Advanced Programming in the UNIX Environment

Week 05, Segment 3: Unix Development Tools: The Compiler Chain, Part I

Department of Computer Science
Stevens Institute of Technology

Jan Schaumann
jschauma@stevens.edu
https://stevens.netmeister.org/631/

Software Development Tools

The UNIX Userland is an IDE – essential tools that follow the paradigm of “Do one thing, and do it right” can be combined.

You are here.

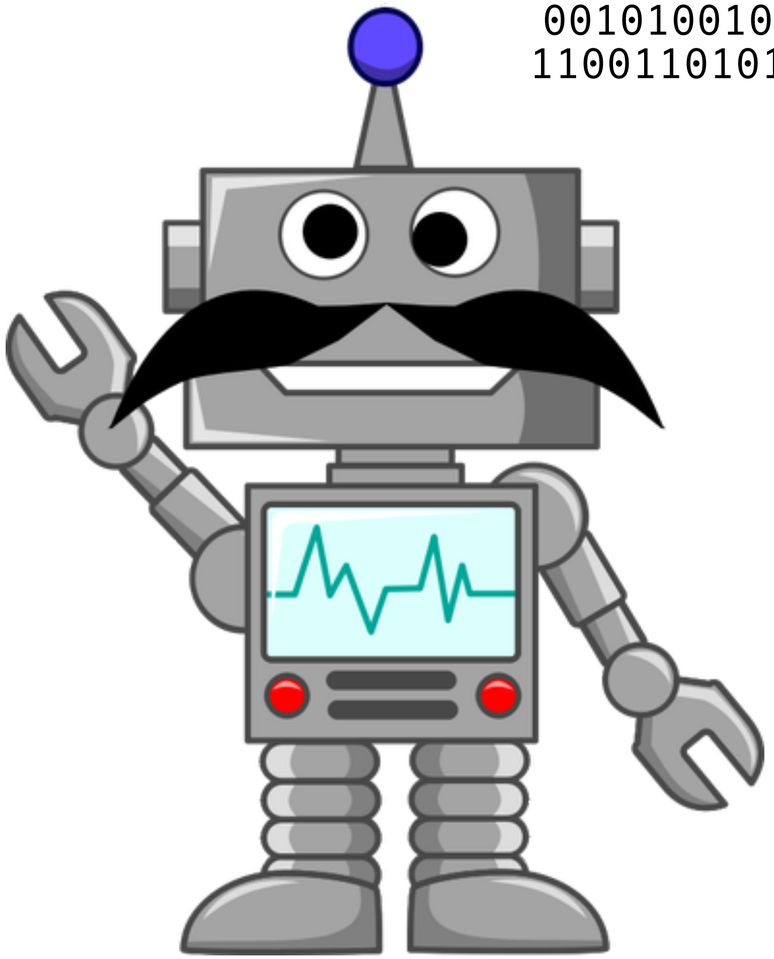
The most important tools are:

- `$EDITOR` ✅
- the compiler toolchain
- `gdb(1)` – debugging your code
- `make(1)` – project build management, maintain program dependencies
- `diff(1)` and `patch(1)` – report and apply differences between files
- `cvs(1)`, `svn(1)`, `git(1)` etc. – revision control, distributed project management

Compilers

```
#include <stdio.h>

int
main(int argc, char **argv) {
        printf("Hello, World!\n");
}
```



```
0010100101110011010111001101010110101101 0
1100110101101011001100101011011011010011
```

Compilers

A compiler translates *source code* from a high-level programming language into *machine code* for a given architecture by performing a number of steps:

- preprocessing

/usr/include/stdio.h

```
#include <stdio.h>
#define NUM 42

int
main(int argc, char **argv) {
        printf("%d\n", NUM);
}
```

```
...
void perror(const char *);
int printf(const char * __restrict, ...)
                __printflike(1, 2);
int putc(int, FILE *);
int putchar(int);
...

int
main(int argc, char **argv) {
        printf("%d\n", 42);
}
```

Compilers

A compiler translates *source code* from a high-level programming language into *machine code* for a given architecture by performing a number of steps:

- preprocessing
- lexical analysis

```
if (num > 42) msg = "Don't panic!";
```

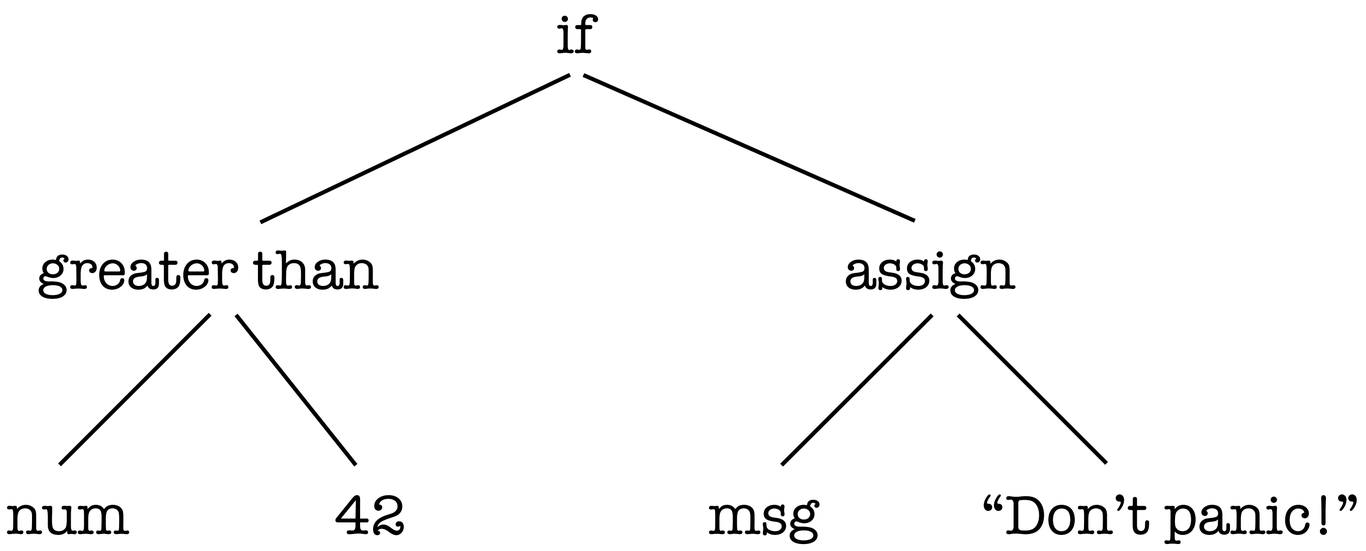
Keyword:	Token:	Identifier:	Operator:	Number:	Token:	Identifier:	Token:	String:	Token:
"if"	"("	"num"	">"	42	")"	"msg"	"="	"Don't panic!"	";"

Compilers

A compiler translates *source code* from a high-level programming language into *machine code* for a given architecture by performing a number of steps:

- preprocessing
- lexical analysis
- syntax analysis

```
if (num > 42) msg = "Don't panic!";
```

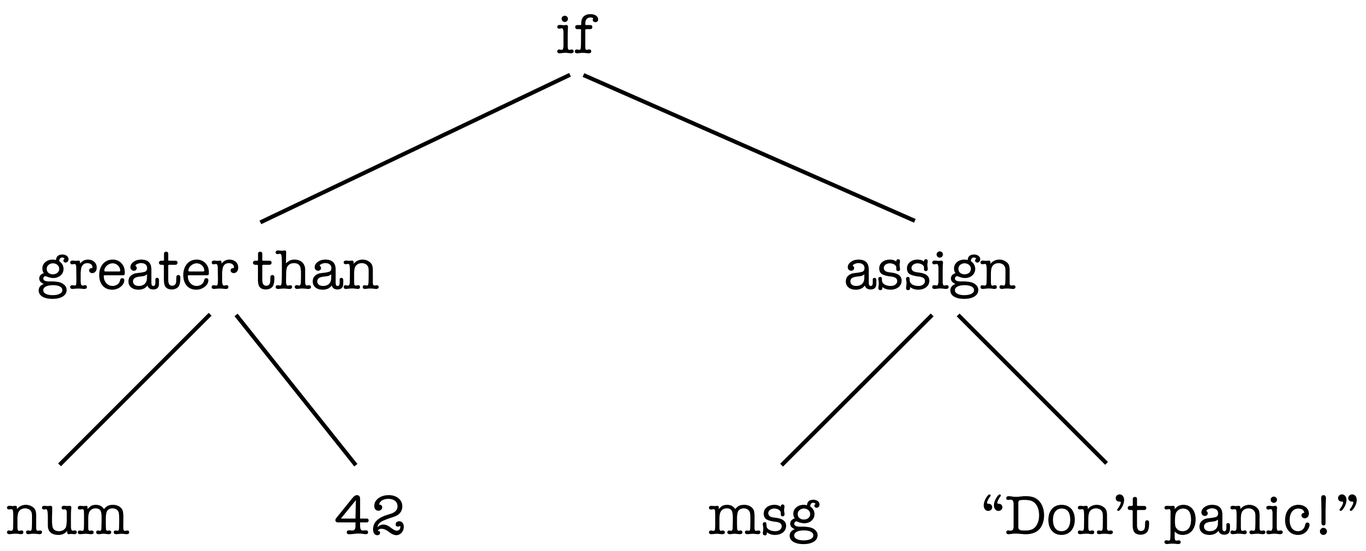


Compilers

A compiler translates *source code* from a high-level programming language into *machine code* for a given architecture by performing a number of steps:

- preprocessing
- lexical analysis
- syntax analysis
- semantic analysis

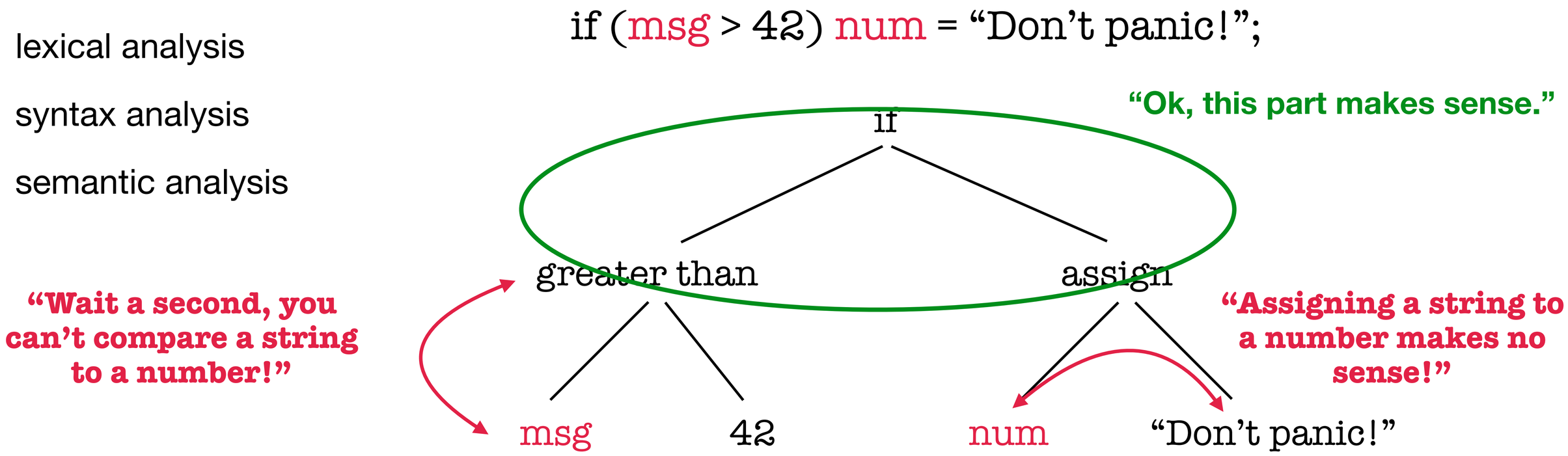
```
if (num > 42) msg = "Don't panic!";
```



Compilers

A compiler translates *source code* from a high-level programming language into *machine code* for a given architecture by performing a number of steps:

- preprocessing
- lexical analysis
- syntax analysis
- semantic analysis



Compilers

A compiler translates *source code* from a high-level programming language into *machine code* for a given architecture by performing a number of steps:

- preprocessing
- lexical analysis
- syntax analysis
- semantic analysis
- code generation
- code optimization

```
int i;

void func() {
    int j;
    j = 42;
    i = 5;
    i = 10;
    return;
    i = 20;
}
```

unused

redundant

→

dead code

```
int i;

void func() {
    i = 10;
    return;
}
```

Compilers

A compiler translates *source code* from a high-level programming language into *machine code* for a given architecture by performing a number of steps:

- preprocessing
- lexical analysis
- syntax analysis
- semantic analysis
- code generation
- code optimization

```
#include <stdio.h>

int
main(int argc, char **argv) {
    printf("Hello, World!\n");
}
```

→

```
        .file   "hw.c"
        .text
        .section        .rodata
.LC0:
        .string "Hello world"
        .text
        .globl  main
        .type   main, @function
main:
.LFB3:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movl    $.LC0, %edi
        call    puts
        movl    $0, %eax
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE3:
        .size   main, .-main
        .ident  "GCC: (nb4 20200810) 7.5.0"
```

Compilers

A compiler translates *source code* from a high-level programming language into *machine code* for a given architecture

- preprocessing
- lexical analysis
- syntax analysis
- semantic analysis
- code generation
- code optimization
- assembly

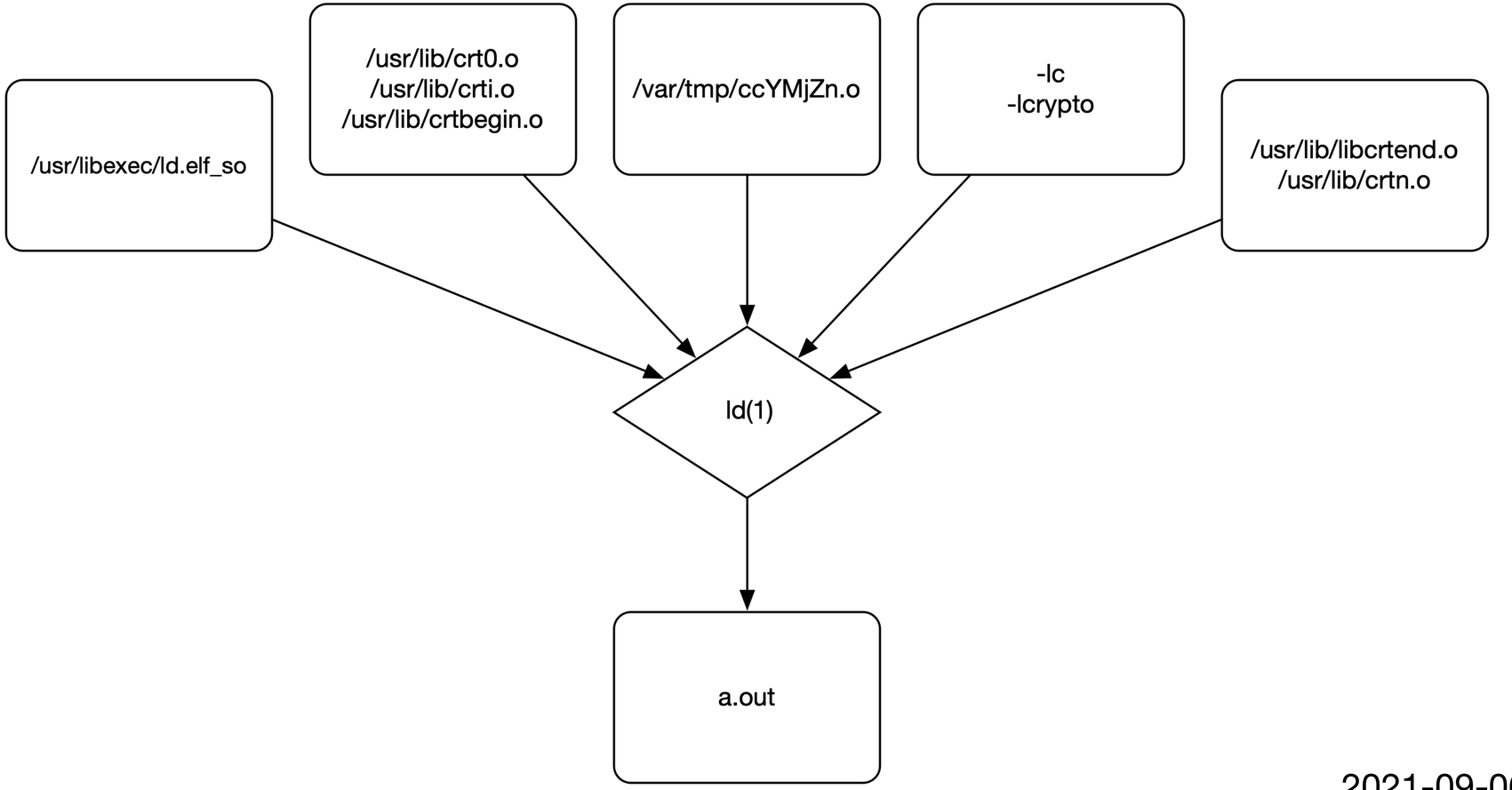
```
        .file   “hw.c"
        .text
        .section        .rodata
.LC0:
        .string "Hello world"
        .text
        .globl  main
        .type   main, @function
main:
.LFB3:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movl    $.LC0, %edi
        call    puts
        movl    $0, %eax
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE3:
        .size   main, .-main
```

```
00000000 7f 45 4c 46 02 01 01 00  | .ELF....|
00000010 00 00 00 00 00 00 00 00  | ........|
00000020 01 00 3e 00 01 00 00 00  | ..>.....|
00000030 00 00 00 00 00 00 00 00  | ........|
*
00000050 58 02 00 00 00 00 00 00  | X.......|
00000060 00 00 00 00 40 00 00 00  | ....@...|
00000070 00 00 40 00 0c 00 0b 00  | ..@.....|
00000100 55 48 89 e5 bf 00 00 00  | UH......|
00000110 00 e8 00 00 00 00 b8 00  | ........|
00000120 00 00 00 5d c3 48 65 6c  | ...].Hel|
00000130 6c 6f 20 77 6f 72 6c 64  | lo world|
00000140 00 00 47 43 43 3a 20 28  | ..GCC: (|
00000150 6e 62 34 20 32 30 32 30  | nb4 2020|
00000160 30 38 31 30 29 20 37 2e  | 0810) 7.|
00000170 35 2e 30 00 00 00 00 00  | 5.0.....|
00000200 14 00 00 00 00 00 00 00  | ........|
00000210 01 7a 52 00 01 78 10 01  | .zR..x..|
00000220 1b 0c 07 08 90 01 00 00  | ........|
00000230 1c 00 00 00 1c 00 00 00  | ........|
00000240 00 00 00 00 15 00 00 00  | ........|
00000250 00 41 0e 10 86 02 43 0d  | .A....C.|
00000260 06 50 0c 07 08 00 00 00  | .P......|
00000270 00 00 00 00 00 00 00 00  | ........|
*
00000320 01 00 00 00 04 00 f1 ff  | ........|
```

Compilers

A compiler translates *source code* from a high-level programming language into *machine code* for a given architecture by performing a number of steps:

- preprocessing
- lexical analysis
- syntax analysis
- semantic analysis
- code generation
- code optimization
- assembly
- linking



Compilers

A compiler translates *source code* from a high-level programming language into *machine code* for a given architecture by performing a number of steps:

- preprocessing
- lexical analysis
- syntax analysis
- semantic analysis
- code generation
- code optimization
- assembly
- linking

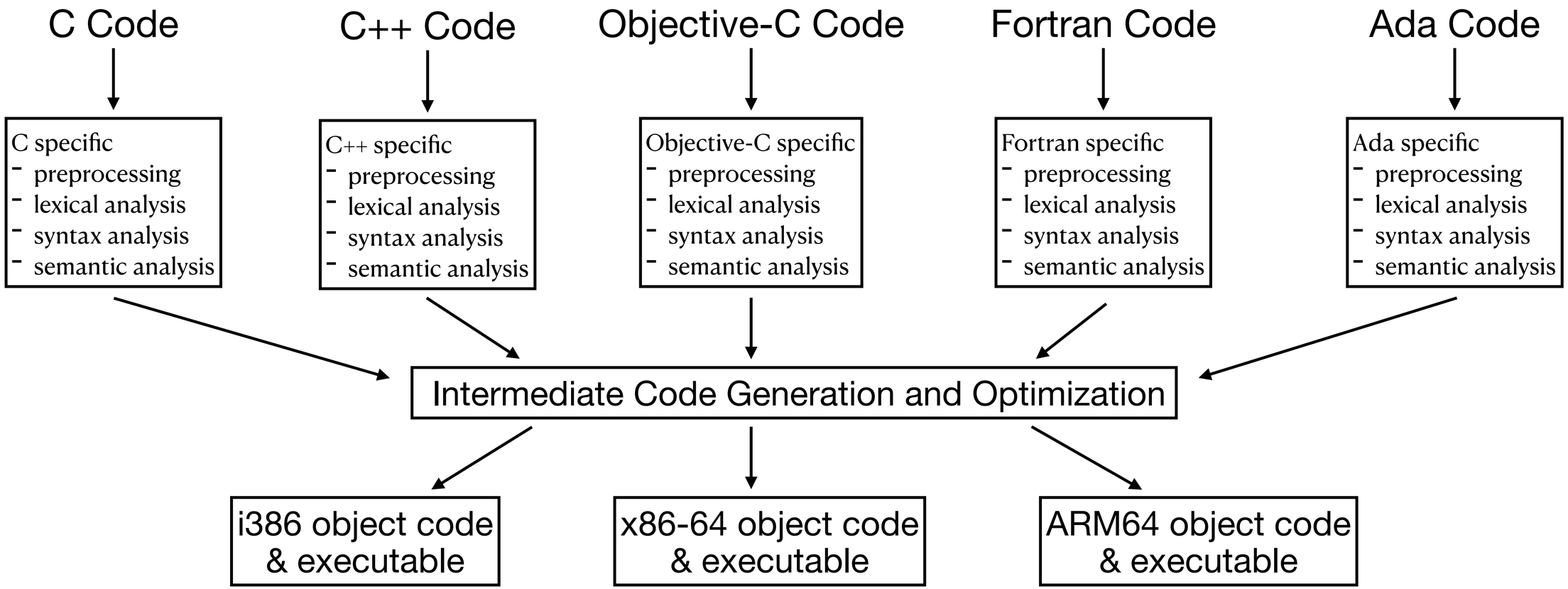
programming language specific

compiler specific

platform specific

Compilers

A compiler translates *source code* from a high-level programming language into *machine code* for a given architecture by performing a number of steps:



Compilers

There are many different closed- and open-source compiler chains:

- Intel C/C++ Compiler (or *icc*)
- Turbo C / Turbo C++ / C++ Builder (Borland)
- Microsoft Visual C++
- Clang (a frontend to *LLVM*)
- GNU Compiler Collection (or *gcc*)
- Portable C Compiler (or *pcc*)
- …

The GNU Compiler Collection

GCC is the default on many Unix platforms, although some have recently switched to `clang(1)` / LLVM.

The compiler chain or driver usually performs preprocessing (*e.g.* via `cpp(1)`), compilation (`cc(1)`), assembly (`as(1)`) and linking (`ld(1)`).

To be continued…