Better signal handling with Clang

Alpha Abdoulaye

EPITA Systems/Security Laboratory (LSE)

November 14, 2017

Introduction

Signals 101 (1/4)

```
static void handler(int signum)
{
    char test[] = "TEST\n";
    write(STDOUT_FILENO, test, sizeof(test));
}

int main(void)
{
    signal(SIGINT, handler);
    while (1) {
        printf("In loop\n");
        sleep(1);
    }
    return 0;
}
```

Signals 101 (2/4)

```c
int main(void)
{
    struct sigaction s;

    memset(&s, 0, sizeof(struct sigaction));
    s.sa_handler = &handler;
    if (sigaction(SIGINT, &s, NULL) == -1) {
        return 1;
    }

    while (1) {
        printf("In loop\n");
        sleep(1);
    }
    return 0;
}
```

Signals 101 (3/4)

$ ./test

```
In loop
In loop
In loop
^CTEST
In loop
In loop
^CTEST
In loop
^CTEST
In loop
[...]
```

Signals 101 (4/4)

- Asynchronous
- Multiple POSIX signals
- Customizable behaviour

Issues

Signal Safety

- Async-signal-safe functions
- *errno* integrity
- Reentrancy
- Deadlocks
- man **signal-safety(7)**

Workarounds (1/2)

```c
static void handle(int signum) {
  int saved = errno;
  while (waitpid(-1, 0, WNOHANG) > 0) {}
  errno = saved;
}

int main(void)
{
    signal(SIGCHLD, handler);

    // [...]

    while ((ret = read(fd, buf, size)) > 0) {
          // Do stuff...
    }
    if (ret == -1)
       perror("Error:")

    return 0;
}
```

Workarounds (2/2)

- Limit to trivial logic
- *sig_atomic_t*
- Try harder ?

Clang

Options

- Categorize
- Custom attributes
  - White list
  - Black list
- Diagnose errno handling

Clang AST

$ clang -Xclang -ast-dump -fsyntax-only test.c

```
FunctionDecl 0x55d62dfb4df8 [...] used handler 'void (int)' static
|-ParmVarDecl 0x55d62dfb4d70 <col:21, col:25> col:25 signum 'int'
`-CompoundStmt 0x55d62dfb5190 <line:13:1, line:19:1>
  |-DeclStmt 0x55d62dfb4fb8 <line:14:5, col:27>
  | `-VarDecl 0x55d62dfb4ee8 [...] used test 'char [6]' cinit
  |   `-StringLiteral [...] <col:19> 'char [6]' lvalue "TEST\n"
  `-CallExpr 0x55d62dfb5120 <line:15:5, col:44> 'ssize_t':'long'
    |-ImplicitCastExpr [...] 'ssize_t (*)(int, const void *, size_t)'
    | `-DeclRefExpr [...] 'write' 'ssize_t (int, const void *, size_t)'
    |-IntegerLiteral [...] </usr/include/unistd.h:214:23> 'int' 1
    |-ImplicitCastExpr [...] 'const void *' <BitCast>
    | `-ImplicitCastExpr [...] 'char *' <ArrayToPointerDecay>
    |   `-DeclRefExpr [...] lvalue Var [...] 'test' 'char [6]'
     `-UnaryExprOrTypeTraitExpr [...] 'unsigned long' sizeof
       `-ParenExpr 0x55d62dfb5068 <col:38, col:43> 'char [6]' lvalue
         `-DeclRefExpr [...] lvalue Var [...] 'test' 'char [6]'
```

Add new attributes

```
static int func(int stuff) __attribute__((sigunsafe))
{
    return stuff + 3;
}

static void handler(int signum) __attribute__((sighandler))
{
    char test[] = "TEST\n";
    write(STDOUT_FILENO, test, sizeof(test));
    func(24);
}
```

Your own AST traversal

```
class DiagnoseSigHandler
  : public RecursiveASTVisitor<DiagnoseSigHandler> {
  // [...]
  bool VisitCallExpr(CallExpr *C) {
    // Custom logic
    return true;
  }

  bool VisitStmt(Stmt *S) {
    // Awesome logic
    return true;
  }
};
```

Output Test

$ clang my_test.c -Wsignal-safe

```
./my_test.c:12:1: warning: function declared with
   'sighandler' attribute [-Wsignal-safe]
static void handler(int signum) __attribute__((sighandler))
^
./my_test.c:18:5: warning: function func is not signal safe
   [-Wsignal-safe]
    func(5);
    ^
```

Misc

- Generic categorization
- Cross Tranlation Unit
- No errno false positives

References

1 Clang
2 man signal(7)

Questions ?

- Contact : alpha@lse.epita.fr

Figure: Ah