

# Automatic Compiler Testing

Christian Lindig – Dagstuhl



# Math Table

...

$$3 + 4 = 7$$

$$3 * 4 = 12$$

$$3 + 5 = 8$$

$$3 * 5 = 15$$

...



# Program

...

```
printf("3 + 4 = %d\n", 3+4);
```

```
printf("3 * 4 = %d\n", 3*4);
```

```
printf("3 + 5 = %d\n", 3+5);
```

```
printf("3 * 5 = %d\n", 3*5);
```

...

...

3 + 4 = 7

3 \* 4 = 12

3 + 5 = 8

3 \* 5 = 15

...



# Functions

```
void f(int x, int y)
{
    printf("%d + %d = %d\n", x, y, x+y);
    printf("%d * %d = %d\n", x, y, x*y);
}
```

...

f(3,4);

f(3,5);

...

...

3 + 4 = 7

3 \* 4 = 12

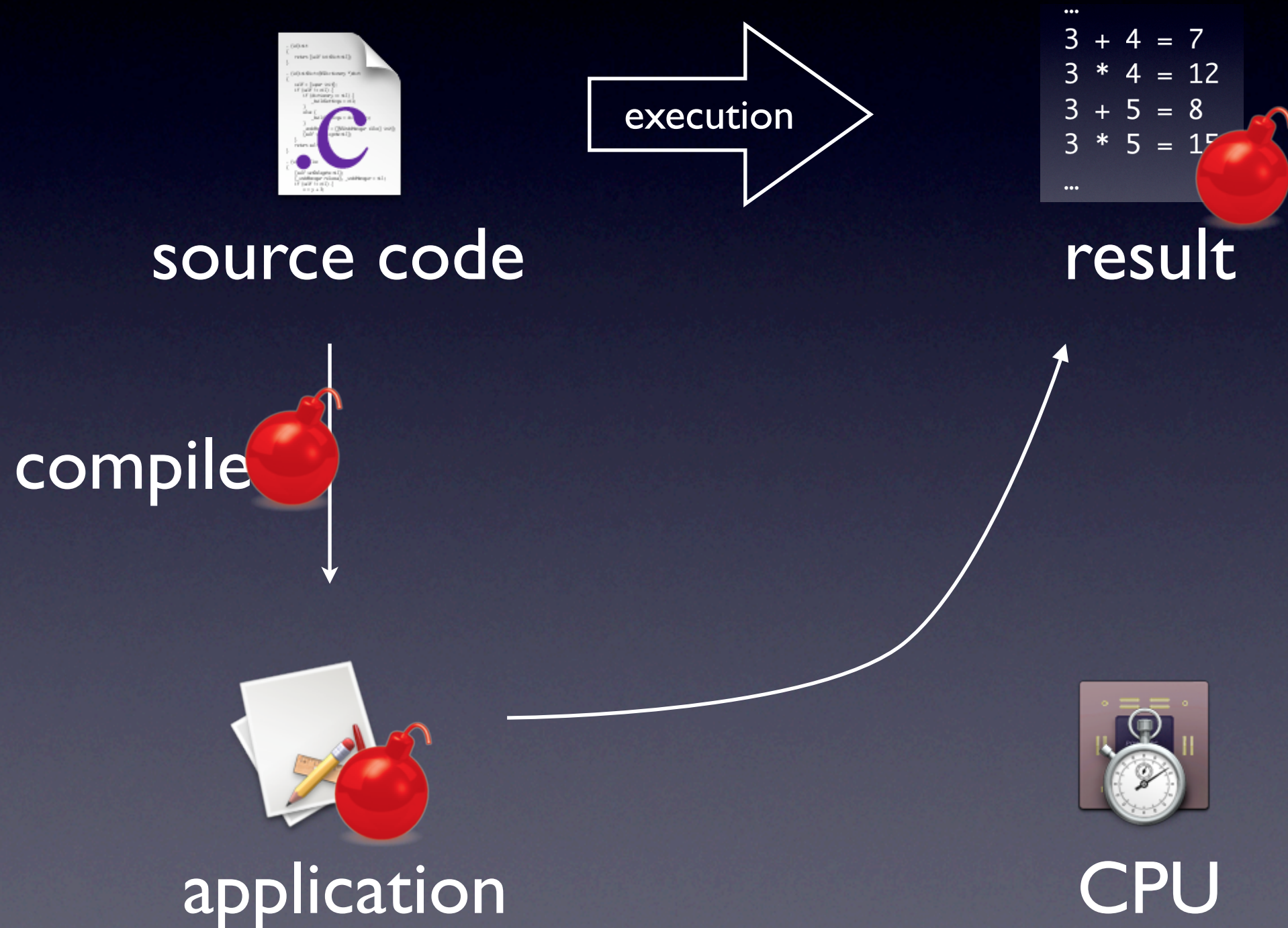
3 + 5 = 8

3 \* 5 = 15

...



# Compilation





# Compiler Bugs

- A compiler bug affects all programs compiled with it
- Operating systems (Linux, Windows, MacOS) are compiled from C
- Resulting programs in embedded systems (car, cell phone, ...) cannot be fixed
- Compiler bugs are expensive



# Self-Checking Programs

```
void f(int i)
{
    assert(i == 1234);
}
```

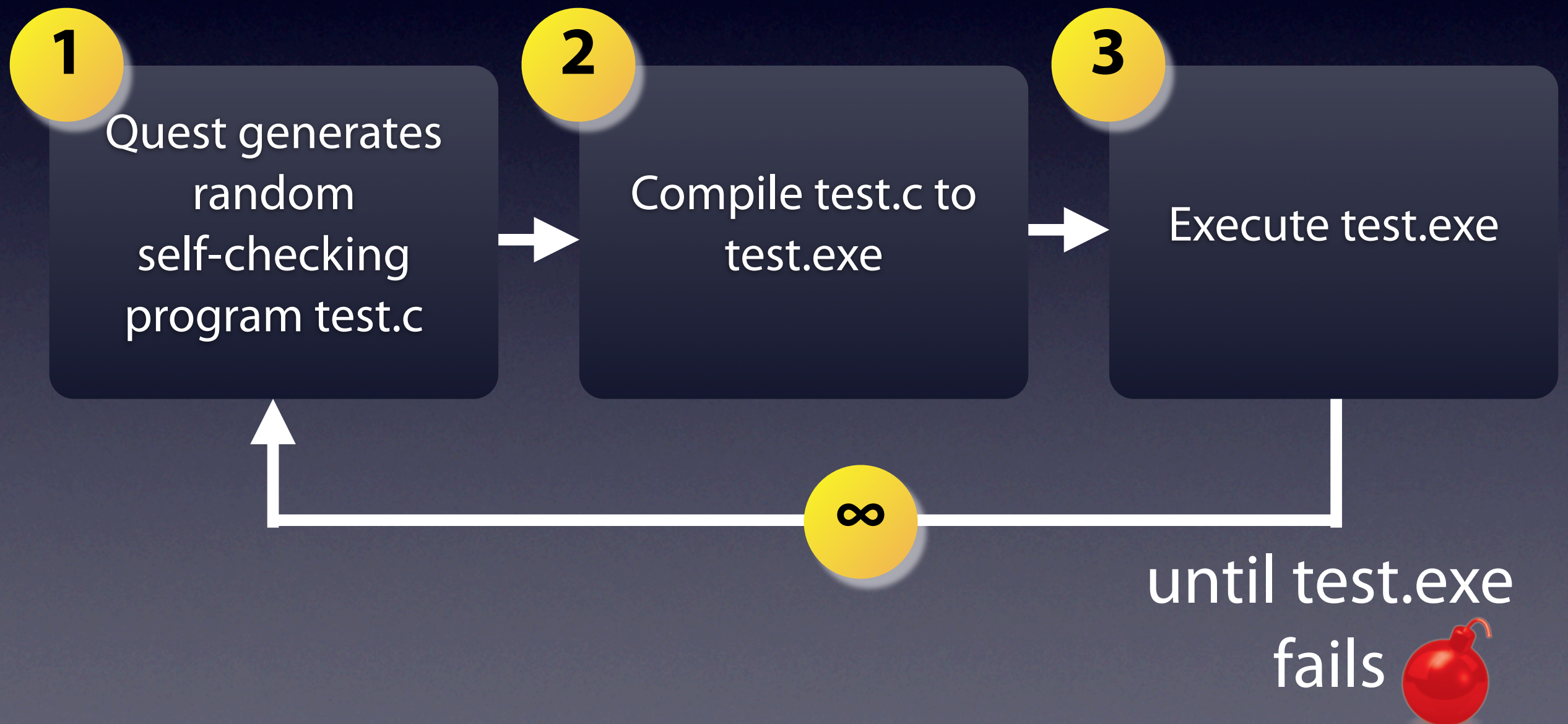
***i** is checked to contain  
**1234**; programs halts  
otherwise*

```
void main()
{
    f(1234);
}
```

***1234** is passed to f*



# Random Testing





# GNU Compiler Bug #18742

```
struct A {long long  a1; int  a2;}    aa = { 1L , 2 };
struct B {double     b1; int  b2;}    bb = { 3.0, 4 };
struct C {short int  c1; }           cc = { 5      };
```

```
int main() {
    f(aa, bb, cc);
    return 0;
}
```

**fails**



```
void f(struct A a, ...) {
    struct B b;
    struct C c;
    va_list ap;
    va_start(ap, a);
    b = va_arg(ap, struct B);
    c = va_arg(ap, struct C);
    va_end(ap);
    assert(c.c1 == cc.c1);
}
```



# 14 Bugs found with Quest

	IRIX	SunOS 5	Linux	MacOS
compiler	MIPS	SPARC	x86	PPC
MipsPro	1			
PathCC			3	
LCC 4.2			1	
ICC 8.1			1	
GCC 2.x		1	1	
GCC 3.x	1		2	1
GCC 4.x				2

All C compilers are mature and used for production

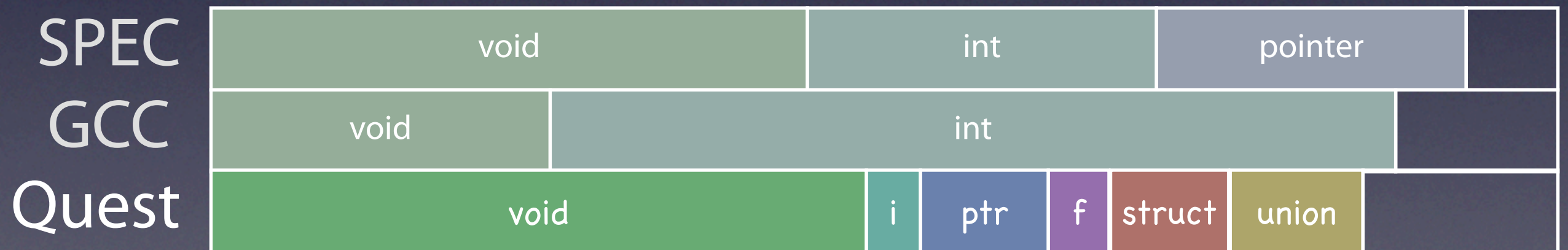


# Compiler Test Suites

## Static Distribution of Argument Types



## Static Distribution of Return Types



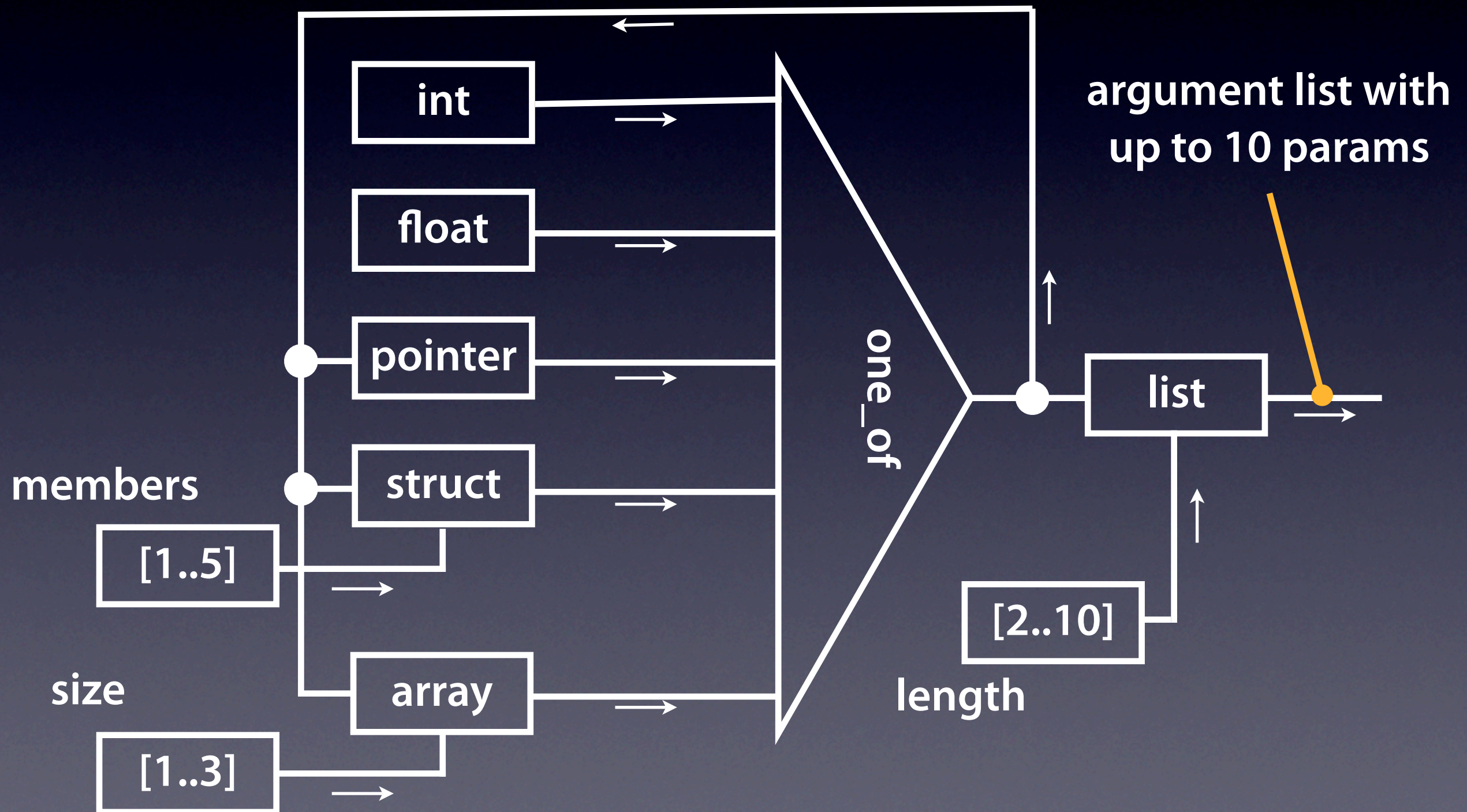


# Correctness—Why so Hard?

- informal specifications
- evolution of C vs. binary compatibility
- lack of orthogonality, many dark corners
- 2 implementations in compiler:  
call site and function definition out of sync
- optimization: PathCC has 300+ flags



# Quest Feature: Composable Randomness





# Future Work

Java - ddreplay-core/src/main/java/org/deltadebugging/ddreplay/framework/common/EventXmlFileWriter.java - Eclipse PL

AbstractEvent.java EventXmlFileWriter.java

```
openWriter();
}

public EventXmlFileWriter(File file) throws IOException {
    this.config = null;

    setXmlFile(file, true);

    openWriter();
}

protected void setXmlFile(File file, boolean createBackup) throws IOException {
    if (file.isDirectory()) {
        String msg = "File is existing directory: " + file;
        logger.warn(msg);
        throw new I
    }

    if (createBackup) {
        File bakFile
        FileUtils.c
        logger.warn
        FileUtils.f
        logger.debu
    }

    this.xmlFile
}
```

Introduce Parameter Object

New parameter object class

Class name: SetXmlFileParameter

Destination: ☒ Top level class ☐ Nested class in 'EventXmlFileWriter'

Select fields for parameter object class:

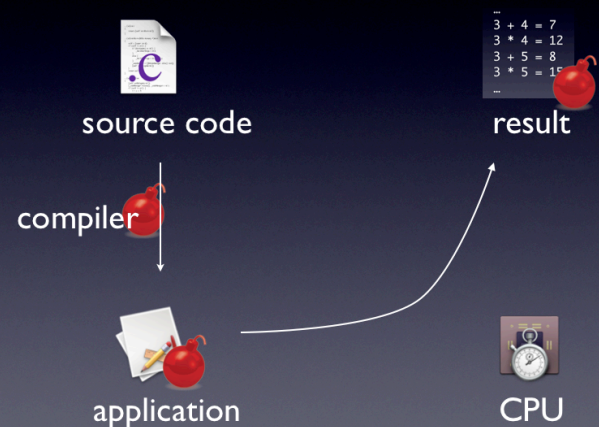
	Type	Name
<input checked="" type="checkbox"/>	File	file
<input checked="" type="checkbox"/>		

Up



# Summary

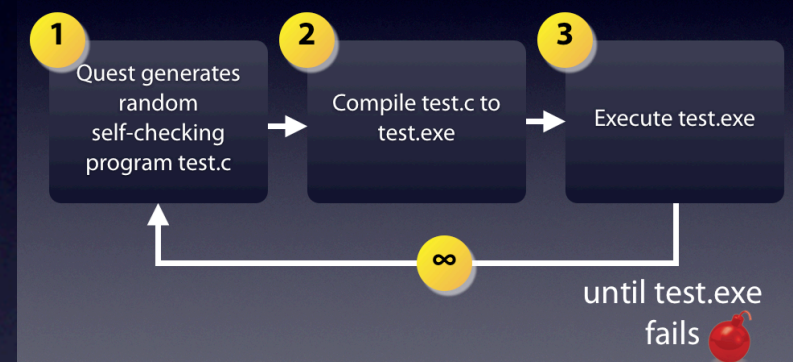
## Compilation



## Compiler Bugs

- A compiler bug affects all programs compiled with it
- Operating systems (Linux, Windows, MacOS) are compiled from C
- Resulting programs in embedded systems (car, cell phone, ...) cannot be fixed
- Compiler bugs are expensive

## Random Testing



## 14 Bugs found with Quest

compiler	IRIX MIPS	SunOS 5 SPARC	Linux x86	MacOS PPC
MipsPro	1			
PathCC			3	
LCC 4.2			1	
ICC 8.1			1	
GCC 2.x		1	1	
GCC 3.x	1		2	1
GCC 4.x				2

all bugs are related to var args or struct/unions

## Compiler Test Suites

Static Distribution of Argument Types									
SPEC	pointer				int			i,c	f,d
GCC	pointer	int					i,c		
Quest	pointer	int	i,c	float	doubl	union	struct		

Static Distribution of Return Types									
SPEC	void				int		pointer		
GCC	void	int							
Quest	void	i	ptr	f	struct	union			

quest-tester - Google Code

lindig@gmail.com | My Profile | Help | My Account | Sign out

Google Code

quest-tester

Automatically test calling conventions of C compilers

Project Home Downloads Wiki Issues Source Administer

Quest generates C code that can uncover bugs in a C compiler. The generated code passes complex arguments between functions and thus tests the translation of function calls. This part of a compiler is hard to test with real-world code because complex functions calls are rare. Not totally surprising, Quest uncovered bugs in production-quality compilers. As such, Quest is most interesting for compiler writers.

License: [New BSD License](#)

Labels: [OCaml](#), [callingconvention](#), [tester](#), [automatic](#), [quickcheck](#), [unix](#), [random](#), [lua](#), [ml](#), [Lindig](#), [compiler](#), [noweb](#), [C](#)

Project owners: [lindig](#)

Bugs Found in GCC with Quest

- [http://gcc.gnu.org/bugzilla/show\\_bug.cgi?id=18742](http://gcc.gnu.org/bugzilla/show_bug.cgi?id=18742)
- [http://gcc.gnu.org/bugzilla/show\\_bug.cgi?id=23324](http://gcc.gnu.org/bugzilla/show_bug.cgi?id=23324)
- [http://gcc.gnu.org/bugzilla/show\\_bug.cgi?id=16819](http://gcc.gnu.org/bugzilla/show_bug.cgi?id=16819)
- [http://gcc.gnu.org/bugzilla/show\\_bug.cgi?id=19268](http://gcc.gnu.org/bugzilla/show_bug.cgi?id=19268)
- [http://gcc.gnu.org/bugzilla/show\\_bug.cgi?id=30148](http://gcc.gnu.org/bugzilla/show_bug.cgi?id=30148)

Documentation

For additional documentation, please see the [Wiki](#) page; for information about installation see the [README](#). The [ManPage](#) has details about using Quest.

Help Wanted

<http://code.google.com/p/quest-tester/>