Lecture 10.2: Overview

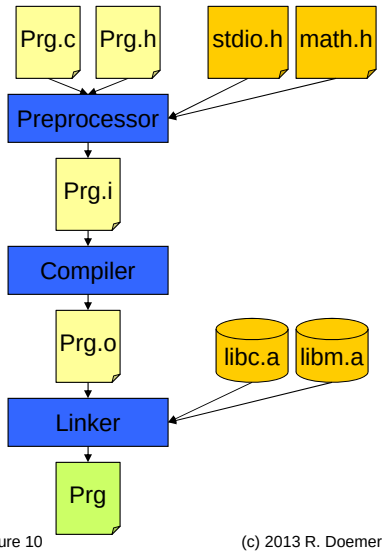
- Translation Units
  - Introduction
  - Compiler components
    - Preprocessor
    - Compiler
    - Linker
  - Modules
  - Program example **`PhotoLab2`**
    - Module **`FileIO`**
    - Module **`Age`**
    - Module **`Main`**

Translation Units

- Introduction
  - C compilation process is a sequence of phases
    1. Preprocessing (handle # directives)
    2. Scanning and parsing (generate internal data structure)
    3. Instruction generation (emit stream of CPU instructions)
    4. Assembly (generate binary object file)
    5. Linking (combine objects into executable file)
  - C compiler consists of separate components
    - Preprocessor (processes # directives)
    - Compiler (compiles and assembles code)
    - Linker (processes object files and libraries)

Translation Units

- Compilation Phases
  - Source code
    - Program files
    - Header files
  - Preprocessed file
  - Object files
  - Library files
  - Executable file



Translation Units

- Source files
  - Header files: **`Program.h`**
    - Inclusion of required header files
    - Definitions of exported constants
    - Declarations of exported global variables
    - Declarations of exported functions
  - Program files: **`Program.c`**
    - Inclusion of required header files
    - Declaration and definition of local variables
    - Declaration and definition of local functions
    - Definitions of exported global variables
    - Definitions of exported functions

Translation Units

- C Preprocessor
  - preprocesses source files
  - handles # directives
- Preprocessing Directives
  - Constant definition
  - Macro definition
  - Header file inclusion
  - Conditional compilation

```
#define WIDTH  640
```

```
#define ABS(x) (x>0 ? x : -x)
```

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

```
#define DEBUG  /* comment out to turn debugging off */
...
#ifdef DEBUG
printf("value of x is now %d\n", x);
#endif
```


Translation Units

- Object files
  - **`Program.o`**
    - Compiled object code of source file **`Program.c`**
    - Use option **`-c`** in GNU compiler call to create object files
      `gcc -c Program.c -o Program.o -Wall -ansi`
  - **`Library.a`**
    - Archive of compiled object files
- Executable file
  - **`Program`**
    - Object files and libraries linked together into a complete file ready for execution
    - GNU compiler recognizes object files by .o suffix, so object files and libraries require no special option
      `gcc Program.o -lc -lm -o Program`

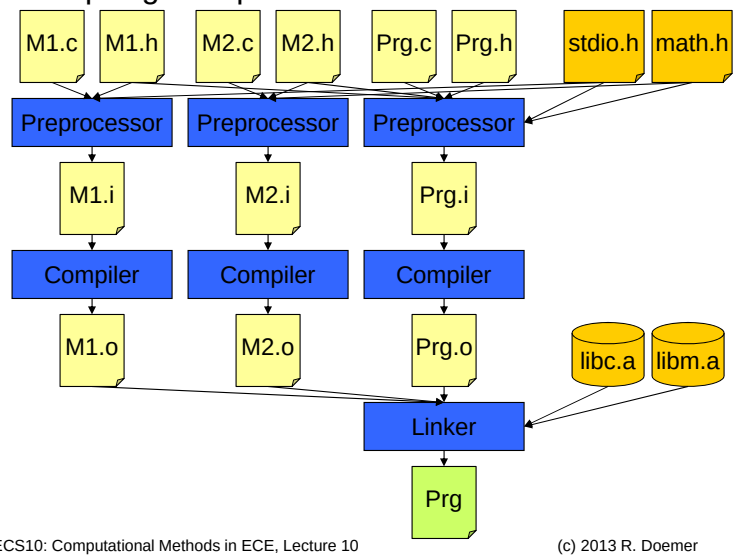
Translation Units

- Multiple Translation Units
  - C programs can be partitioned into multiple translation units, aka. *modules*
  - Modules typically consist of
    - Module header file (file suffix **`.h`**)
    - Module program file (file suffix **`.c`**)
    - Module object file (file suffix **`.o`**)
  - Modules are *linked* together
    - Linker combines object files and required libraries into an executable file
    - **`gcc Program.o Mod1.o Mod2.o –lc –lm -Wall –ansi –o Program`**

EECS10: Computational Methods in ECE, Lecture 10 (c) 2013 R. Doemer 7

Translation Units

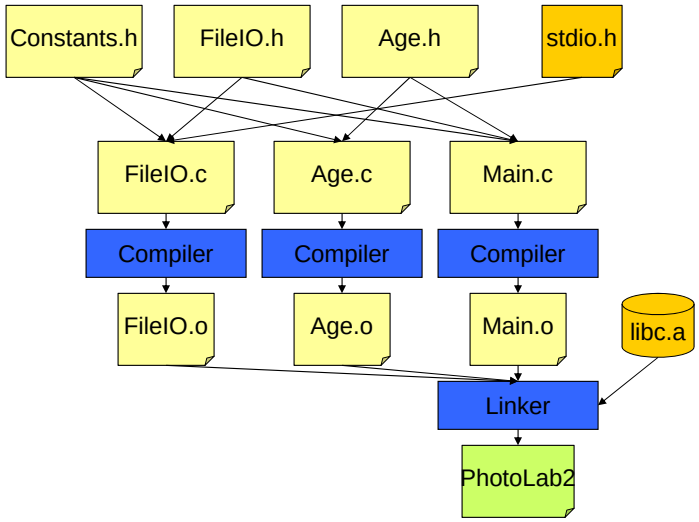
- Compiling Multiple Translation Units



EECS10: Computational Methods in ECE, Lecture 10 (c) 2013 R. Doemer 8

Translation Units

- Example: **`PhotoLab2`**



Translation Units

- Example: Header file **`Constants.h`**

```
/*********************************************************/
/* Constants.h: header file for constant definitions     */
/* author: Rainer Doemer                                 */
/* modifications: (most recent first)                    */
/* 09/02/13 RD  version for Summer 2013                  */
/*********************************************************/

#ifndef CONSTANTS_H
#define CONSTANTS_H

/*** global definitions ***/

#define WIDTH  640      /* image width */
#define HEIGHT 410      /* image height */
#define SLEN    80      /* max. string length */

#endif /* CONSTANTS_H */

/* EOF Constants.h */
```

Translation Units

- Example: Header file **`FileIO.h`**

```
/*********************************************************/
/* FileIO.h: header file for I/O module                  */
/*********************************************************/
#ifndef FILE_IO_H
#define FILE_IO_H

#include "Constants.h"

int ReadImage(       /* read image from file */
        char Filename[SLEN],
        unsigned char R[WIDTH][HEIGHT],
        unsigned char G[WIDTH][HEIGHT],
        unsigned char B[WIDTH][HEIGHT]);

int WriteImage(      /* write image to file */
        char Filename[SLEN],
        unsigned char R[WIDTH][HEIGHT],
        unsigned char G[WIDTH][HEIGHT],
        unsigned char B[WIDTH][HEIGHT]);

#endif /* FILE_IO_H */
/* EOF FileIO.h */
```


Translation Units

- Example: Program file **`FileIO.c`**

```
/*********************************************************/
/* FileIO.c: program file for I/O module                 */
/*********************************************************/

#include <stdio.h>
#include "FileIO.h"

/*** function definitions ***/

int ReadImage(char Filename[SLEN],
        unsigned char R[WIDTH][HEIGHT],
        unsigned char G[WIDTH][HEIGHT],
        unsigned char B[WIDTH][HEIGHT])
{   /* ... function body ... */
} /* end of ReadImage */
int WriteImage(char Filename[SLEN],
        unsigned char R[WIDTH][HEIGHT],
        unsigned char G[WIDTH][HEIGHT],
        unsigned char B[WIDTH][HEIGHT])
{   /* ... function body ... */
} /* end of WriteImage */
/* EOF FileIO.c */
```

Translation Units

- Example: Header file **Age.h**

```
/*********************************************************/
/* Age.h: header file for aging operation                */
/*********************************************************/

#ifndef AGE_H
#define AGE_H

/*** header files ***/

#include "Constants.h"

/*** function declarations ***/

void Age(     /* age the image */
        unsigned char R[WIDTH][HEIGHT],
        unsigned char G[WIDTH][HEIGHT],
        unsigned char B[WIDTH][HEIGHT]);

#endif /* AGE_H */

/* EOF Age.h */
```


Translation Units

- Example: Program file **Age.c**

```
/*********************************************************/
/* Age.c: program file for aging operation               */
/*********************************************************/

#include “Age.h"

/*** function definitions ***/

/* age the image so that it looks like an old photo */

void Age(
        unsigned char R[WIDTH][HEIGHT],
        unsigned char G[WIDTH][HEIGHT],
        unsigned char B[WIDTH][HEIGHT])
{
    /* ... function body ... */
} /* end of Age */

/* EOF Age.c */
```

Translation Units

- Example: Program file **`Main.c`**

```
/*********************************************************/
/* Main.c: main program file                             */
/*********************************************************/
#include "Constants.h"
#include "FileIO.h"
#include “Age.h"

int main(void)
{   unsigned char R[WIDTH][HEIGHT];
    unsigned char G[WIDTH][HEIGHT];
    unsigned char B[WIDTH][HEIGHT];

    if (ReadImage(“UCI_Firetrucks.ppm", R, G, B) != 0)
    { return 10; }
    Age(R, G, B);
    if (WriteImage(“UCI_Firetrucks_old.ppm", R, G, B) != 0)
    { return 10; }

    return 0;
} /* end of main */
/* EOF Main.c */
```


Translation Units

- Example session:

```
% vi Constants.h
% vi FileIO.h
% vi FileIO.c
% vi Age.h
% vi Age.c
% vi Main.c
```

```
% gcc -c FileIO.c -o FileIO.o -Wall –ansi
% gcc -c Age.c -o Age.o -Wall –ansi
% gcc -c Main.c -o Main.o -Wall –ansi
% gcc FileIO.o Age.o Main.o -o PhotoLab2
% PhotoLab2
%
```

UCI_Firetrucks.ppm

UCI_Firetrucks_old.ppm