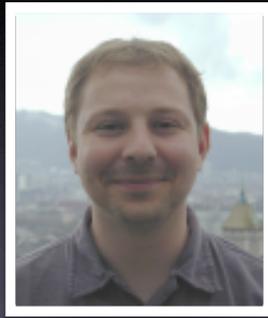


Andreas Leitner

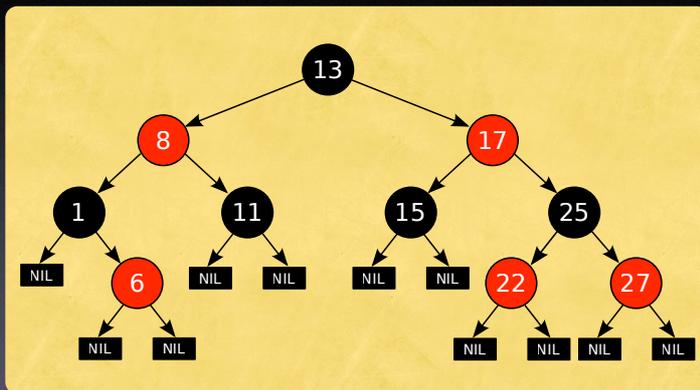


- “Reproducing Crashes” on Thursday

4

4

Complex Invariants



5

5

http://en.wikipedia.org/wiki/Red-black_tree

Complex Invariants

```
class RedBlackTree {  
    ...  
    boolean sane() {  
        assert (rootHasNoParent());  
        assert (rootIsBlack());  
        assert (redNodesHaveOnlyBlackChildren());  
        assert (equalNumberOfBlackNodesOnSubtrees());  
        assert (treeIsAcyclic());  
        assert (parentsAreConsistent());  
  
        return true;  
    }  
}
```

6

6

Memory Invariants

Even within a single process, some invariants must hold over the entire run

- code integrity
- data integrity

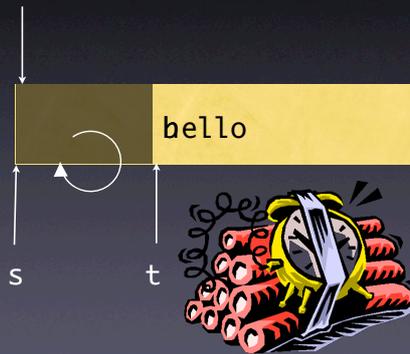
This is a major issue in C and C++

10

10

Heap Misuse

```
s = malloc(30)  free_list
free(s)
t = malloc(20)
strcpy(t, "hello")
s[10] = 'b'
free(s)
```



11

11

Heap Assertions

The *GNU C runtime library* provides a simple check against common errors:

```
$ MALLOC_CHECK_=2 myprogram myargs
free() called on area that was already free'd()
Aborted (core dumped)
$ _
```

12

12

Memory Assertions

The *Valgrind* tool checks *all* memory accesses:

```
$ valgrind sample 11 14
Invalid read of size 4
  at 0x804851F: shell_sort (sample.c:18)
  by 0x8048646: main (sample.c:35)
  by 0x40220A50: __libc_start_main (in /lib/libc.so)
  by 0x80483D0: (within /home/zeller/sample)
```

Valgrind works as an *interpreter* for x86 code

16

16

Valgrind Checks

- ▶ Use of uninitialized memory
- ▶ Accessing free'd memory
- ▶ Accessing memory beyond malloc'd block
- ▶ Accessing inappropriate stack areas
- ▶ Memory leaks: allocated area is not free'd
- ▶ Passing uninitialized memory to system calls

17

17

Shadow Memory

- V-bit set = corresponding bit is initialized
- A-bit set = corresponding byte is accessible



18

18

V-Bits

- When a bit is first written, its V-bit is set
- Simple read accesses to uninitialized memory do not result in warnings:

```
struct S { int x; char c; };  
struct S s1, s2;  
s1.x = 42; ← 5 bytes initialized  
s1.c = 'z'; ←  
s2 = s1; ← 8 bytes copied (no warning)
```

19

19

V-Bits Warnings

Reading uninitialized data causes a warning if

- a value is used to generate an *address*
- a *control flow decision* is to be made
- a value is passed to a *system call*

20

20

A-Bits

- When the program starts, all global data is marked “accessible” (= A-bits are set)
- malloc() sets A-bits for the area returned; free() clears them
- Local variables are “accessible” on entry and “non-accessible” on exit
- Accessing “non-accessible” data ⇒ error

21

21

Things to Check

- **Critical results.** If lives, health, or money depend on a result, it had better be checked.
- **External conditions.** Any conditions which are not within our control must be checked for integrity.

Points to Consider

- The more active assertions, the greater the chance to catch infections.
- The sooner a program fails, the easier it is to track the defect.
- Defects that escape into the field are the hardest to track.

More to Consider

- By default, failing assertions are not user-friendly.
 - ➡ *Handle assertions in a user-friendly way*
- Assertions impact performance.
 - ➡ *First measure; then turn off only the most time-consuming assertions*

Concepts

- ★ To check memory integrity, use specialized tools to detect errors at run time
- ★ Apply such tools before any other method
- ★ To fully prevent memory errors, use another language (or dialect, e.g. Cyclone)
- ★ Turning assertions off seldom justifies the risk of erroneous computation
