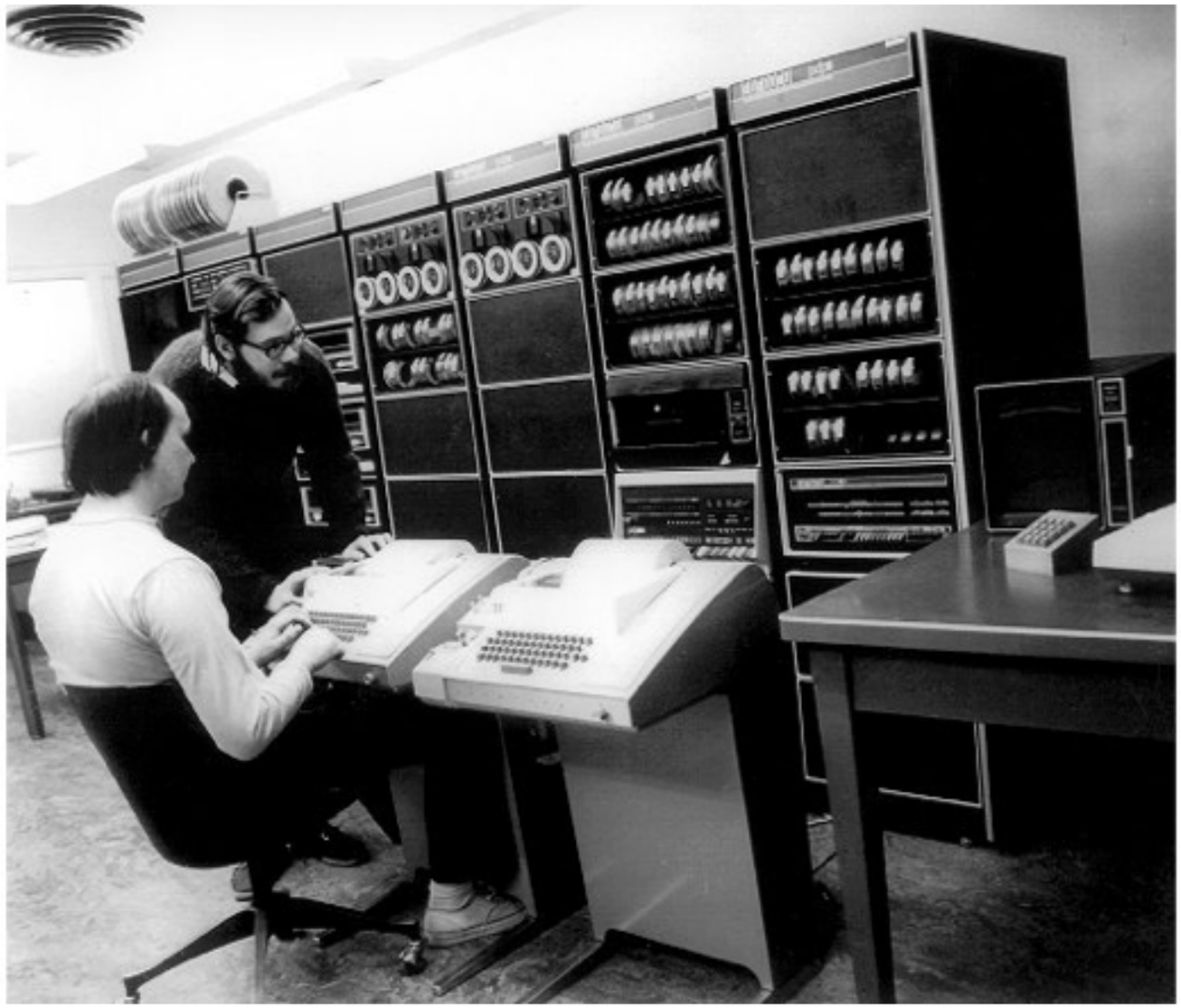
Tools

"to get a deeper understanding of the language"



Deep C - a 3 day course
Jon Jagger & Olve Maudal

A glimpse into tools often used when developing C

Exercise: Deep thought, Part 1

dt.c

```
#include "dt.h"

int dt_base_value;
#define MULTIPLIER 7
static int dt_answer;

static void run_computer(void)
{
    dt_answer = dt_base_value * MULTIPLIER;
}

int dt_get_answer(void)
{
    run_computer();
    return dt_answer;
}
```

dt.h

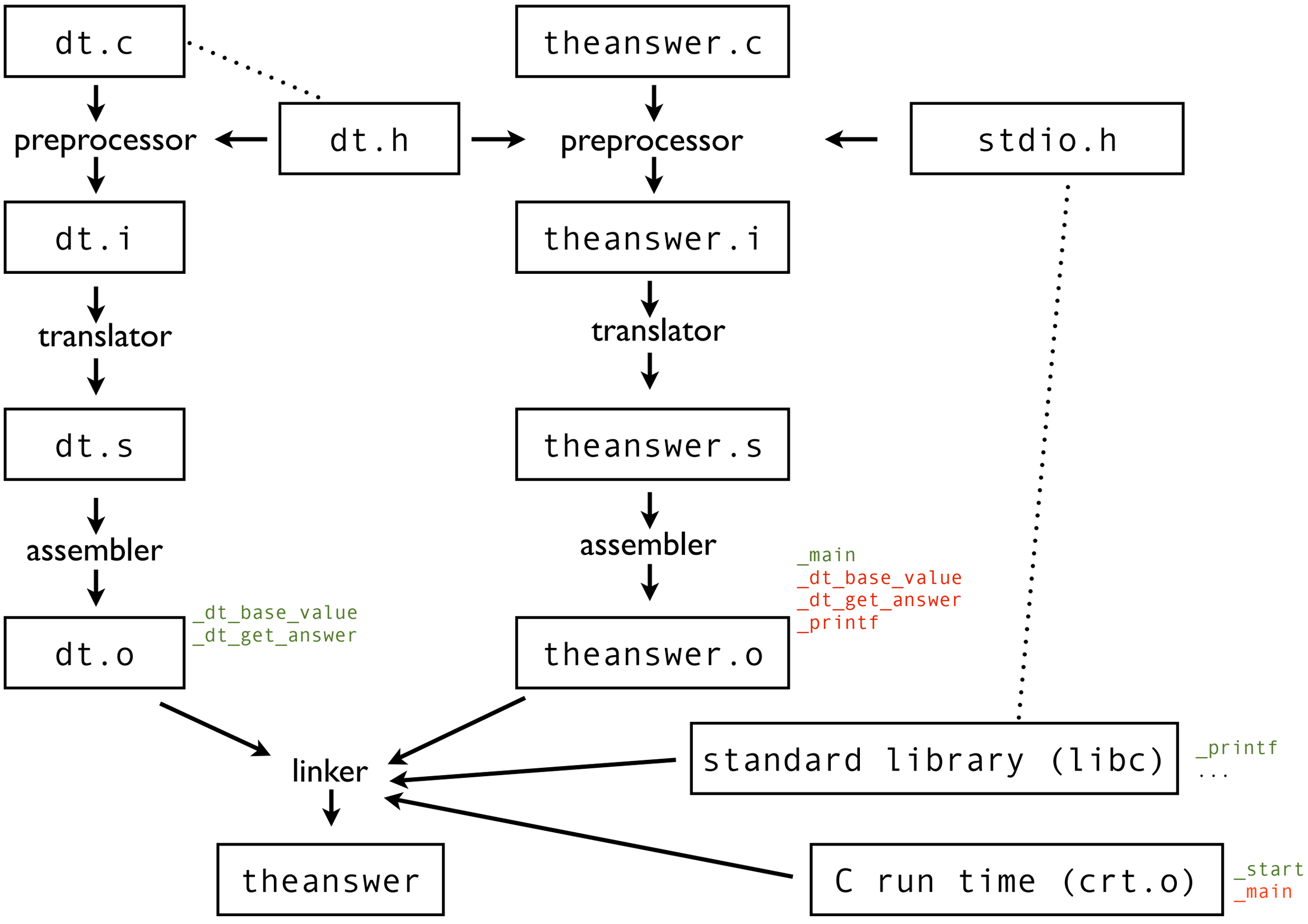
```
extern int dt_base_value;
int dt_get_answer(void);
```

theanswer.c

```
#include "dt.h"
#include <stdio.h>

int main(void)
{
    dt_base_value = 6;
    int answer = dt_get_answer();
    printf("The answer is %d\n",
            answer);
}
```

```
$ cc -c dt.c
$ cc -c theanswer.c
$ cc -o theanswer theanswer.o dt.o
$ ./theanswer
The answer is 42
$
```


dt.c
dt.h
theanswer.c
stdio.h
preprocessor
preprocessor
dt.i
theanswer.i
translator
translator
dt.s
theanswer.s
assembler
assembler
dt.o
_dt_base_value
_dt_get_answer
theanswer.o
_main
_dt_base_value
_dt_get_answer
_printf
linker
standard library (libc)
_printf
...
theanswer
C run time (crt.o)
_start
_main

Exercise: Deep thought, Part 2

dt.c

```
#include "dt.h"

int dt_base_value;
static int dt_answer;

static void run_computer(int multiplier)
{
    dt_answer = dt_base_value * multiplier;
}

void dt_init(void)
{
    dt_base_value = 6;
}

int dt_compute_answer(void)
{
    run_computer(7);
    return dt_answer;
}
```

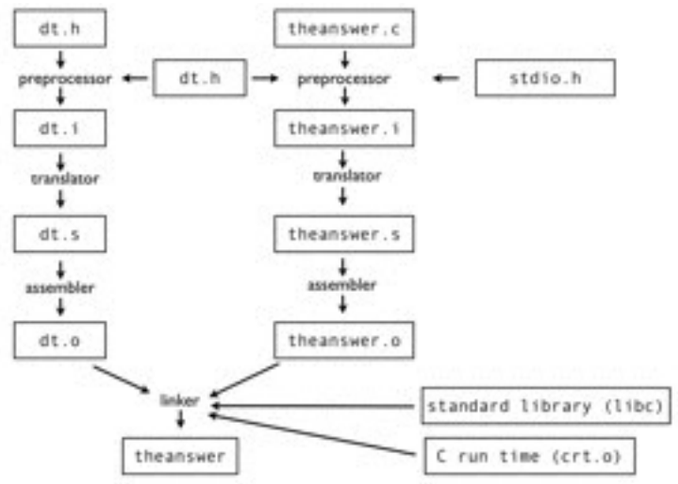
dt.h

```
void dt_init(void);
int dt_compute_answer(void);
```

theanswer.c

```
#include <stdio.h>
#include "dt.h"

int main(void)
{
    dt_init();
    int answer = dt_compute_answer();
    printf("The answer is %d\n",
           answer);
}
```




```
$ cc -E dt.c >dt.i
$ cat dt.i
$ cc -S dt.i
$ cat dt.s
$ cc -c dt.s
$ nm dt.o

$ cc -c -save-temps theanswer.c
$ ls theanswer.*
$ nm theanswer.o

$ ld -lc -o theanswer dt.o theanswer.o /usr/lib/crt1.o
$ ./theanswer
The answer is 42
```

Exercise: Deep thought, Part 3

dt.c

```
#include "dt.h"

int dt_base_value;
static int dt_answer;

static void run_computer(int multiplier)
{
    dt_answer = dt_base_value * multiplier;
}

void dt_init(void)
{
    dt_base_value = 6;
}

int dt_compute_answer(void)
{
    run_computer(7);
    return dt_answer;
}
```

dt.h

```
void dt_init(void);
int dt_compute_answer(void);
```

theanswer.c

```
#include <stdio.h>
#include "dt.h"

int main(void)
{
    dt_init();
    int answer = dt_compute_answer();
    printf("The answer is %d\n",
           answer);
}
```

```
$ cc -g -o theanswer dt.c theanswer.c
$ gdb theanswer
(gdb) run
(gdb) break run_computer
(gdb) run
(gdb) set dt_base_value = 8
(gdb) cont
(gdb) disassemble run_computer
(gdb) set disassembly-flavor intel
(gdb) disassemble run_computer
(gdb) help
(gdb) quit
```

Summary

- hello world!
- behaviour
- vocabulary of the language
- compiler, translator, assembler, linker
- standard library and C run-time
- memory layout and execution stack

TODO: profiling