EECS 10: Computational Methods in Electrical and Computer Engineering

Lecture 17

Rainer Dömer

doemer@uci.edu

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

Lecture 17: Overview

- File Processing
  - Standard input and output streams
  - File streams, I/O
  - Standard library functions in **`stdio.h`**
  - Program example **`PhotoLab.c`**
- Translation Units
  - Compiler components
    - Preprocessor, compiler, linker
  - Modules
  - Program example **`PhotoLab2`**
    - Modules **`FileIO`**, **`Age`**, **`Main`**

File Processing

- Introduction
  - Up to now, all data processed is available only during program run time
    - At program completion, all data is lost
  - *Persistent data* is stored even after a program exits
  - Persistent data is stored in files...
    - ... on the harddisk
    - ... on a removable disk (CD, memory stick, ...)
    - ... on a tape, ...
  - Input and output from/to files is organized as *I/O streams*

File Processing

- I/O Streams
  - Standard I/O streams (opened by the system)
    - **`stdin`** standard input stream (i.e. **`scanf()`**)
    - **`stdout`** standard output stream (i.e. **`printf()`**)
    - **`stderr`** standard error stream (i.e. **`perror()`**)
  - File I/O streams (explicitly opened by a program)
    - Open a file **`fopen()`**
    - Write data to a file **`fprintf()`**, **`fputs()`**, etc.
    - Read data from a file **`fscanf()`**, **`fgets()`**, etc.
    - Close a file **`fclose()`**
  - In C, all I/O functions are ...
    - ... declared in header file **`stdio.h`**
    - ... implemented in the standard C library

Standard I/O Functions

- Functions declared in **`stdio.h`** (part 1/4)
  - **`int printf(const char *fmt, ...);`**
  - **`int scanf(const char *fmt, ...);`**
    - formatted output/input to/from stream **`stdin`**/**`stdout`**
  - **`int sprintf(char *s, const char *fmt, ...);`**
  - **`int sscanf(const char *s, const char *fmt, ...);`**
    - formatted output/input to/from a string **`s`**
  - **`int getchar(void);`**
  - **`int putchar(int c);`**
    - input/output of a single character to/from stream **`stdin`**/**`stdout`**
  - **`char *gets(char *s);`**
  - **`int puts(const char *s);`**
    - input/output of strings to/from stream **`stdin`**/**`stdout`**

Standard I/O Functions

- Functions declared in **`stdio.h`** (part 2/4)
  - **`typedef __FILE FILE;`**
    - opaque type for a file handle
  - **`FILE *fopen(const char *n, const char *m);`**
    - open file named **`n`** for input (**`"r"`**), output (**`"w"`**), or append (**`"a"`**)
    - returns a file handle, or **`NULL`** in case of an error
  - **`int fclose(FILE *f);`**
    - closes an open file handle
  - **`int fprintf(FILE *f, const char *fmt, ...);`**
  - **`int fscanf(FILE *f, const char *fmt, ...);`**
  - **`int fgetc(FILE *f);`**
  - **`char *fgets(char *s, int n, FILE *f);`**
  - **`int fputc(int c, FILE *f);`**
  - **`int fputs(const char *s, FILE *f);`**
    - input/output functions from/to stream **`f`**
  - **`int fflush(FILE *f);`**
    - flushes any unwritten data from a buffer into the file

Standard I/O Functions

- Functions declared in **`stdio.h`** (part 3/4)
  - **`typedef unsigned int size_t;`**
    - type for size of a block of memory (number of bytes)
  - **`size_t fread(void *p, size_t s, size_t n, FILE *f);`**
    - binary input to memory location **`p`** for **`n`** times **`s`** bytes from file **`f`**
  - **`size_t fwrite(const void *p, size_t s, size_t n, FILE *f);`**
    - binary output from memory location **`p`** for **`n`** times **`s`** bytes to file **`f`**
  - **`long ftell(FILE *f);`**
    - return the current position in file **`f`** (from beginning)
  - **`int fseek(FILE *f, long pos, int w);`**
    - move to position **`pos`** in file **`f`** (from beginning/current pos/end)
  - **`void rewind(FILE *f);`**
    - move to beginning of file **`f`**
  - **`int feof(FILE *f);`**
    - check if end of file **`f`** is reached

Standard I/O Functions

- Functions declared in **`stdio.h`** (part 4/4)
  - **`int ferror(FILE *f);`**
    - returns the current error status for file **`f`**
  - **`void perror(const char *prg);`**
    - print current error for program **`prg`** to stream **`stderr`**
  - **`int remove(const char *filename);`**
    - delete file **`filename`**
  - **`int rename(const char *old, const char *new);`**
    - rename file **`old`** to new name **`new`**

File Processing

- Program example: **PhotoLab**
  - Digital image manipulation
    - Read an image from a file
    - Manipulate the image in memory
    - Write the modified image to file
  - Portable Pixel Map (PPM) file format
    - Simple uncompressed file format for color images
    - ➢ Header section (including picture width, height)
    - ➢ Data section (pixel intensities for red, green, and blue)

```
P6
640 480
255
RGBRGBRGB…
```


File Processing

- Program example: **PhotoLab.c** (part 1/10)

```
/*************************************************************/
/* PhotoLab.c: final assignment for EECS 10 in Fall '18  */
/*                                                           */
/* modifications: (most recent first)                        */
/* 07/24/18 RD    adjusted for lecture usage                 */
/*************************************************************/

#include <stdio.h>
#include <stdlib.h>

/*** global definitions ***/

#define WIDTH  640      /* image width */
#define HEIGHT 480      /* image height */
#define SLEN    80      /* max. string length */

...
```

File Processing

- Program example: **`PhotoLab.c`** (part 2/10)

```
...
/*** function definitions ***/

/* write the RGB image to a PPM file    */
/* (return 0 for success, >0 for error) */
int WriteImage(char Filename[SLEN],
               unsigned char R[WIDTH][HEIGHT],
               unsigned char G[WIDTH][HEIGHT],
               unsigned char B[WIDTH][HEIGHT])
{
    FILE *File;
    int  x, y;

    File = fopen(Filename, "w");
    if (!File)
    {   printf("\nCannot open file \"%s\"!\n", Filename);
        return(1);
    }
...
```


File Processing

- Program example: **`PhotoLab.c`** (part 3/10)

```
...
    fprintf(File, "P6\n");
    fprintf(File, "%d %d\n", WIDTH, HEIGHT);
    fprintf(File, "255\n");
    for(y=0; y<HEIGHT; y++)
    {   for(x=0; x<WIDTH; x++)
        {   fputc(R[x][y], File);
            fputc(G[x][y], File);
            fputc(B[x][y], File);
        }
    }
    if (ferror(File))
    {   printf("\nFile error while writing to file!\n");
        return(2);
    }
    fclose(File);
    return(0);  /* success! */
} /* end of WriteImage */
...
```

File Processing

- Program example: **`PhotoLab.c`** (part 4/10)

```
...
/* read an RGB image from a PPM file    */
/* (return 0 for success, >0 for error) */

int ReadImage(char Filename[SLEN],
              unsigned char R[WIDTH][HEIGHT],
              unsigned char G[WIDTH][HEIGHT],
              unsigned char B[WIDTH][HEIGHT])
{
    FILE *File;
    char Type[SLEN];
    int  Width, Height, MaxValue, x, y;

    File = fopen(Filename, "r");
    if (!File)
    {   printf("\nCannot open file \"%s\"!\n", Filename);
        return(1);
    }
...
```


File Processing

- Program example: **`PhotoLab.c`** (part 5/10)

```
...
    fscanf(File, "%79s", Type);
    if (Type[0] != 'P' || Type[1] != '6' || Type[2] != 0)
    {   printf("\nUnsupported file format!\n");
        return(2);
    }
    fscanf(File, "%d", &Width);
    if (Width != WIDTH)
    {   printf("\nUnsupported image width %d!\n", Width);
        return(3);
    }
    fscanf(File, "%d", &Height);
    if (Height != HEIGHT)
    {   printf("\nUnsupported image height %d!\n", Height);
        return(4);
    }
...
```

File Processing

- Program example: **PhotoLab.c** (part 6/10)

```
...
    fscanf(File, "%d", &MaxValue);
    if (MaxValue != 255)
    {   printf("\nUnsupported maximum %d!\n", MaxValue);
        return(5);
    }
    if ('\n' != fgetc(File))
    {   printf("\nCarriage return expected!\n");
        return(6);
    }
    for(y=0; y<HEIGHT; y++)
    {   for(x=0; x<WIDTH; x++)
        {   R[x][y] = fgetc(File);
            G[x][y] = fgetc(File);
            B[x][y] = fgetc(File);
        }
    }
...
```


File Processing

- Program example: **PhotoLab.c** (part 7/10)

```
...
    if (ferror(File))
    {   printf("\nFile error while reading from file!\n");
        return(7);
    }
    fclose(File);
    return(0);  /* success! */
} /* end of ReadImage */

...
```

File Processing

- Program example: **`PhotoLab.c`** (part 8/10)

```
...
/* modify the image...  ;-) */

void ModifyImage(unsigned char R[WIDTH][HEIGHT],
                 unsigned char G[WIDTH][HEIGHT],
                 unsigned char B[WIDTH][HEIGHT])
{   int  x, y;

    for(y=0; y<HEIGHT; y++)
    {   for(x=0; x<WIDTH; x++)
        {
            B[x][y] = (R[x][y] + G[x][y] + B[x][y]) / 5;
            R[x][y] = (unsigned char) (B[x][y]*1.6);
            G[x][y] = (unsigned char) (B[x][y]*1.6);
        }
    }

} /* end of ModifyImage */
...
```


File Processing

- Program example: **`PhotoLab.c`** (part 9/10)

```
...
/*** main program ***/

int main(void)
{
    /* image data */
    unsigned char R[WIDTH][HEIGHT];
    unsigned char G[WIDTH][HEIGHT];
    unsigned char B[WIDTH][HEIGHT];
    /* file name */
    char Filename[SLEN];

...
```

File Processing

- Program example: **`PhotoLab.c`** (part 10/10)

```
...
    printf("Enter input file name:  ");
    scanf("%79s", Filename);
    if (ReadImage(Filename, R,G,B) != 0)
    { exit(10); }

    /* modify the image */
    ModifyImage(R, G, B);

    printf("Enter output file name: ");
    scanf("%79s", Filename);
    if (WriteImage(Filename, R,G,B) != 0)
    { exit(10); }

    return 0;
} /* end of main */

/* EOF */
```


File Processing

- Example session: **`PhotoLab.c`**

```
% vi PhotoLab.c
% gcc PhotoLab.c -o PhotoLab -Wall -ansi
% ./PhotoLab
Enter input file name:  library.ppm
Enter output file name: aging.ppm
%
```

library.ppm

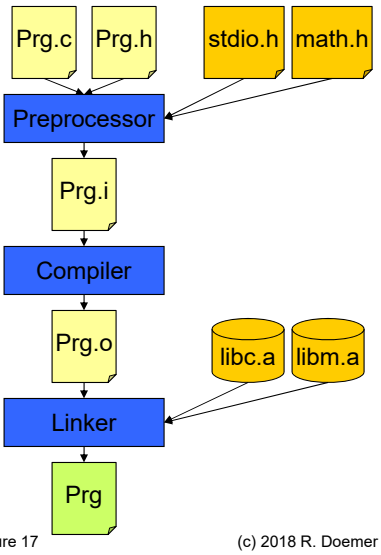
aging.ppm

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`

EECS10: Computational Methods in ECE, Lecture 17 (c) 2018 R. Doemer 25

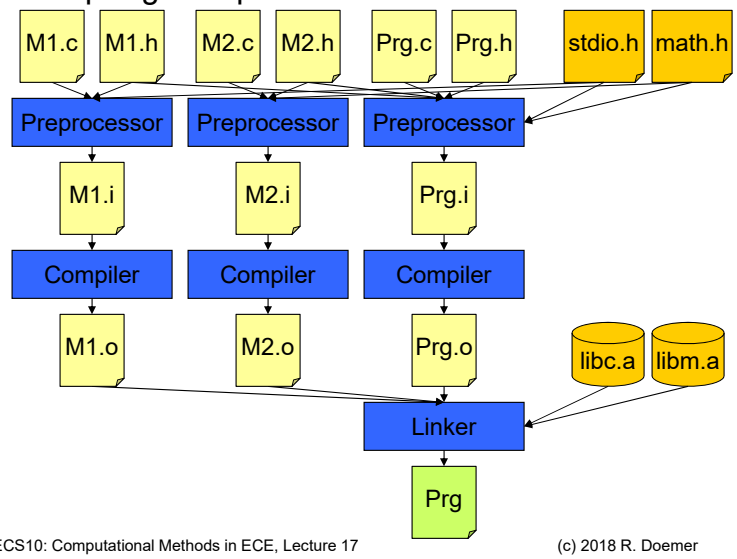
Translation Units

- 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 17 (c) 2018 R. Doemer 26

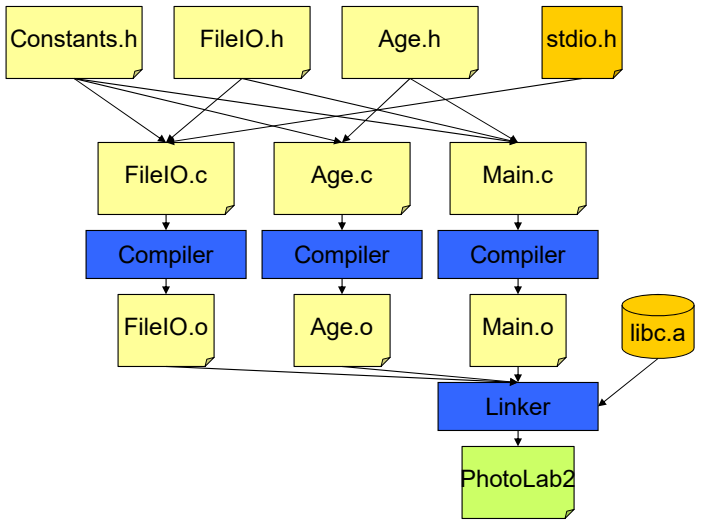
Translation Units

- Compiling Multiple Translation Units



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)                    */
/* 11/29/18 RD  version for Fall 2018                    */
/*********************************************************/

#ifndef CONSTANTS_H
#define CONSTANTS_H

/*** global definitions ***/

#define WIDTH  640      /* image width */
#define HEIGHT 480      /* 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 */
```

EECS

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 */
```

EECS10: Computational Methods in ECE, Lecture 17 (c) 2018 R. Doemer 32

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(“library.ppm", R, G, B) != 0)
    { return 10; }
    Age(R, G, B);
    if (WriteImage(“aging.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
%
```

library.ppm

aging.ppm