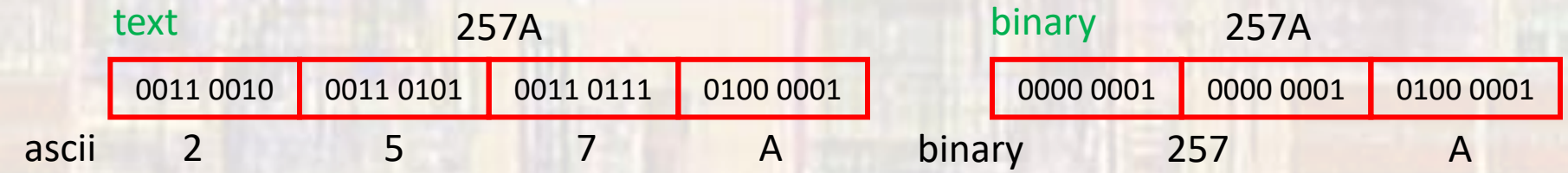
File I/O - Binary

Last updated 6/5/24

These slides introduce binary file operations

File I/O - Binary

- File Formats
  - Files can contain information in 2 different formats
  - Text
    - Stores characters (numbers are stored as their ascii values)
    - Line terminated by a newline (\n)
  - Binary
    - Raw bytes
    - File terminated by “end of file” EOF



This assumes 257 was a 16b integer
a full sized int would require 4 bytes 0x00000101

File I/O - Binary

- Stream Pointer Creation
  - Need to create a stream to transfer the data to/from our file
  - Identify the stream by name
  - Use a pointer of type FILE

```
FILE * pointer_name;

FILE * Student_Data_ptr;      // create a pointer
                              // to point to a stream
                              // currently NULL
```

File I/O - Binary

- Stream Pointer Assignment - fopen
  - Need to identify the file we are creating the stream to/from
  - "open" the file

    and
  - assign the pointer to the opened file

```
fopen("filename", "mode");          // returns a pointer to the file
```

the file extension .bin is commonly used

```
Student_Data_ptr = fopen("ee1601.bin", "rb");
```

the second \ is part of the path, the first \ says to treat the second \ like any other character – not a special character

```
Student_Data_ptr =
        fopen("C:\\users\\tim\\winter\\ee1601.bin", "rb");
```

File I/O - Binary

- fopen – binary modes

rb	read binary only, start at beginning if the file does not exist → error
wb	write binary only, start at beginning (erase all contents) if the file does not exist → creates it
ab	append binary only, start at end of current data if the file does not exist → creates it

Returns: address(pointer) of file or NULL if an error occurs

NULL is defined in the stdio library

File I/O - Binary

- fopen – binary modes – cont'd

r+b	read binary (can write), start at beginning if the file does not exist → error
w+b	write binary (can read), start at beginning (erase all contents) if the file does not exist → creates it
a+b	append binary (can read), start at end of current data if the file does not exist → creates it

Returns: address(pointer) of file or NULL if an error occurs

NULL is defined in the stdio library

File I/O - Binary

- Error checking
  - If the fopen() returns a NULL – we have an error

```
// create a stream pointer for the file
FILE * DataFile_strm_ptr;

//create a new file
if((DataFile_strm_ptr = fopen("myDataFile.bin", "wb")) == NULL){
     printf("Error opening file myDataFile.bin\n");
     exit (100);    // terminate program
    } // end if
```

exit – exits the program
requires <stdlib.h>

File I/O - Binary

- Close a file - fclose

```
fclose(file_pointer);

fclose(Student_Data_ptr );
```

File I/O - Binary

- Formatting output stream data – fwrite()
  - File write
  - Block format – no conversions, raw bytes

```
int fwrite(  void *    out_location_ptr,     // ptr to data to output
             int       element_size,         // size of data to output
             int       count,                // # items to output
             FILE *    stream_ptr);          // stream pointer
```

returns the # of items written

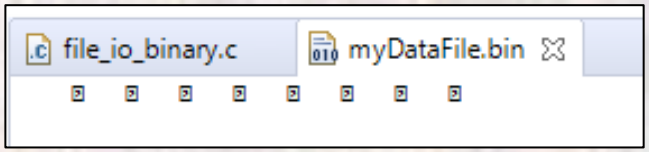
All elements must be the same size

File I/O - Binary

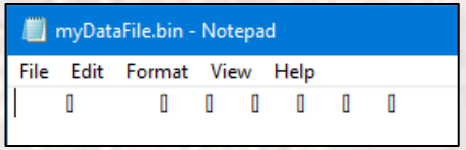
- Write a series of integers to a file

```c
/*
 * file_io_binary.c
 *
 *  Created on: Jun 4, 2024
 *      Author: johnsontimoj
 */
//////////////////////
//
// write integers to a data file - binary format
//
//////////////////////
#include <stdio.h>
#include <stdlib.h>

int main(void){
   setbuf(stdout, NULL);  // disable buffering

   // create a stream pointer for the file
   FILE * DataFile_strm_ptr;

   //create a new file
   if((DataFile_strm_ptr = fopen("myDataFile.bin", "wb")) == NULL){
        printf("Error opening file myDataFile.bin\n");
        exit (100);    // terminate program
       } // end if

   // write a series of integers - 1 at a time
   int i;
   for(i=0; i<10; i++){
       fwrite(&i, sizeof(int), 1, DataFile_strm_ptr);
   }

   // close the file
   fclose(DataFile_strm_ptr);

   return 0;
} // end main
```

View file in Eclipse



View file in Notepad in Windows




```
          Path: Z:\msoe_current\21_Q2_EE1910\Workspace_V10_EE1910\Class_Cons_Project\myDataFile.bin

          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  00 00 00 00 01 00 00 00 02 00 00 00 03 00 00 00  ................
00000010  04 00 00 00 05 00 00 00 06 00 00 00 07 00 00 00  ................
00000020  08 00 00 00 09 00 00 00                          ........
```

Using Format-Hex in Windows PowerShell

Note: Little Endian

File I/O - Binary

- Write a series of structures to a file

```c
/*
 * file_io_binary.c
 *
 *  Created on: Jun 4, 2024
 *      Author: johnsontimoj
 */
//////////////////////
//
// write structures to a data file - binary format
//
//////////////////////
#include <stdio.h>
#include <stdlib.h>

// structure definitions
// typedef version
typedef struct{
    int id;
    char name[26];
    float gpa;
} student;

int main(void){
   setbuf(stdout, NULL);  // disable buffering

   // create a stream pointer for the file
   FILE * DataFile_strm_ptr;

   //create a new file
   if((DataFile_strm_ptr = fopen("myDataFile.bin", "wb")) == NULL){
      printf("Error opening file myDataFile.bin\n");
      exit (100);    // terminate program
   } // end if
```

```c
   // create some student variables
   student stu1 = {234,
                   "Joe Smith",
                   3.45
   };
   student stu2 = {.gpa=3.2, .name="Sara Jones", .id=222};
   student stu3;

   // create an array to hold the students
   student std_ary[3] = {stu1, stu2, stu3};

   // output the array - 3 structures at a time
   // since we know they are stored sequentially in the array
   fwrite(std_ary, sizeof(student), 3, DataFile_strm_ptr);

   return 0;
}// end main
```

234

222

```
Path: Z:\msoe_current\21_Q2_EE1910\Workspace_V10_EE1910\Class_Cons_Project\myDataFile.bin

           00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   EA 00 00 00 4A 6F 65 20 53 6D 69 74 68 00 00 00  ê...Joe Smith...
00000010   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000020   CD CC 5C 40 DE 00 00 00 53 61 72 61 20 4A 6F 6E  ÍÌ\@Þ...Sara Jon
00000030   65 73 00 00 00 00 00 00 00 00 00 00 00 00 00 00  es..............
00000040   00 00 22 76 CD CC 4C 40 BC FE 61 00 F5 6F 23 76  .."vÍÌL@¼þa.õo#v
00000050   1C 43 28 76 FC FE 61 00 51 65 23 76 08 00 00 00  .C(vüþa.Qe#v....
00000060   FD 6E 22 76 E3 6E 22 76 FE 12 33 5C              ýn"vãn"vþ.3\
```

Using Format-Hex in Windows PowerShell

File I/O - Binary

- Formatting input stream data – fread()
  - File read
  - Block format – no conversions, raw bytes

```
int fread(   void *    in_location_ptr,     // ptr to data to read
             int       element_size,        // size of data to read
             int       count,               // # items to read
             FILE *    stream_ptr);         // stream pointer
```

returns the # of items read

All elements must be the same size

File I/O - Binary

- Read a series of integers from a file

```c
/*
 * file_io_binary.c
 *
 *  Created on: Jun 4, 2024
 *      Author: johnsontimoj
 */
//////////////////////
//
// read integers from a data file - binary format
//
//////////////////////
#include <stdio.h>
#include <stdlib.h>

int main(void){
   setbuf(stdout, NULL);  // disable buffering

   // create a stream pointer for the file
   FILE * DataFile_strm_ptr;

   //open an existing file
   if((DataFile_strm_ptr = fopen("myDataFile.bin", "rb")) == NULL){
        printf("Error opening file myDataFile.bin\n");
        exit (100);    // terminate program
       } // end if

   // create and initialize an array
   int my_array[20];
   int i;
   for(i=0; i<20; i++){
       my_array[i] = 0;
   }
   for(i=0; i<20; i++){
       printf("%i ", my_array[i]);;
   }
   printf("\n");

   // read from the file - 10 at a time
   // since we know they are stored sequentially in the array
   fread(my_array, sizeof(int), 10, DataFile_strm_ptr);

   // print myArray
   for(i=0; i<20; i++){
       printf("%i ", my_array[i]);;
   }

   // close the file
   fclose(DataFile_strm_ptr);

   return 0;
 } // end main
```

Using the integer file from the write example

```
<terminated> (exit value: 0) Class_Cons_Project.e
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 2 3 4 5 6 7 8 9 0 0 0 0 0 0 0 0 0 0
```

File I/O - Binary

- Read a series of integers from a file until the end

```c
/*
 * file_io_binary.c
 *
 *  Created on: Jun 4, 2024
 *      Author: johnsontimoj
 */
//////////////////////
//
// read integers from a data file until none left - binary format
//
//////////////////////
#include <stdio.h>
#include <stdlib.h>

int main(void){
   setbuf(stdout, NULL);  // disable buffering

   // create a stream pointer for the file
   FILE * DataFile_strm_ptr;

   //open an existing file
   if((DataFile_strm_ptr = fopen("myDataFile.bin", "rb")) == NULL){
        printf("Error opening file myDataFile.bin\n");
        exit (100);    // terminate program
       } // end if

   // create and initialize an array
   int my_array[20];
   int i;
   for(i=0; i<20; i++){
       my_array[i] = 0;
   }
   for(i=0; i<20; i++){
       printf("%i ", my_array[i]);;
   }
   printf("\n");

   int tmp_val;
   i = 0;
   // read from the file - ending at the EOF
   while(fread(&tmp_val, sizeof(int), 1, DataFile_strm_ptr) != 0){
       my_array[i++] = tmp_val;
   }

   // print myArray
   for(i=0; i<20; i++){
       printf("%i ", my_array[i]);;
   }

   // close the file
   fclose(DataFile_strm_ptr);

   return 0;
} // end main
```

Using the integer file from the write example

```
<terminated> (exit value: 0) Class_Cons_Project.e
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 2 3 4 5 6 7 8 9 0 0 0 0 0 0 0 0 0 0
```

fread returns the number of things read

at the end of the file nothing is read
and fread returns 0

File I/O - Binary

- Read a structure from a file

```c
/*
 * file_io_binary.c
 *
 *  Created on: Jun 4, 2024
 *      Author: johnsontimoj
 */
//////////////////////
//
// read structures from a data file - binary format
//
//////////////////////
#include <stdio.h>
#include <stdlib.h>

// structure definitions
// typedef version
typedef struct{
    int id;
    char name[26];
    float gpa;
} student;

int main(void){
   setbuf(stdout, NULL);  // disable buffering

   // create a stream pointer for the file
   FILE * DataFile_strm_ptr;

   //create a new file
   if((DataFile_strm_ptr = fopen("myDataFile.bin", "rb")) == NULL){
      printf("Error opening file myDataFile.bin\n");
      exit (100);    // terminate program
   } // end if

   // create an array to hold the students
   student std_ary[3];

   // read into the array
   fread(std_ary, sizeof(student), 3, DataFile_strm_ptr);

   // print structure [1]
   printf("%i %s %f", std_ary[1].id, (*(std_ary+1)).name, (std_ary + 1)->gpa);

   return 0;
}// end main
```

Using the structure file from the write example

```
<terminated> (exit value: 0) Cla
222 Sara Jones 3.200000
```