

## Check a single file:

```
mmdarden@c4cs-w18:~/share/281$ cppcheck my_compress.cpp
Checking my_compress.cpp...
[my_compress.cpp:445]: (error) Memory leak: dict
Checking my_compress.cpp: DEBUG...
Checking my_compress.cpp: DEBUG2...
```

## Check a whole project for everything

```
mmdarden@c4cs-w18:~/share/281$ cppcheck --enable=all .
...
```

# Static Analysis in action: scan-build

```
sudo apt install clang
```

This tool dynamically re-writes make rules (!)

- Won't work if you've hardcoded `g++` (should be `$(CXX)`)

```
bad: bad.cpp  
    g++ bad.cpp
```

```
good: good.cpp  
     $(CXX) $(CPPFLAGS) $(CXXFLAGS) good.cpp
```

```
mmdarden@c4cs-w18:~/share/281$ scan-build make  
...  
scan-build: 7 bugs found.  
scan-build: Run 'scan-view /tmp/scan-build-2016-11-30' to examine bug reports
```

# (10 min) Try it out

Try running `cppcheck` and `scan-build` on an old project

```
$ cppcheck --enable=all .  
$ scan-build make
```

Did they find any errors?

Try running them on a current project

# Developer Surveys

C4CS

StackOverflow.com

# Next Week

Special topics lectures

Should be fun :)

---

**warning:** end-of-semester slightly shrinks the window to turn in Advanced Exercise 14

- Double check the OH on the course calendar!