Binary Files in C

What is a Binary File

- Binary files are usually thought of a being a sequence bytes.
  - In fact the data will not be interpreted as a sequence of single characters like in a text file.
  - The data will be stored in the same format and sequence of bytes when used in you program.
    - A variable stored into double on the computer memory will be stored into a binary file in the same order and sequence of bytes.
- Example:
  - double x = 0.00887776665551, will be stored in a an 8-byes memory space. The same data in a text file will be stored in a 16-byte memory space.

What is a Binary File (continued)

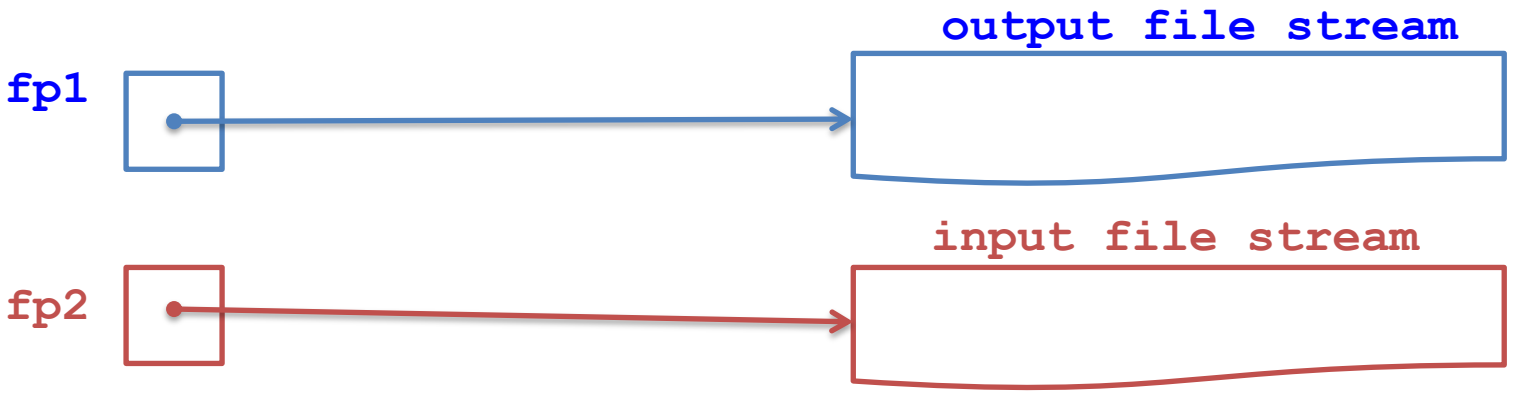
- A binary file is normally more compressed that a text file.
  - Most digital data are stored in binary files
- Reading ad writing data from and into file are faster, using binary data.
- Binary file can be viewed or read properly like a text file using a text editor. Here is an example of a binary file that I opened by an editor on a Mac computer:

oeˆ.¿†8![.[__text__TEXT#.‹a(.__debug_frame__DWARF$|%‰cdebug_info__DWARF†.K`dc__debug_abbrev__DWARF.P.EW__debug_aranges__DWARFT
Z__debug_macinfo__DWARFT‹Z__debug_loc__DWARFT‹Z__debug_pubnames__DWARFT.‹Z__debug_pubtypes__DWARF≥T$s[__debug_str__DWARF◊T.[__debug_ranges__DWARF◊T.[__data__DATA◊T.[__StaIcInit__TEXT‡T{†[‹d.__bss__DATA[__cstring__TEXT[UCÄ__mod_init_func__DATA†U`Ä

Opening files in binary mode

- Us fopen in the following format:

```
FILE* fp1, fp2;
fp1 = fopen("output.bin", "wb");
fp2 = fopen("input.bin", "rb");
```



How to write into a binary file

- You can use **fwrite**, to write any data into the output stream --In our example into: ***output.bin.***
- Here is the prototype of the fwrite library function:

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

- You can write the values of a double value into the file **output.bin** as follows

```
int n;
double b = 4.5
n = fwrit(&b, sizeof(double),1, fp1);
```

- fwrite returns the number items successfully written into the stream. In this case n will be 1, if it is successfully written into the stream.
- You need to close the file when writing is done.

How to read from a binary file

- You can use **fread**, to read any data from input stream --In our example from: **input.bin.**
- Here is the prototype of the fwrite library function:

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

- For example you can read a double value from the file **output.bin** as follows

```
int n;
double b;
n = fread(&b, sizeof(double), 1, fp2);
```

- fread returns the number items successfully read from stream. In this case n will be 1.
- You need to close the file when reading is done.

Example – Writing Data into a Binary File

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 5
int main() {
   const char* outfile = “/usres/mydir/myoutput.bin”;
   int a[SIZE] = {2543, 465, 100, 300, 600};
   FILE    *outp ;
   outp = fopen(outfile, “wb”);

   if (outp == NULL){
    fprintf (stderr, "Error: cannot open the file %s: “, outfile);
     exit(1);
   }

   fwrite(a, sizeof(a), 1, outp);
   fclose(outp);
   return 0;
}
```

The same program can be written in a different way:

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 5
int main() {
   const char* outfile = "/usres/mydir/myoutput.bin";
   int a[SIZE] = {2543, 465, 100, 300, 600};
   FILE    *outp ;
   outp = fopen(outfile, "wb");

   if (outp == NULL){
      fprintf (stderr, "Error: cannot open the file %s: ", outfile);
      exit(1);
   }

   for(int j = 0; j < SIZE; j++)
      fwrite(&a[i], sizeof(int), 1, outp);

  fclose(outp);
   return 0;
}
```

Random Access to the File

- Library function fseek allows us to set the file position indicator for the stream to an offset position.

  int fseek(FILE *stream, long offset, int origin);

  - Return value: 0 upon success, nonzero value otherwise.
- Sets the file position indicator for the file stream stream to an offset position from origin.
- Origin can be set to:
  - **SEEK_SET**
  - **SEEK_CUR**
  - **SEEK_END**
- Library function ftell, allows us to indcate the current value of the position of indicator of the file stream in number of bytes:

  long int ftell (FILE * stream);
- Returns the current value of the position indicator of the stream.