EECS 10: Computational Methods in Electrical and Computer Engineering

Lecture 24

Rainer Dömer

doemer@uci.edu

The Henry Samueli School of Engineering
Electrical Engineering and Computer Science
University of California, Irvine

Lecture 24: Overview

- Course Administration
  - Reminder: Final course evaluation
- Translation Units
  - Introduction
  - Compiler components
  - Modules
  - Program example **`PhotoLab`**
    - Module **`FileIO`**
    - Module **`Mirror`**
    - Module **`Main`**

Course Administration

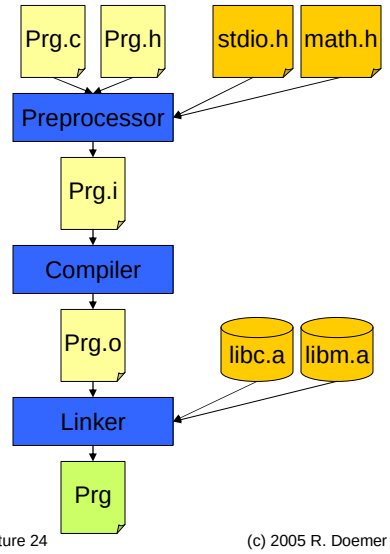
- Final Course Evaluation
  - Started last week, ends end of this week
  - Nov. 21, 2005, 8am through Dec. 5, 2005, 8am
  - Online via EEE Evaluation application
- Evaluation of Course and Instructor
  - Voluntary
  - Anonymous
  - Very valuable
    - Help to improve this class!
- Please spend 5 minutes!

Translation Units

- Introduction
  - C compilation process is a sequence of phases
    - Preprocessing (handle # directives)
    - Scanning and parsing (generate internal data structure)
    - Instruction generation (emit stream of CPU instructions)
    - Assembly (generate binary object file)
    - 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 file
    - Header file(s)
  - Preprocessed file
  - Object file
  - Library file(s)
  - 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

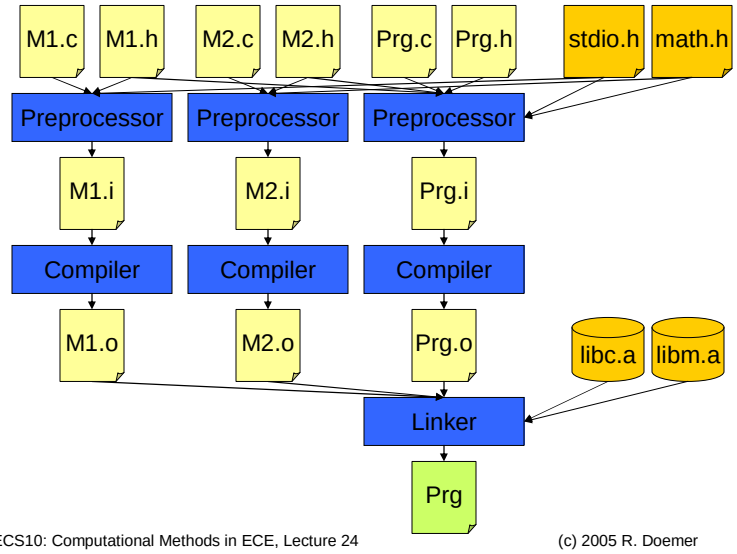
- Object files
  - **`Program.o`**
    - Compiled object code of source file **`Program.c`**
    - Use option **`-c`** in GNU compiler call to create object files
      `gcc -c Program.c -o Program.o -Wall -ansi`
  - **`Library.a`**
    - Archive of compiled object files
- Executable file
  - **`Program`**
    - 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 -o Program`**

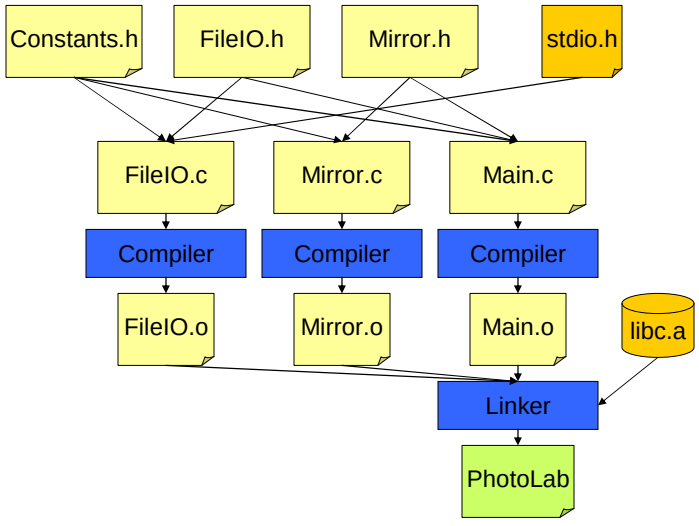
Translation Units

- Multiple Translation Units



Translation Units

- Example: **`PhotoLab`**



Translation Units

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

```
/*********************************************************/
/* Constants.h: header file for constant definitions     */
/* author: Rainer Doemer                                 */
/* modifications: (most recent first)                    */
/* 11/30/04 RD  initial version                          */
/*********************************************************/

#ifndef CONSTANTS_H
#define CONSTANTS_H

/*** global definitions ***/

#define WIDTH  480      /* width of photo */
#define HEIGHT 640      /* height of photo */
#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 ReadPhotoPPM(       /* read a photo from file */
        char Filename[SLEN],
        unsigned char R[WIDTH][HEIGHT],
        unsigned char G[WIDTH][HEIGHT],
        unsigned char B[WIDTH][HEIGHT]);

int WritePhotoPPM(      /* write a photo 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 ReadPhotoPPM(char Filename[SLEN],
        unsigned char R[WIDTH][HEIGHT],
        unsigned char G[WIDTH][HEIGHT],
        unsigned char B[WIDTH][HEIGHT])
{   /* ... function body ... */
}
int WritePhotoPPM(char Filename[SLEN],
        unsigned char R[WIDTH][HEIGHT],
        unsigned char G[WIDTH][HEIGHT],
        unsigned char B[WIDTH][HEIGHT])
{   /* ... function body ... */
}
/* EOF FileIO.c */
```


Translation Units

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

```
/*******************************************************/
/* Mirror.h: header file for mirror operation          */
/*******************************************************/

#ifndef MIRROR_H
#define MIRROR_H

/*** header files ***/

#include "Constants.h"

/*** function declarations ***/

void Mirror(     /* flip the image horizontally */
        unsigned char R[WIDTH][HEIGHT],
        unsigned char G[WIDTH][HEIGHT],
        unsigned char B[WIDTH][HEIGHT]);

#endif /* MIRROR_H */

/* EOF Mirror.h */
```

Translation Units

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

```
/*******************************************************/
/* Mirror.c: program file for mirror operation         */
/*******************************************************/

#include "Mirror.h"

/*** function definitions ***/

/* mirror effect: flip the image horizontally */

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

/* EOF Mirror.c */
```


Translation Units

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

```
/*******************************************************/
/* Main.c: main program file                           */
/*******************************************************/

#include "Constants.h"
#include "FileIO.h"
#include "Mirror.h"

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

    ReadPhotoPPM(“Input.ppm", R, G, B);
    Mirror(R, G, B);
    WritePhotoPPM("Output.ppm", R, G, B);

    return 0;
} /* end of main */

/* EOF Main.c */
```

Translation Units

- Example session:

```
% vi Constants.h
% vi FileIO.h
% vi FileIO.c
% vi Mirror.h
% vi Mirror.c
% vi Main.c
% gcc -c FileIO.c -o FileIO.o -Wall –ansi
% gcc -c Mirror.c -o Mirror.o -Wall –ansi
% gcc -c Main.c -o Main.o -Wall –ansi
% gcc FileIO.o Mirror.o Main.o -o PhotoLab
% PhotoLab
% pnmtojpeg Output.ppm > Output.jpg
%
```