



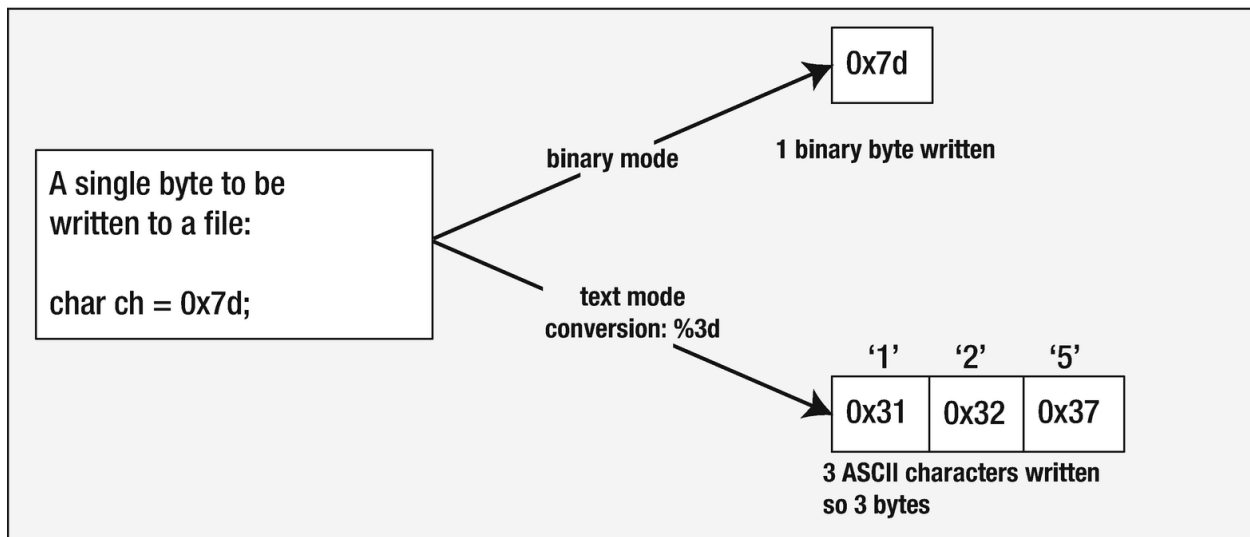
# File: Binary

☰ Tags	
🕒 Created time	@November 21, 2024 12:18 PM
☑ Reviewed	<input type="checkbox"/>

The alternative to text mode operations on a file is binary mode. In this mode, no transformation of the data takes place, and there is no need for a format string to control input or output. It is much simpler than text mode. The binary data is transferred directly into the file as it appears in the memory.

Binary mode has the advantage that no data are transformed or precision lost, as can happen due to the conversion process with text mode. It's also faster than text mode because there's no transformation of data.

Figure below compares the storage of a value `125` in binary and text file. In binary mode, the exact binary data is stored in hexadecimal value with 1 byte written into the file. Meanwhile in text mode, it is transferred to 3 ASCII values: `1, 2, 5` so 3 bytes needed.



# Opening a file in Binary mode

Similar to opening a text file, `fopen` can be used to open a file in binary mode.

Mode	Description
rb	Open a binary file to read to it
rb+ or r+b	Open a binary file to read and write
wb	Open or create a binary file to write to it. If the file exists, its length is truncated to zero so the contents will be overwritten
wb+ or w+b	Truncate an existing binary file to zero length and open it for update. If the file does not exist, create it and open it for updating
ab	Open or create a binary file to append to it. All writes will be placed at the end of the file.
ab+ or a+b	Open or create a binary file for update, with all writes happening at the end of the file

## Writing a Binary File

```
unsigned int fwrite(const void * pdata, unsigned int size, unsigned int nitems, FILE * pfile)
```

- The first parameter is the address of an array of data items to be written. With a parameter type `void*`, any type of array can be passed as the argument to the function.
- The second parameter is the size of an array element
- The third parameter is the number of array elements.
- The last parameter is the pointer to the file stream.
- The integer returned is the number of items written. This will be less than `nitems` if a write error occurs that prevents all of the data from being written. If `size` or `nitems` is 0, nothing is written to the file.

```

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

int main()
{
    FILE *pfile = NULL; // File pointer
    char *filename = "myfile.bin"; //So a binary file has an extension
    pfile = fopen(filename, "wb"); //open a file with a binary mode
    if(!pfile) //check if the open operation work correctly
    {
        printf("Error opening %s for writing. Program terminated\n", filename);
        exit(1);
    }

    int data[] = {7,8,9}; //the array of integers
    int num_items = sizeof(data)/sizeof(int); //calculate the number of items
    int wcount = fwrite(data, sizeof(int), num_items, pfile); //write the data

    fclose(pfile); //close the file
    return 0;
}

```

## Reading a Binary File

```

unsigned int fread(void * pdata, unsigned int size, unsigned int nitems, FILE * pfile)

```

The parameters are the same as for `fwrite()`:

- `pdata` is the address of an array into which the data items are to be read,
- `size` is the number of bytes per item,
- `nitems` is the number of items to be read,

- `pfile` is the file pointer.

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

int main()
{
    FILE *pfile = NULL; // File pointer
    char *filename = "myfile.bin"; //So a binary file has an extension
    pfile = fopen(filename, "rb"); //open a file with a binary mode
    if(!pfile) //check if the open operation work correctly
    {
        printf("Error opening %s for writing. Program terminated\n", filename);
        exit(1);
    }

    int data[] = {0,0,0}; //Create an array to store the data
    int num_items = sizeof(data)/sizeof(int); //Identify the length of the array
    int wcount = fwrite( data, sizeof(int), num_items, pfile); //Write the data to the file
    fclose(pfile); //close the file

    /* print the array to see if the file has been read correctly */
    for(int i = 0; i < num_items; ++i)
    {
        printf("%d ", data[i]);
    }

    return 0;
}
```



Write a program to store detail of a student into a binary file. The program then reads the file and display the result.

```
#include<stdio.h>
#include<string.h>

typedef struct Student Student;

struct Student{
    char name[50];
    char college[40];
    int age;
    float grade;
};

int main()
{
    FILE *pfile = NULL;
    char *filename = "studentBinary.bin";
    pfile = fopen(filename, "wb");
    if(!pfile)
        printf("Failed to open %s.\n", filename);

    Student s;

    //Enter student detail from keyboard
    printf("Name: \n");
    scanf("%s", s.name);
    printf("College: \n");
    scanf("%s", s.college);
    printf("Age: \n");
    scanf("%d", &s.age);
    printf("Grade: \n");
```

```

scanf("%f", &s.grade);

//write the data into the binary file:
int wcount1 = fwrite(s.name, 1, strlen(s.name), pfile); //w
int wcount2 = fwrite(s.college, 1, strlen(s.college), pfile);
int wcount3 = fwrite(&s.age, sizeof(int), 1, pfile); //write
int wcount4 = fwrite(&s.grade, sizeof(float), 1, pfile); //w

fclose(pfile);

//Read the binary file
pfile = fopen(filename, "rb"); //Now the mode is read binary
if(!pfile)
    printf("Failed to open %s.\n", filename);

Student s2;

int rcount1 = fread(s2.name, sizeof(char), strlen(s.name), pfile);
int rcount2 = fread(s2.college, sizeof(char), strlen(s.college), pfile);
int rcount3 = fread(&s2.age, sizeof(int), 1, pfile);
int rcount4 = fread(&s2.grade, sizeof(float), 1, pfile);

fclose(pfile);

printf("The following student information is found in the file:\n");
printf("Name: %s\n", s2.name);
printf("College: %s\n", s2.college);
printf("Age: %d\n", s2.age);
printf("Grade: %f\n", s2.grade);

return 0;
}

```