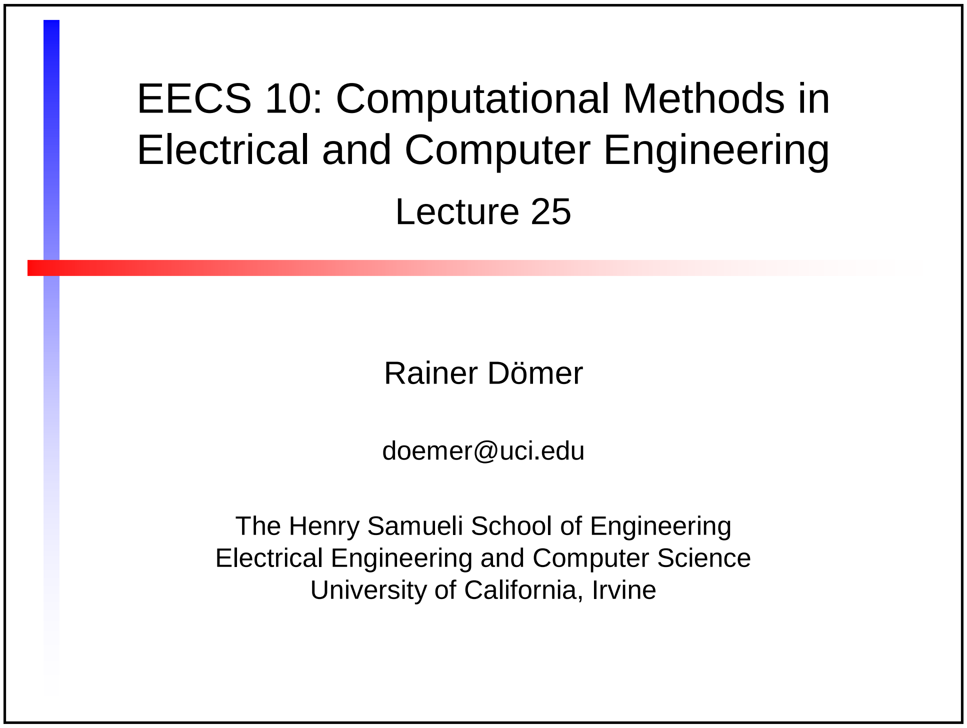


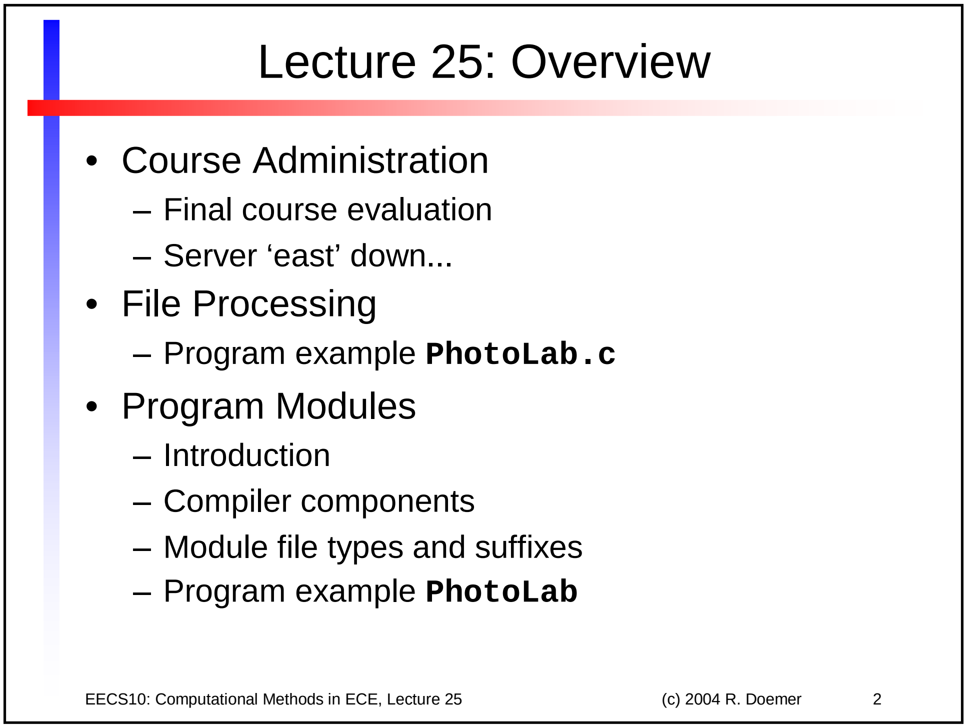
EECS 10: Computational Methods in Electrical and Computer Engineering

Lecture 25

Rainer Dömer

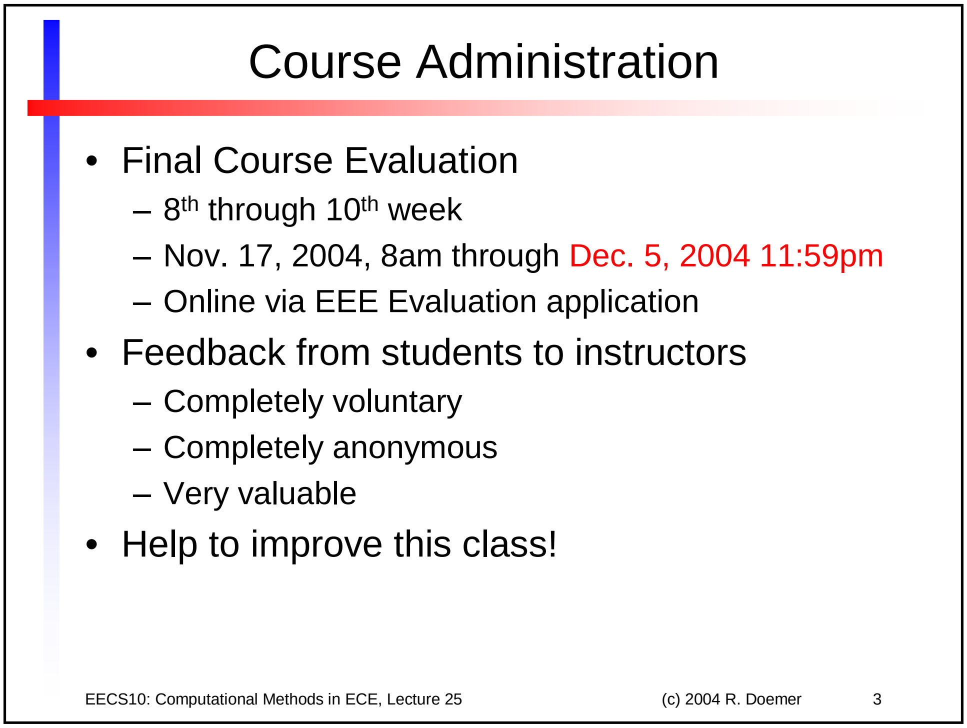
doemer@uci.edu

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



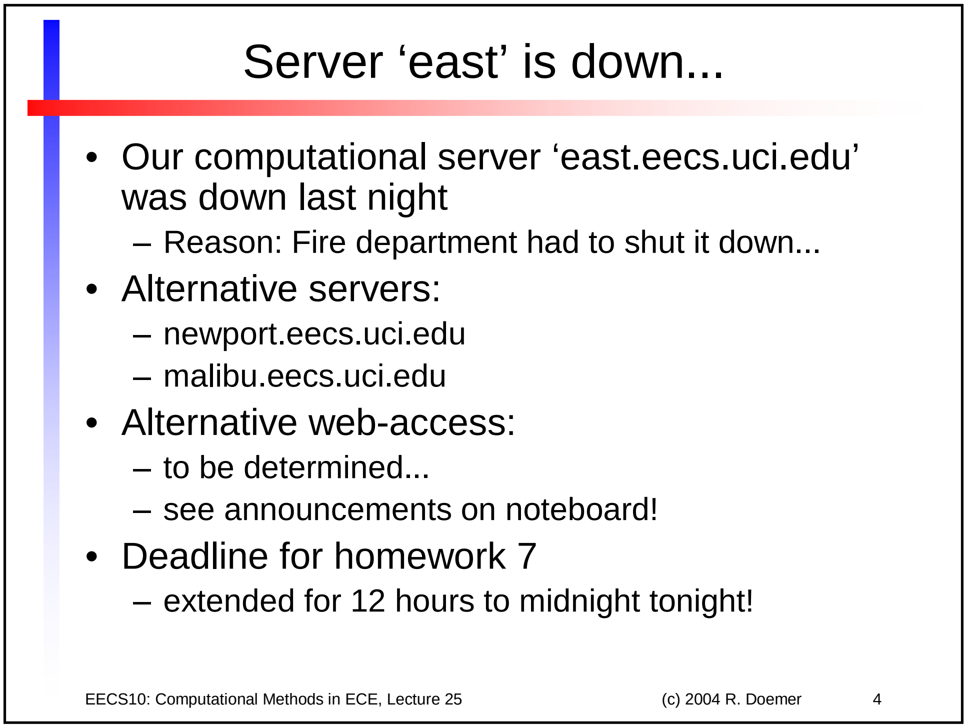
Lecture 25: Overview

- Course Administration
  - Final course evaluation
  - Server ‘east’ down...
- File Processing
  - Program example **`PhotoLab.c`**
- Program Modules
  - Introduction
  - Compiler components
  - Module file types and suffixes
  - Program example **`PhotoLab`**



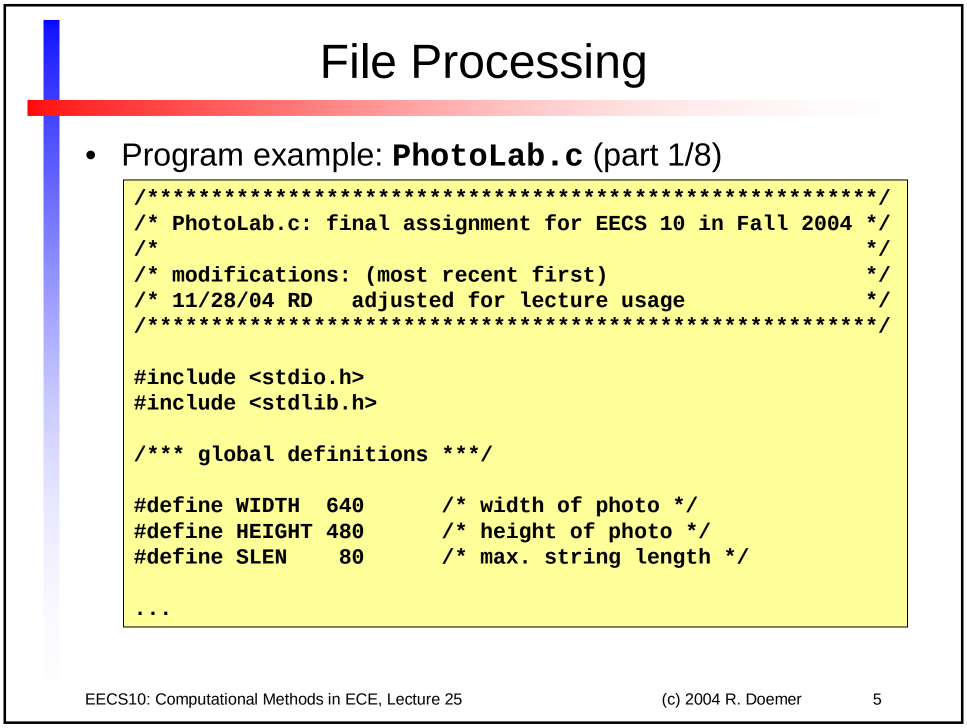
Course Administration

- Final Course Evaluation
  - 8th through 10th week
  - Nov. 17, 2004, 8am through Dec. 5, 2004 11:59pm
  - Online via EEE Evaluation application
- Feedback from students to instructors
  - Completely voluntary
  - Completely anonymous
  - Very valuable
- Help to improve this class!



Server ‘east’ is down...

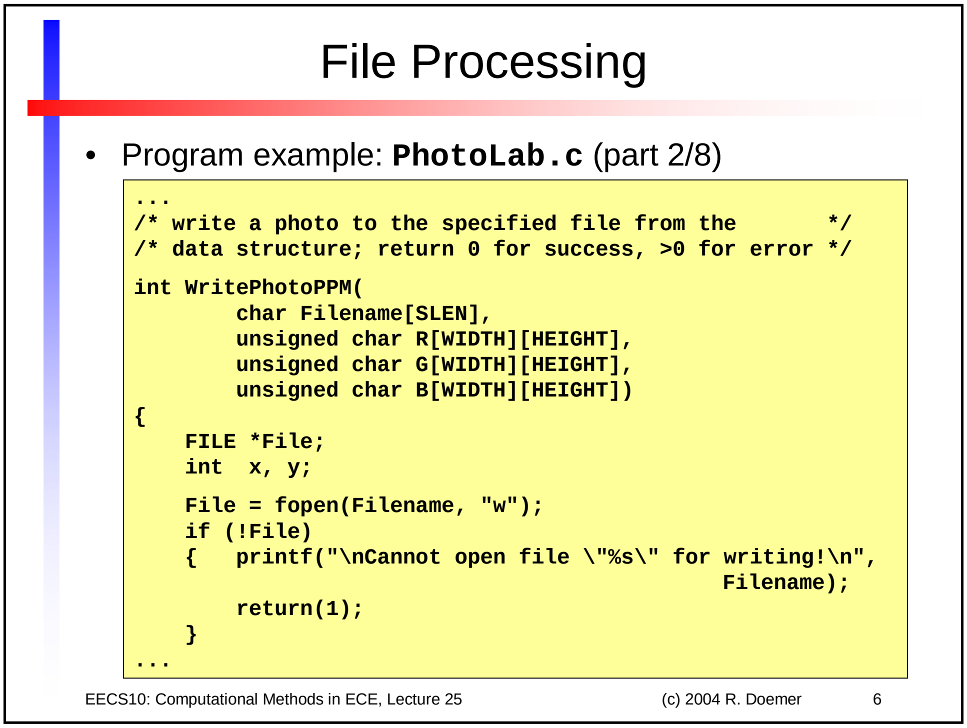
- Our computational server ‘east.eecs.uci.edu’ was down last night
  - Reason: Fire department had to shut it down...
- Alternative servers:
  - newport.eecs.uci.edu
  - malibu.eecs.uci.edu
- Alternative web-access:
  - to be determined...
  - see announcements on noteboard!
- Deadline for homework 7
  - extended for 12 hours to midnight tonight!



File Processing

- Program example: **PhotoLab.c** (part 1/8)

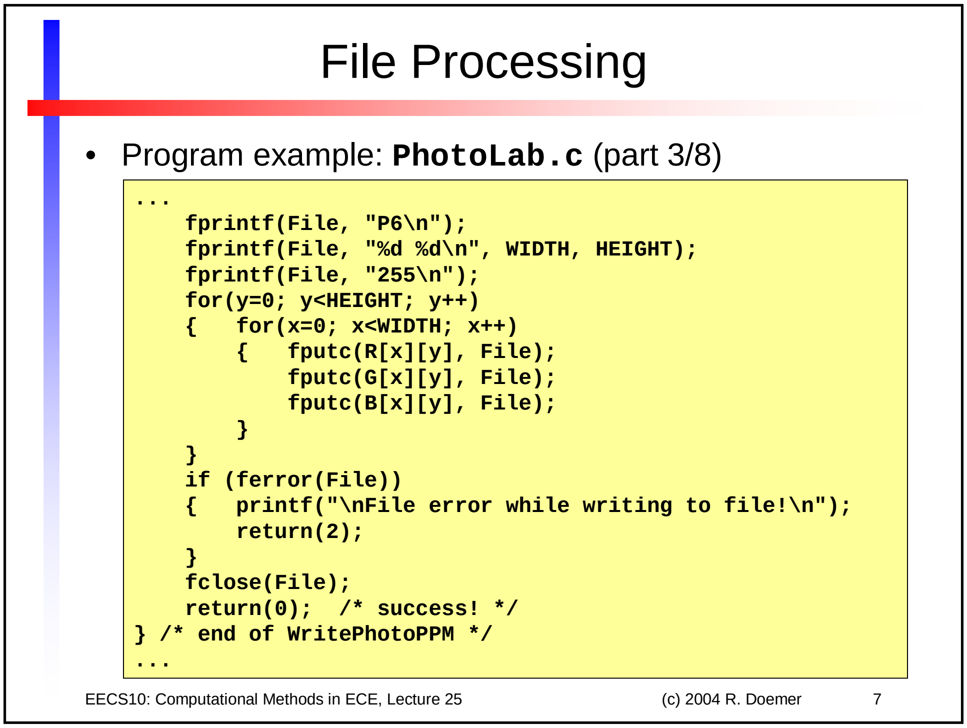
```
/*********************************************************/
/* PhotoLab.c: final assignment for EECS 10 in Fall 2004 */
/*                                                       */
/* modifications: (most recent first)                    */
/* 11/28/04 RD   adjusted for lecture usage              */
/*********************************************************/

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

/*** global definitions ***/

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

...
```



File Processing

- Program example: **PhotoLab.c** (part 2/8)

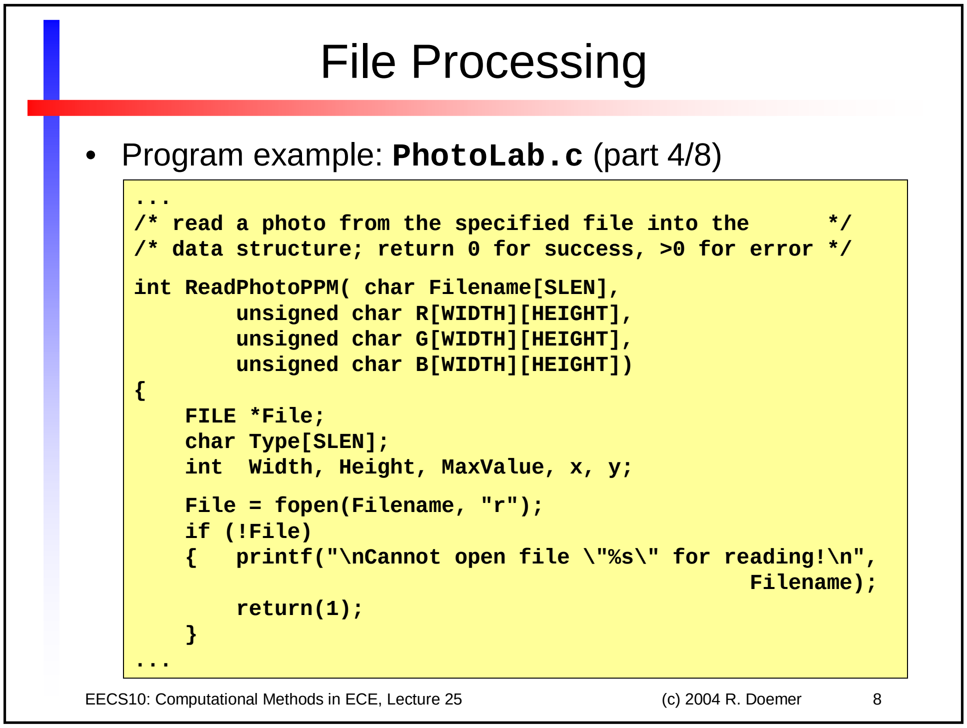
```
...
/* write a photo to the specified file from the       */
/* data structure; return 0 for success, >0 for error */
int WritePhotoPPM(
        char Filename[SLEN],
        unsigned char R[WIDTH][HEIGHT],
        unsigned char G[WIDTH][HEIGHT],
        unsigned char B[WIDTH][HEIGHT])
{
    FILE *File;
    int  x, y;
    File = fopen(Filename, "w");
    if (!File)
    {   printf("\nCannot open file \"%s\" for writing!\n",
                                              Filename);
        return(1);
    }
...
```



File Processing

- Program example: **`PhotoLab.c`** (part 3/8)

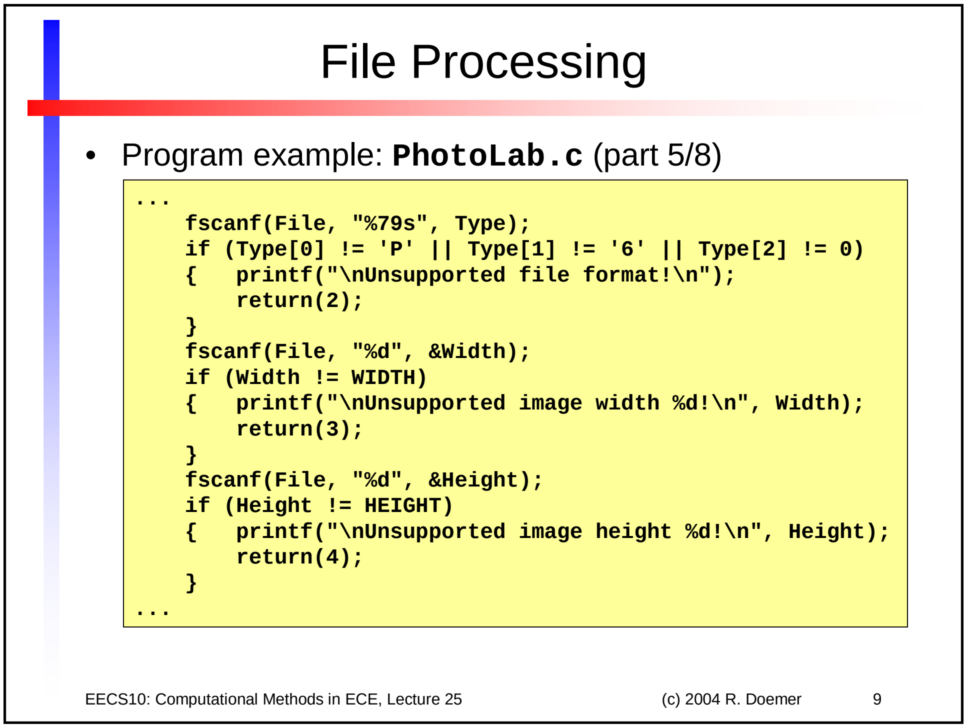
```
...
    fprintf(File, "P6\n");
    fprintf(File, "%d %d\n", WIDTH, HEIGHT);
    fprintf(File, "255\n");
    for(y=0; y<HEIGHT; y++)
    {   for(x=0; x<WIDTH; x++)
        {   fputc(R[x][y], File);
            fputc(G[x][y], File);
            fputc(B[x][y], File);
        }
    }
    if (ferror(File))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



File Processing

- Program example: **`PhotoLab.c`** (part 4/8)

```
...
/* read a photo from the specified file into the      */
/* data structure; return 0 for success, >0 for error */

int ReadPhotoPPM( 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\" for reading!\n",
                                                Filename);
        return(1);
    }
...
```

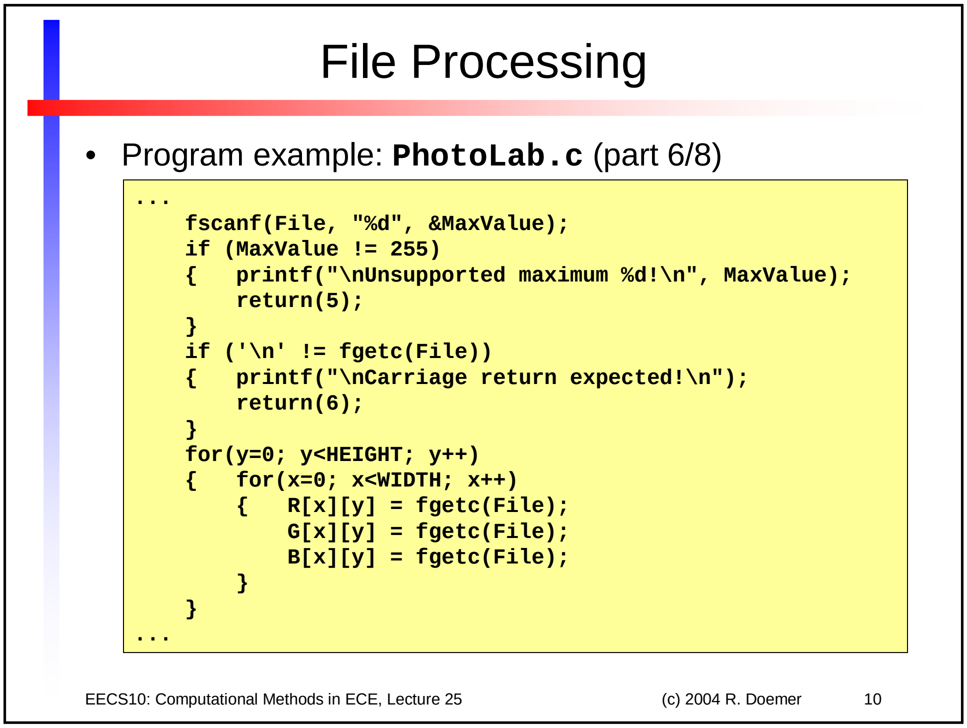


File Processing

- Program example: **`PhotoLab.c`** (part 5/8)

```
...
    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);
    }
...
```

EECS10: Computational Methods in ECE, Lecture 25 (c) 2004 R. Doemer 9

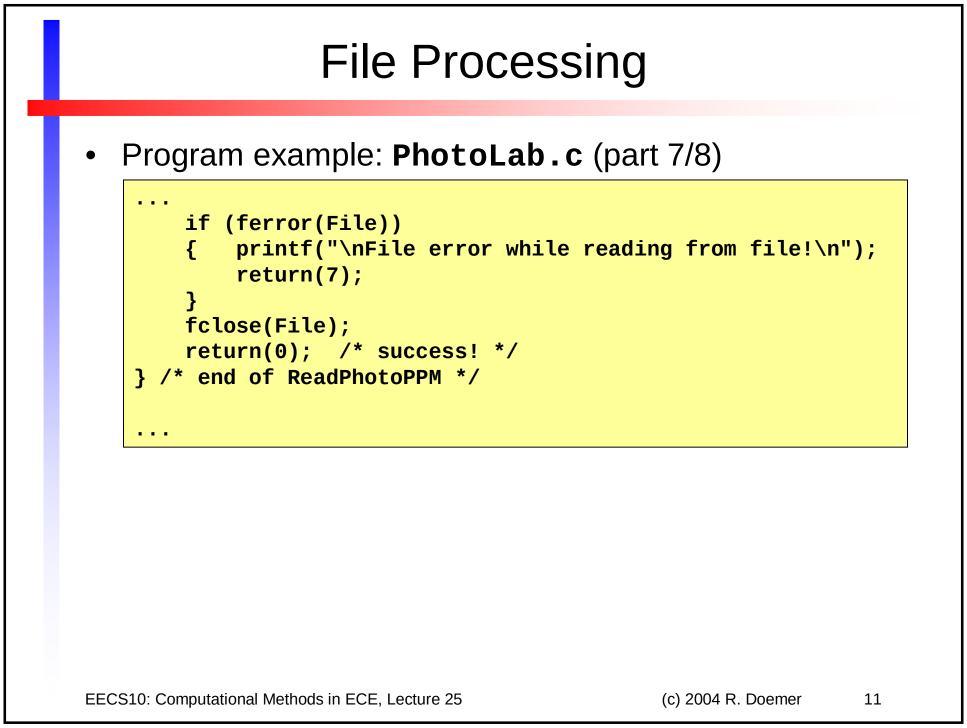


File Processing

- Program example: **`PhotoLab.c`** (part 6/8)

```
...
    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);
        }
    }
...
```

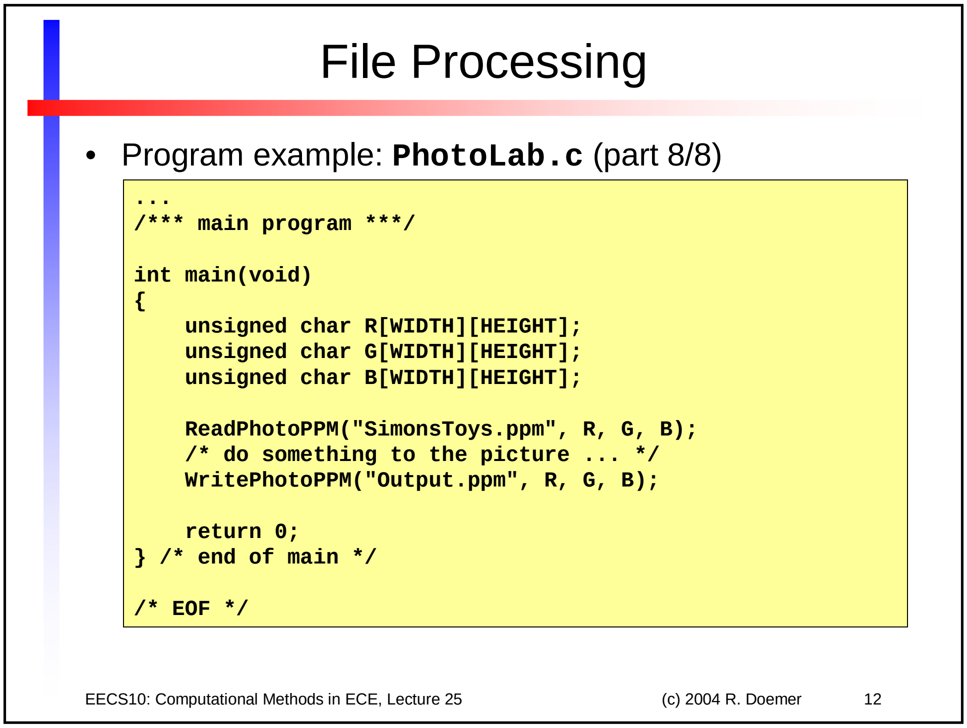
EECS10: Computational Methods in ECE, Lecture 25 (c) 2004 R. Doemer 10



File Processing

- Program example: **PhotoLab.c** (part 7/8)

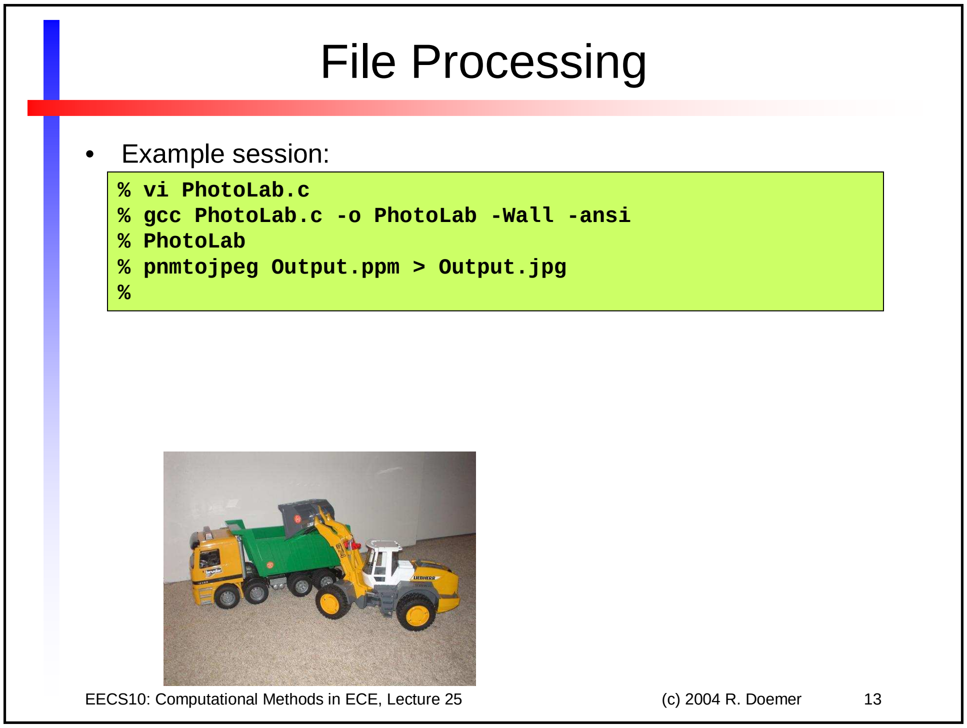
```
...
    if (ferror(File))
    {   printf("\nFile error while reading from file!\n");
        return(7);
    }
    fclose(File);
    return(0);  /* success! */
} /* end of ReadPhotoPPM */

...
```



File Processing

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

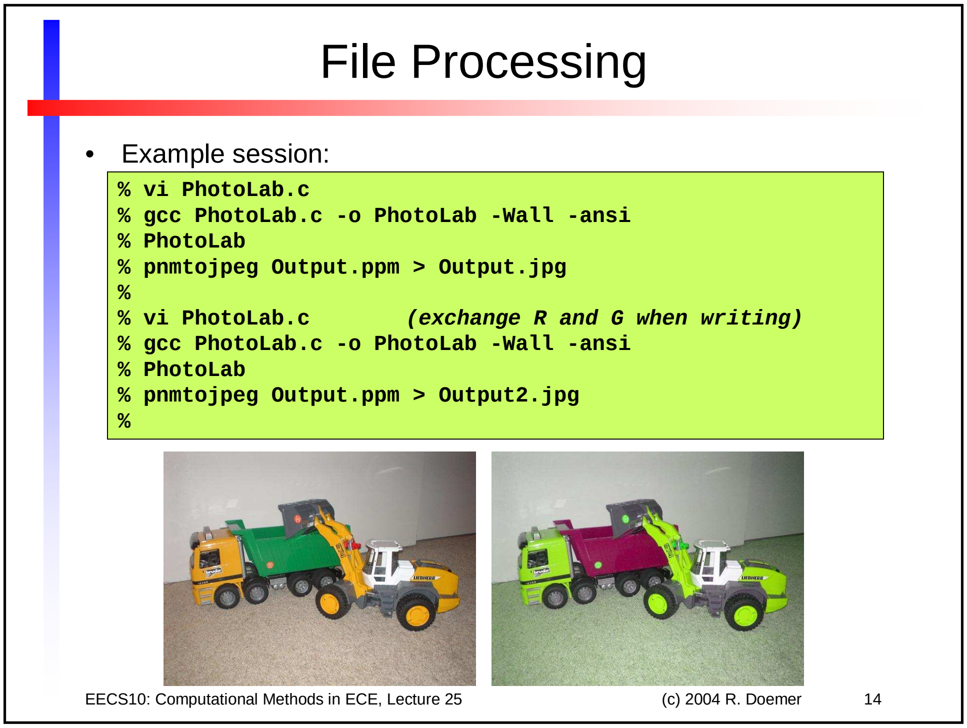
```
...
/*** main program ***/

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

    ReadPhotoPPM("SimonsToys.ppm", R, G, B);
    /* do something to the picture ... */
    WritePhotoPPM("Output.ppm", R, G, B);

    return 0;
} /* end of main */

/* EOF */
```



File Processing

- Example session:

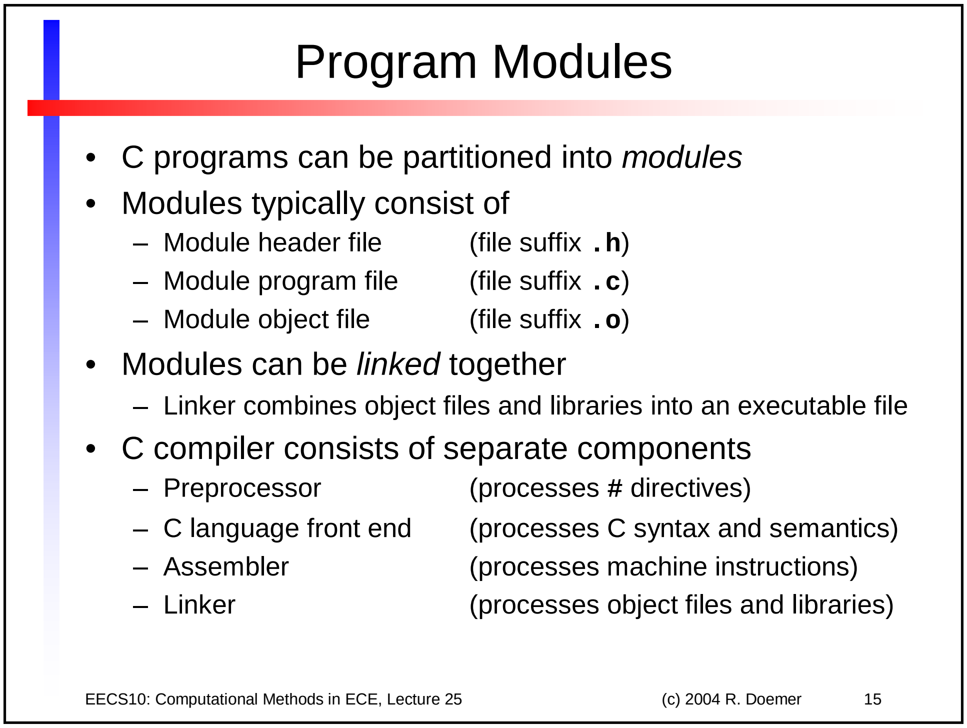
```
% vi PhotoLab.c
% gcc PhotoLab.c -o PhotoLab -Wall -ansi
% PhotoLab
% pnmtojpeg Output.ppm > Output.jpg
%
```



File Processing

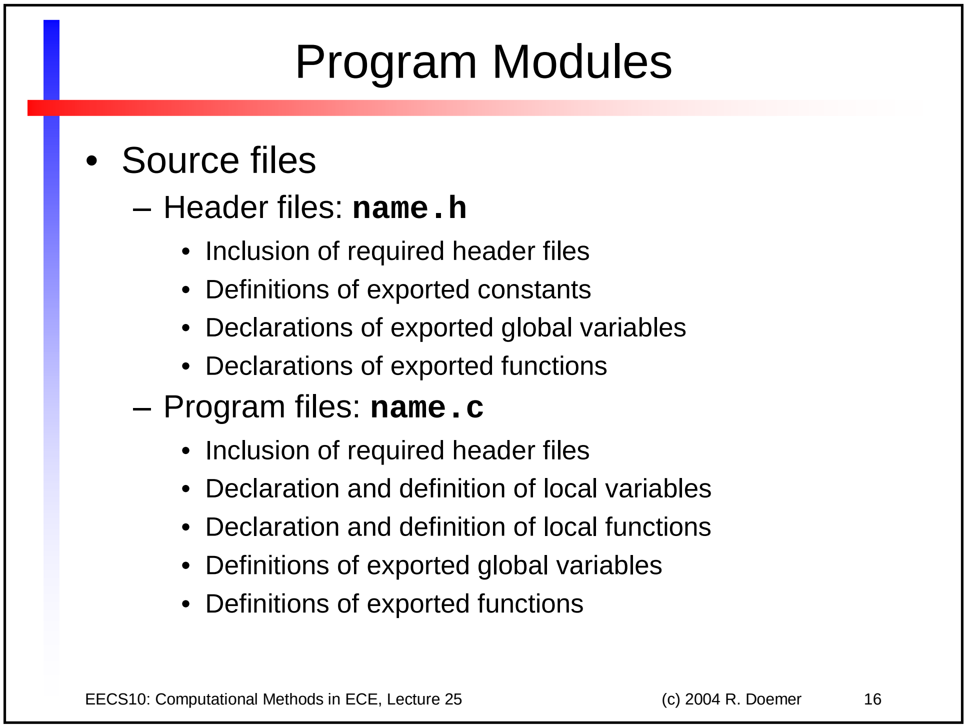
- Example session:

```
% vi PhotoLab.c
% gcc PhotoLab.c -o PhotoLab -Wall -ansi
% PhotoLab
% pnmtojpeg Output.ppm > Output.jpg
%
% vi PhotoLab.c        (exchange R and G when writing)
% gcc PhotoLab.c -o PhotoLab -Wall -ansi
% PhotoLab
% pnmtojpeg Output.ppm > Output2.jpg
%
```



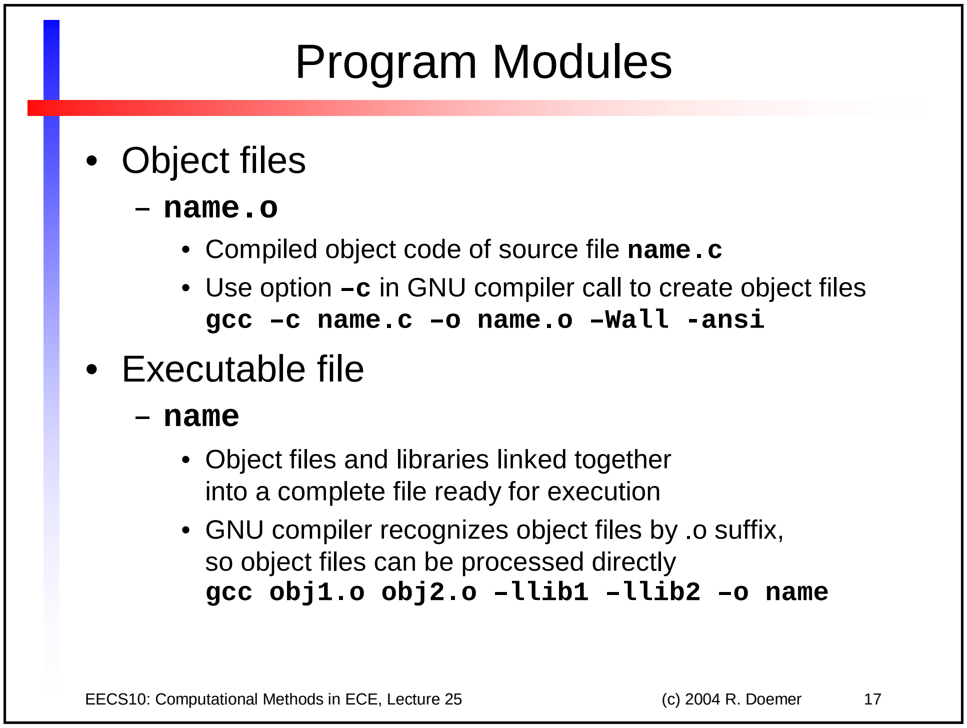
Program Modules

- C programs can be partitioned into *modules*
- Modules typically consist of
  - Module header file (file suffix **`.h`**)
  - Module program file (file suffix **`.c`**)
  - Module object file (file suffix **`.o`**)
- Modules can be *linked* together
  - Linker combines object files and libraries into an executable file
- C compiler consists of separate components
  - Preprocessor (processes # directives)
  - C language front end (processes C syntax and semantics)
  - Assembler (processes machine instructions)
  - Linker (processes object files and libraries)



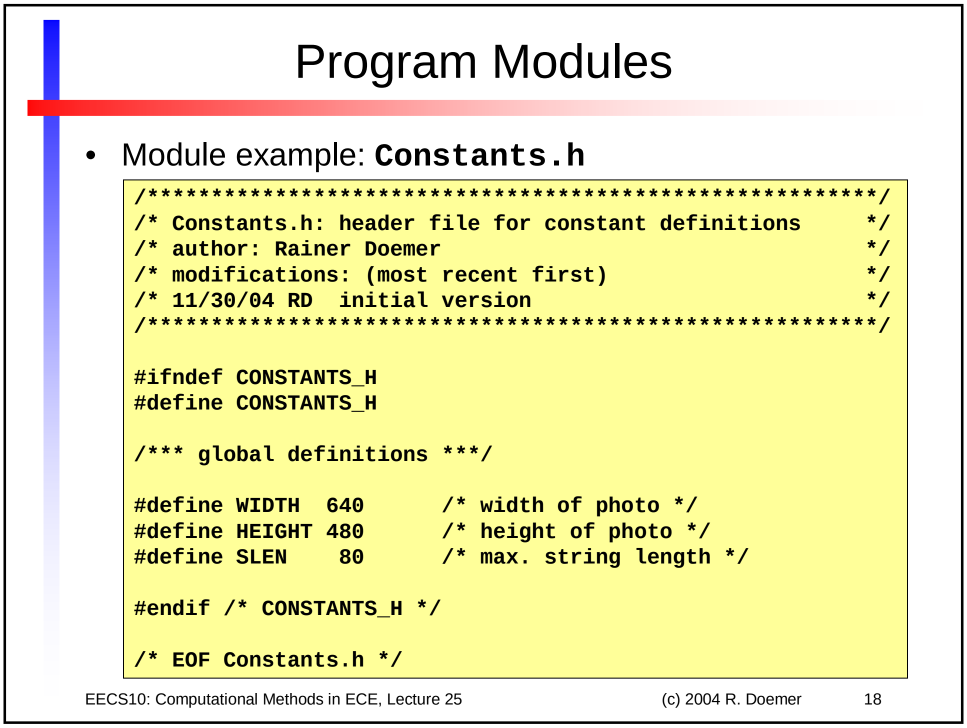
Program Modules

- Source files
  - Header files: **`name.h`**
    - Inclusion of required header files
    - Definitions of exported constants
    - Declarations of exported global variables
    - Declarations of exported functions
  - Program files: **`name.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



Program Modules

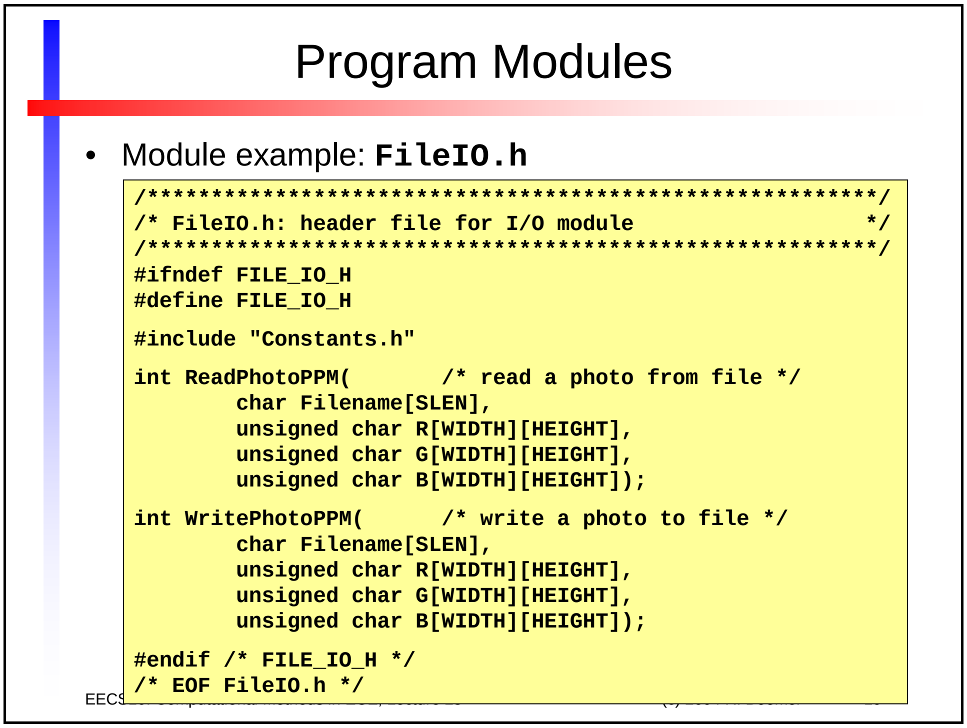
- Object files
  - **name.o**
    - Compiled object code of source file **name.c**
    - Use option **-c** in GNU compiler call to create object files
      gcc -c name.c -o name.o -Wall -ansi
- Executable file
  - **name**
    - Object files and libraries linked together
      into a complete file ready for execution
    - GNU compiler recognizes object files by .o suffix,
      so object files can be processed directly
      gcc obj1.o obj2.o -llib1 -llib2 -o name



Program Modules

- Module example: **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  640      /* width of photo */
#define HEIGHT 480      /* height of photo */
#define SLEN    80      /* max. string length */

#endif /* CONSTANTS_H */

/* EOF Constants.h */
```

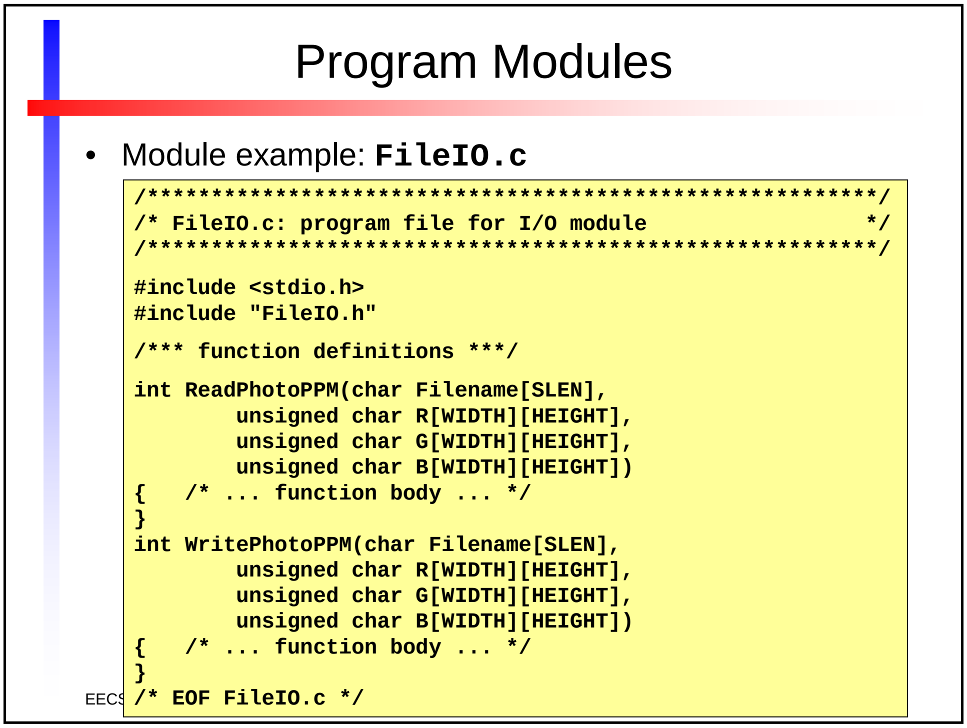


Program Modules

- Module example: **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 */
```

EECS

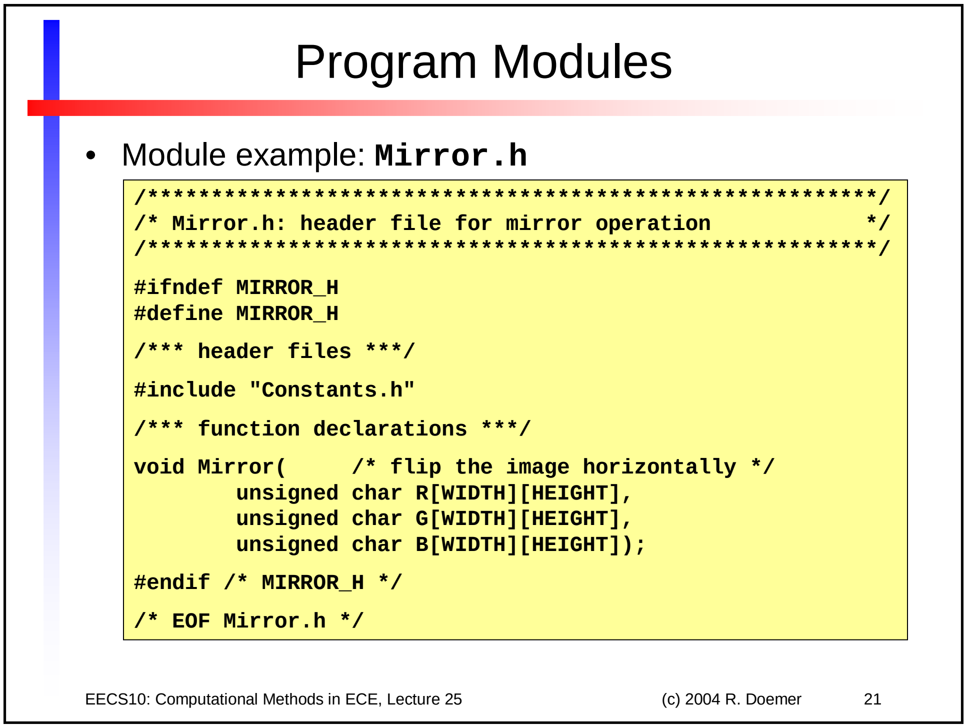


Program Modules

- Module example: **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 */
```

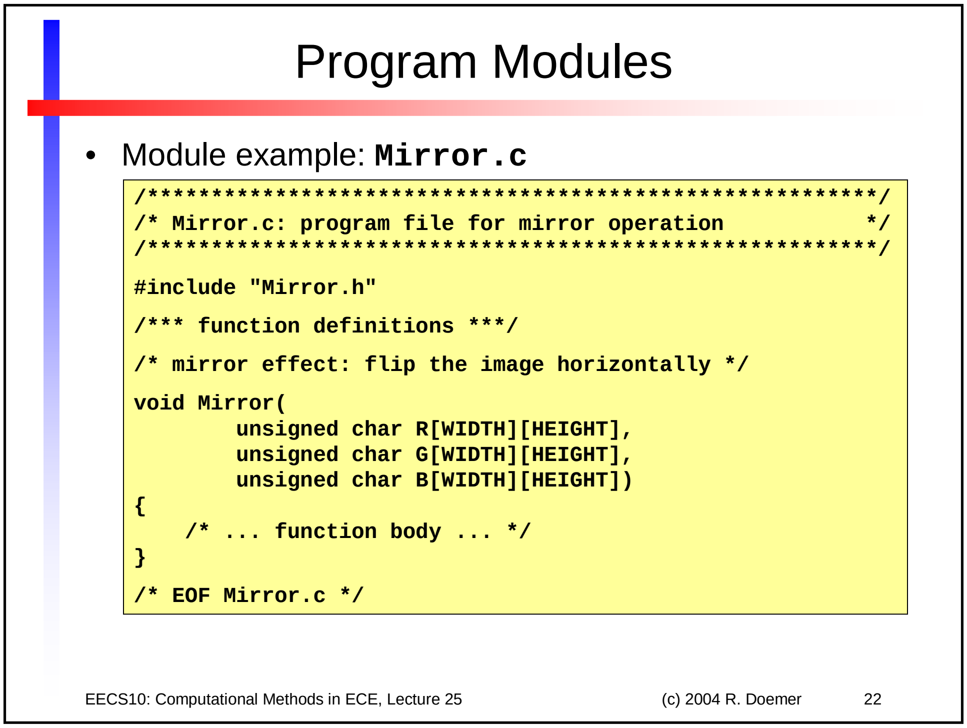
EECS



Program Modules

- Module example: **`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 */
```



Program Modules

- Module example: **`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 */
```

Program Modules

- Module example: **`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("SimonsToys.ppm", R, G, B);
    Mirror(R, G, B);
    WritePhotoPPM("Output.ppm", R, G, B);

    return 0;
} /* end of main */

/* EOF Main.c */
```

Program Modules

- 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
%
```