cxx2rs

What?

Convert C headers to Rust (using Python)

Why?

Lazy

Safety

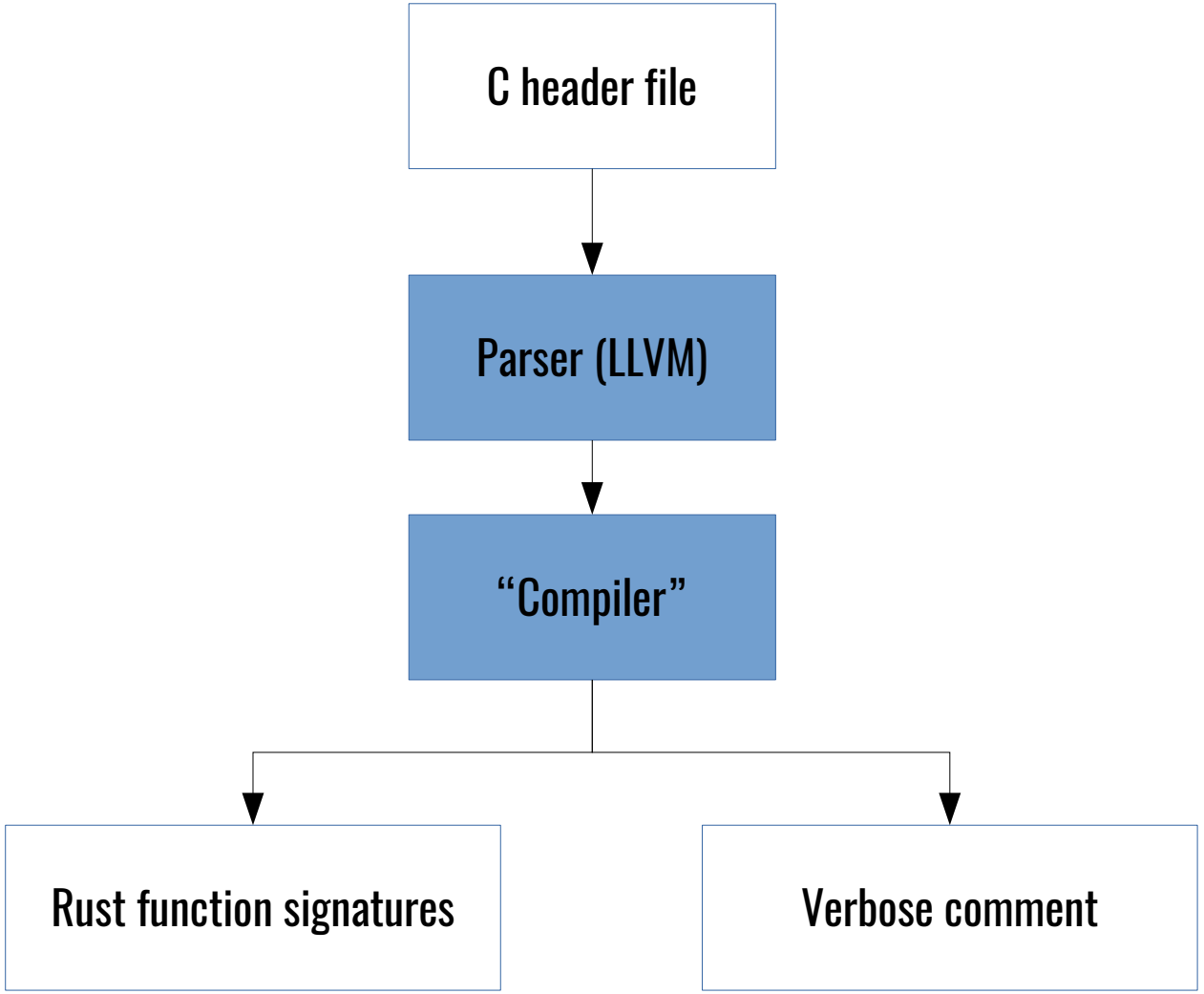
time_t time(time_t *tloc)

Incorrect stack layout modifies debug

```
extern time_t time(void);

int main(int argc, char* argv[]){
        uname(&node);
        before = time();

        […]
        if (debug == 0){
                close(0);
                close(1);
                close(2);
        }
        […]
        fd_source = open(argv[optind+1], O_RDONLY);
        fd_dest = creat(argv[optind+2], 0600);
        if ((fd_source < 0) || (fd_dest < 0))
                return 1;

        /* Serious stuff starts here... */
        handle_header();
        do_the_work();
        if (encipher)
                add_random_stuff(argv[optind+2]);

        […]

        debugger(&decipher, “decipher”);
        debugger(&encipher, “encipher”);
        debugger(&debug, “debug”);

        after = time();
        if ((debug == 1) && (encipher == 1))
                printf(”Node %s (%s) enciphered %d bytes with key %s in %ds.n”,
                        node.nodename, node.machine,
                        (int)global_size, argv[optind], (int)(after - before));
        return 0;
}
```

The Underhanded C Contest Winner 2007

www.underhanded-c.org


C header file
Parser (LLVM)
“Compiler”
Rust function signatures
Verbose comment

```
pip install cxx2rs
```

```
cxx2rs lzma /usr/include/lzma.h \
    `pkg-config --cflags-only-I liblzma`
    > lzma.rs
```

```
clang.cindex.LibclangError:libclang.so:
  cannot open shared object file:
    No such file or directory.
  To provide a path to libclang use
  Config.set_library_path() or
  Config.set_library_file().
```

```
LD_LIBRARY_PATH=/usr/lib/llvm-3.5/lib \
cxx2rs lzma /usr/include/lzma.h \
   `pkg-config --cflags-only-I liblzma`
   > lzma.rs
```

Conversion

C:

```
void foo(int some_arg);
```

Rust:

```
/*
void foo()
   (int) some_arg
*/

#[link(name="lzma")]   // gcc -llzma
extern "C" {
   pub fn foo(some_arg: libc::c_int);
}
```

`

```
/*
lzma_ret lzma_stream_buffer_decode() [unsigned int]
        (uint64_t *) memlimit [unsigned long *]
        (uint32_t) flags [unsigned int]
        (lzma_allocator *) allocator [lzma_allocator *]
        (const uint8_t *) in [const unsigned char *]
        (size_t *) in_pos [unsigned long *]
        (size_t) in_size [unsigned long]
        (uint8_t *) out [unsigned char *]
        (size_t *) out_pos [unsigned long *]
        (size_t) out_size [unsigned long]
*/
#[link(name="lzma")]
extern "C" {
        pub fn lzma_stream_buffer_decode(
            memlimit:  *mut libc::c_ulong,
            flags:     libc::c_uint,
            allocator: *mut lzma_allocator,
            in_:       *const libc::c_uchar,
            in_pos:    *mut libc::c_ulong,
            in_size:   libc::c_ulong,
            out:       *mut libc::c_uchar,
            out_pos:   *mut libc::c_ulong,
            out_size:  libc::c_ulong) -> libc::c_uint;
}
```

```
/*
lzma_ret lzma_stream_buffer_decode() [unsigned int]
        (uint64_t *) memlimit [unsigned long *]
        (uint32_t) flags [unsigned int]
        (lzma_allocator *) allocator [lzma_allocator *]
        (const uint8_t *) in [const unsigned char *]
        (size_t *) in_pos [unsigned long *]
        (size_t) in_size [unsigned long]
        (uint8_t *) out [unsigned char *]
        (size_t *) out_pos [unsigned long *]
        (size_t) out_size [unsigned long]
*/
#[link(name="lzma")]
extern "C" {
        pub fn lzma_stream_buffer_decode(
            memlimit:  *mut libc::c_ulong,
            flags:     libc::c_uint,
            allocator: *mut lzma_allocator,
            in_:       *const libc::c_uchar,
            in_pos:    *mut libc::c_ulong,
            in_size:   libc::c_ulong,
            out:       *mut libc::c_uchar,
            out_pos:   *mut libc::c_ulong,
            out_size:  libc::c_ulong) -> libc::c_uint;
}
```

C Structs

```
/*
struct lzma_filter
                (lzma_vli) id [unsigned long]
                (void *) options
*/

#[repr(C)]
pub struct lzma_filter {
        pub id:      libc::c_ulong,
        pub options: *mut libc::c_void,
}
```

#enum 1/2

```
enum GTraverseFlags
{
  G_TRAVERSE_LEAFS    = 1 << 0,
  G_TRAVERSE_NON_LEAFS  = 1 << 1,
  G_TRAVERSE_ALL  = G_TRAVERSE_LEAFS |
                                    G_TRAVERSE_NON_LEAFS,
  G_TRAVERSE_MASK = 0x03
};
```

#enum 2/2

```
/*
enum GTraverseFlags {
    G_TRAVERSE_LEAFS     =  0x00000001 (1)
    G_TRAVERSE_NON_LEAFS =  0x00000002 (2)
    G_TRAVERSE_ALL       =  0x00000003 (3)
    G_TRAVERSE_MASK      =  0x00000003 (3)
}
*/

bitflags! {
    flags GTraverseFlags: libc::c_uint {
        const G_TRAVERSE_LEAFS     =  1 as libc::c_uint,
        const G_TRAVERSE_NON_LEAFS =  2 as libc::c_uint,
        const G_TRAVERSE_ALL       =  3 as libc::c_uint,
        const G_TRAVERSE_MASK      =  3 as libc::c_uint,
    }
}
```

#define

C:

```
#define LZMA_STREAM_HEADER_SIZE 12 /**
 * \brief       Options for encoding/decoding Stream
Header and Stream Footer
 */
```

Rust:

```
/* LZMA_STREAM_HEADER_SIZE 12 /**
 * \brief       Options for encoding/decoding Stream
Header and Stream Footer
 */ */
pub const LZMA_STREAM_HEADER_SIZE: i32 = 12;
```

Keywords

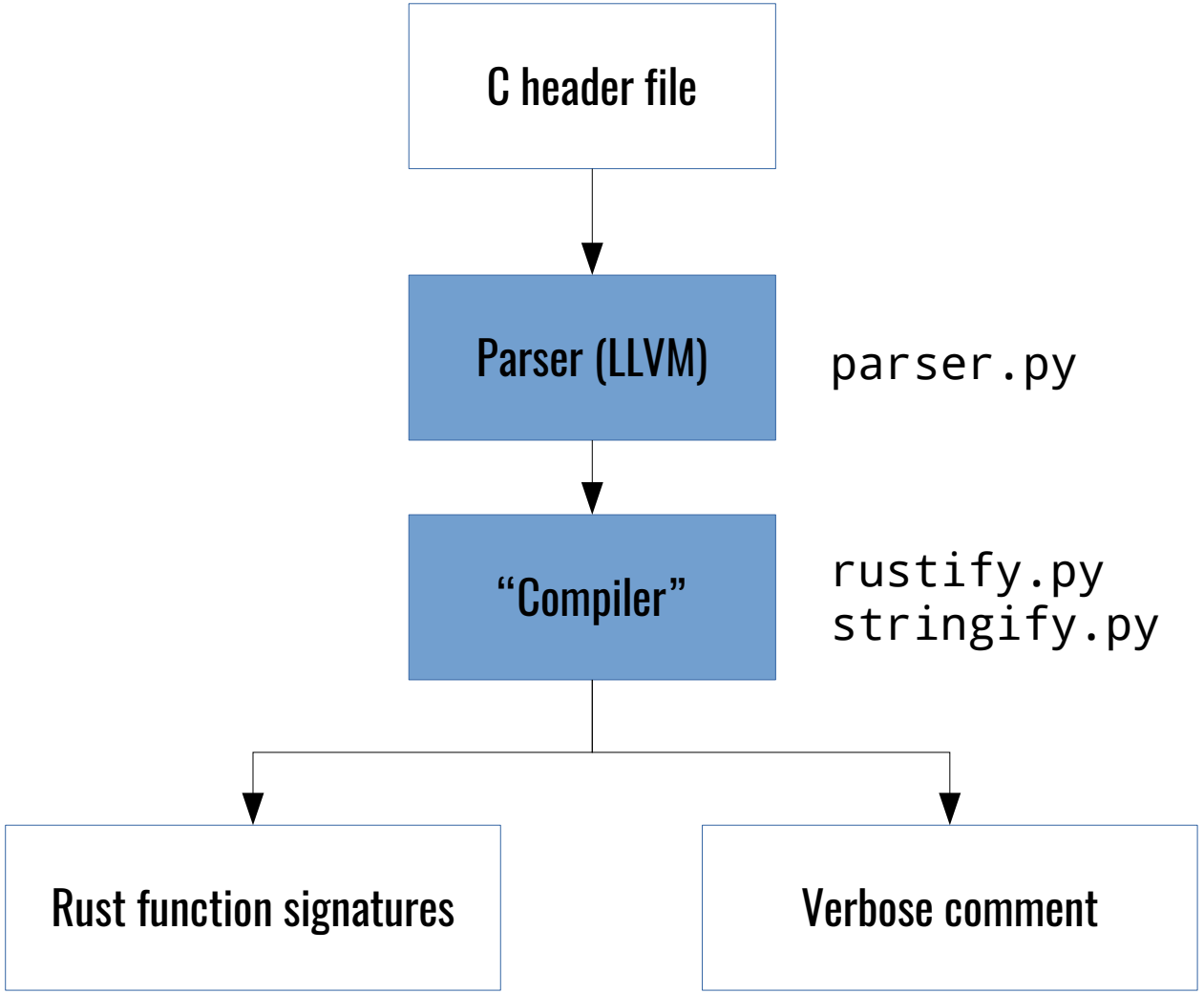
C:

```
void foo(int ref)
```

Rust:

```
extern "C" {
   pub fn foo(ref_: libc::c_int);
}
```

```
keywords = ['priv', 'loop', 'ref', 'in', 'type',
'where', 'impl', 'self', 'as', 'pub']
```


C header file
Parser (LLVM)
parser.py
“Compiler”
rustify.py
stringify.py
Rust function signatures
Verbose comment

Use and Contribute (GPLv2)

github.com/manuels/cxx2rs

pip install cxx2rs