



Binary Files

Abdallah Karakra

Computer Science Department

Comp 133

Monday, August 24, 2015

Binary Files

- Files containing binary numbers that are the computer's internal representation of each file component.

Advantages of Binary files

- Assume that two bytes are used store an `int` value.
 - 244 (128+64+32+16+4)
 - 11110100
- In text files,
 - Write: 11110100 → 2 4 4 and blank
 - Read: 244 → 11110100
 - It takes more time.
 - It takes more space. (Four bytes versus two)
- Precision

Disadvantages of Binary files

- A binary file created on one computer is rarely readable on another type of computers.
- A binary file can not be created or modified in a word processor.
- `fread` ↔ `fwrite` //binary file
- `fscanf` ↔ `fprintf` //text file

Opening binary files

- Add “b” to the **fopen** mode string
 - “rb” : read a binary file
 - “wb” : write a binary file
 - “ab” : append to a binary file

```
FILE *fp = fopen("myfile.bin", "rb");//read  
FILE *fp = fopen("myfile.bin", "wb");//write
```

Writing to binary files

`size_t` fwrite (const void * `ptr`, size_t `size`, size_t `count`, FILE * stream)

INPUT

- *A ptr to an array of elements (or just one)*
- The size of each element
- The number of elements
- Pointer to a FILE object that specifies an output stream (File pointer)

OUTPUT

- Returns the number of elements written
- If return value is different than count, there was an error

Writing to binary files

```
...  
  
FILE *fp = fopen("myfile.bin", "wb");  
  
...  
  
int nums[] = {1,2,3};  
fwrite(nums, sizeof(int), 3, fp);  
  
double dub = 3.1;  
fwrite(&dub, sizeof(double), 1, fp);  
  
...
```

sizeof operator used to find the number of bytes used for storage of a data type

Reading binary files

```
size_t fread ( void * ptr, size_t size, size_t count, FILE * stream )
```

INPUT

- **A ptr to some memory of size at least (size * count)**
- **The size of each element to read**
- **The number of elements to read**
- **Pointer to a FILE object that specifies an input stream (File Pointer)**

OUTPUT

- **Returns number of elements read**
- **If return value is different than *count*, there was an error or the end of the file was reached**

Reading binary files

```
...  
FILE *fp = fopen("myfile.bin","rb");  
...  
int nr;  
  
int nums[3];  
nr = fread(nums, sizeof(int), 3, fp);  
//Check for errors  
  
double dub;  
nr = fread(&dub, sizeof(double), 1, fp);  
//Check for errors  
  
...
```

sizeof operator used to find the number of bytes used for storage of a data type

Example: Creating a Binary File of Integers

```
1. FILE *binaryp;  
2. int i;  
3.  
4. binaryp = fopen("nums.bin", "wb");  
5.  
6. for (i = 2; i <= 500; i += 2)  
7.     fwrite(&i, sizeof (int), 1, binaryp);  
8.  
9. fclose(binaryp);
```

Example: Writing to a binary file

```
#include <stdio.h>
#define SIZE 100
int main()
{
    int x=20,A[SIZE]={0,1,2,3};
    FILE* fptr_out=fopen("out.bin", "wb");

    fwrite(&x, sizeof(int), 1, fptr_out);
    fwrite(A, sizeof(int), SIZE, fptr_out);
    fclose(fptr_out);
    return 0;
}
```

Example: Reading from a binary file

```
#include <stdio.h>
#define SIZE 100
int main()
{
    int x,A[SIZE];
    FILE* fptr_inp=fopen("in.bin", "rb");

    fread(&x, sizeof(int), 1, fptr_inp);
    fread(A, sizeof(int), SIZE, fptr_inp);
    fclose(fptr_inp);
    return 0;
}
```

Example: writing and reading a complex number

```
#include <stdio.h>
typedef struct {
    int real;
    int imag;
} complex_t;
int main()
{
    complex_t x1={2,3};
    complex_t x2={4,5};
    complex_t x3;
    FILE *fptr_inp;
    FILE *fptr_out;
    fptr_out=fopen("out.bin","wb");//for writing
    fwrite(&x1,sizeof(complex_t),1,fptr_out);
    fclose(fptr_out);
    fptr_inp=fopen("out.bin","rb");//for reading
    fread(&x3,sizeof(complex_t),1,fptr_inp);
    fclose(fptr_inp);
    printf("%d %d\n%d %d",x3.real,x3.imag,x2.real,x2.imag);
    return 0;
}
```

Text File Vs Binary File

TABLE 11.5

Example	Text File I/O	Binary File I/O	Purpose
5	<pre>for (i = 0; i < MAX; ++i) fscanf(doub_txt_inp, "%lf", &nums[i]);</pre>	<pre>fread(nums, sizeof (double), MAX, doub_bin_inp);</pre>	Fill array <code>nums</code> with type <code>double</code> values from input file.
6	<pre>for (i = 0; i < MAX; ++i) fprintf(doub_txt_outp, "%e\n", nums[i]);</pre>	<pre>fwrite(nums, sizeof (double), MAX, doub_bin_outp);</pre>	Write contents of array <code>nums</code> to output file.
7	<pre>n = 0; for (status = fscanf(doub_txt_inp, "%lf", &data); status != EOF && n < MAX; status = fscanf(doub_txt_inp, "%lf", &data)) nums[n++] = data;</pre>	<pre>n = fread(nums, sizeof (double), MAX, doub_bin_inp);</pre>	Fill <code>nums</code> with data until EOF encountered, setting <code>n</code> to the number of values stored.
8	<pre>fclose(plan_txt_inp); fclose(plan_txt_outp); fclose(doub_txt_inp); fclose(doub_txt_outp);</pre>	<pre>fclose(plan_bin_inp); fclose(plan_bin_outp); fclose(doub_bin_inp); fclose(doub_bin_outp);</pre>	Close all input and output files.

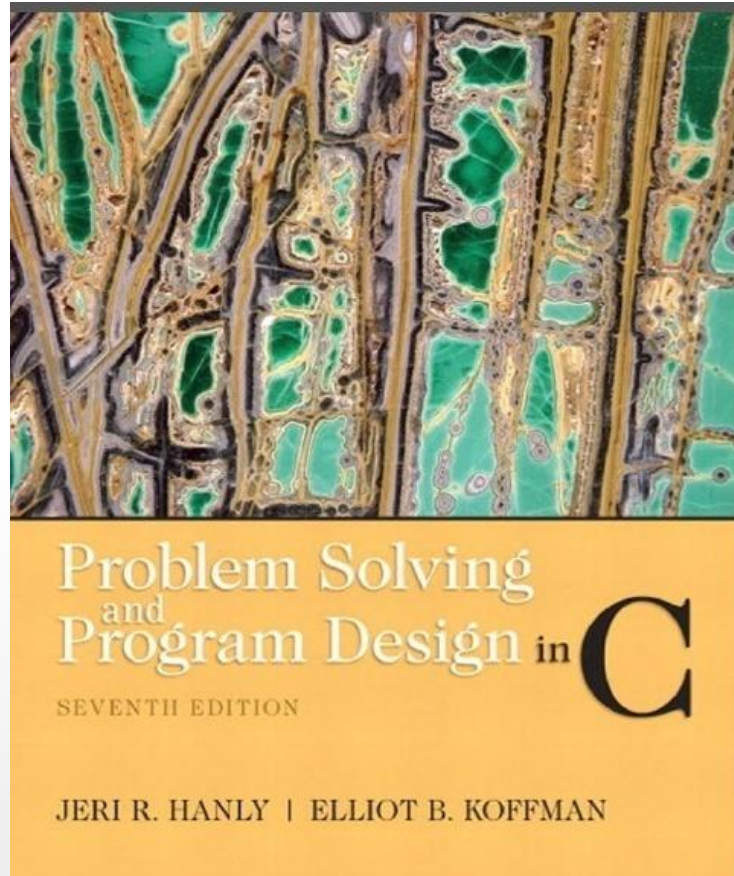
sizeof operator used to find the number of bytes used for storage of a data type

More Examples On board

Question?



“Success is the sum of small efforts, repeated day in and day out.”
Robert Collier



References:

Problem Solving & Program Design in C (main reference)

<http://www.cplusplus.com/reference/cstdio/>