Binary Files (raw bytes)

Up till now we've being playing around with files which contain ASCII text. Can C read files with binary data? Let's find out.

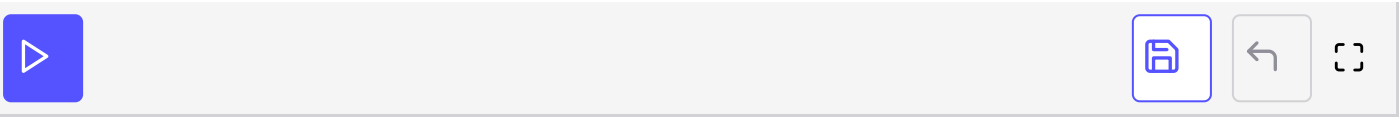
There are many circumstances in which you may want to read from and write to binary files. Binary files are not plain text (ascii) files where each chunk of bytes represents an ascii character. In binary files, you store raw bytes, in whatever format you want. For example Optotrak stores its data files as binary files: a header of a given length (number of bytes) followed by data, in a specific byte format.

Advantages of binary files over ascii files is that they are typically smaller in size, and they can be read from and written to faster (no need to convert between raw bytes and ascii characters). Disadvantages of binary files are that they are not human readable (you can't open in them in a text editor and "look" at them).

The `fread()` and `fwrite()` functions are used to read and write binary data (raw bytes) from and to binary files. Here is an example of writing some data to a binary file. We first write a 16 byte header containing the date (4 + 4 + 4 = 12 bytes) and the number of data points (4 bytes). We then write out the data array, 4 bytes per element. In this example the data are integer values.

```
#include <stdio.h>

int main(int argc, char *argv[]) {

  FILE *fp;
  int year = 2012;
  int month = 8;
  int day = 26;
  int mydata[5] = {2, 4, 6, 8, 10};

  fp = fopen("output/data.bin", "w");
  if (fp == NULL) {
    printf("error opening data.bin\n");
    return 1;
  }
  else {
    // write out the header
    int bytesout;
    bytesout = fwrite(&year, sizeof(year), 1, fp);
    bytesout = fwrite(&month, sizeof(month), 1, fp);
    bytesout = fwrite(&day, sizeof(day), 1, fp);
    // write the data
```

```
    bytesout = fwrite(mydata, sizeof(int), 5, fp);
    fclose(fp);
  }


  return 0;
}
```



Here is an example program to read from the binary data file:

```
#include <stdio.h>

int main(int argc, char *argv[]) {

  FILE *fp;
  int bytesread;
  int yy, mm, dd;
  int thedata[5];

  fp = fopen("data.bin", "r");
  if (fp == NULL) {
    printf("error opening data.bin\n");
    return 1;
  }
  else {
    // read the header
    bytesread = fread(&yy, sizeof(int), 1, fp);
    bytesread = fread(&mm, sizeof(int), 1, fp);
    bytesread = fread(&dd, sizeof(int), 1, fp);
    printf("year=%d, month=%d, day=%d\n", yy, mm, dd);
    // read the data
    bytesread = fread(thedata, sizeof(int), 5, fp);
    printf("data = [%d,%d,%d,%d,%d]\n",
       thedata[0], thedata[1],thedata[2],thedata[3],thedata[4]);
    fclose(fp);
  }

  return 0;
}
```

```
year=2012, month=8, day=26
data = [2,4,6,8,10]
```

Output

The bottom line is, as long as you know what the binary **format** is (that is, how many bytes represent each value) then you can read and write them in “raw” binary using `fread()` and `fwrite()`.

We’ve reached the end of this section. Hopefully, you have a good idea of interacting with the terminal and data files through C. Be sure to take a look at the

exercises and the further reading links ahead.

Next, we'll tackle the C preprocessor and macros.