

File Functions

<code>int fscanf(FILE <i>stream</i>, const char <i>format</i>, ...)</code>	reads formatted input from a stream.
<code>int fprintf(FILE <i>stream</i>, const char <i>format</i>, ...)</code>	sends formatted output to a stream.
<code>FILE <i>fopen</i>(const char <i>filename</i>, const char <i>mode</i>)</code>	opens the filename pointed to, by filename using the given mode. "r" Opens a file for reading. The file must exist. "w" Creates an empty file for writing. If a file with the same name already exists, its content is erased and the file is considered as a new empty file. "a" Appends to a file. Writing operations, append data at the end of the file. The file is created if it does not exist.
<code>int fseek(FILE <i>*f</i>, long int <i>offset</i>, int <i>origin</i>)</code>	go forward offset times without reading the files origins: SEEK_SET - from start of file SEEK_END - end of file SEEK_CUR - move from current location
<code>int ftell(FILE <i>*f</i>)</code>	returns the distance from cursor to start of file
<code>void rewind(FILE <i>*f</i>)</code>	go back to start of file
<code>int ferror(FILE <i>*F</i>)</code>	returns 0 if no errors occurred
<code>fclose(FILE <i>*F</i>)</code>	closes the file
<code>size_t fread(void <i>ptr</i>, size_t <i>size</i>, size_t <i>nmemb</i>, FILE <i>stream</i>)</code>	reads data from the given stream into the array pointed to, by ptr.
<code>int fgetc(FILE <i>*stream</i>)</code>	Gets the next character (an unsigned char) from the specified stream and advances the position indicator for the stream.
<code>char <i>fgets</i>(char <i>str</i>, int <i>n</i>, FILE <i>*stream</i>)</code>	read line to str. stop when newline or eof is read
<code>int fputc(int <i>char</i>, FILE <i>*stream</i>)</code>	Writes a character specified by the argument char to the specified stream and advances the position indicator for the stream.
<code>int fputs(const char <i>str</i>, FILE <i>stream</i>)</code>	Writes a string to the specified stream up to but not including the null character.

Reading and writing binary files

Reading file into struct (cont)

```
>   people = realloc(people, sizeof(person)(++age)size);
    strcpy((people+size-1)->name, name);
    (people+size-1)->age = age;
    }
    fclose(f);
}
```

assume people.txt is formatted like this:

```
name1 age
name2 age
```

Writing to file

```

#include <stdio.h>
#define SIZE 20
struct Person{
char name[SIZE];
long id;
float age;
} typedef person_t;
void main(){
person_t p1={"mo mo", 1111,
23.5}, p2 = {"go go", 2222,
24.8}, p3, p4;
FILE* f = fopen( " per son s.b -
in", " wb");
fwrite (&p1, sizeof (pe rso n_t), 1, f);
fwrite (&p2, sizeof (pe rso n_t), 1, f);
fclose(f);
f = fopen( " per son s.b in",
" rb");
fread( &p3, sizeof (pe rso n_t),
1, f);
fread( &p4, sizeof (pe rso n_t),
1, f);
fclose(f);
printf ("p3: name: %s\t id:
%ld\t age: %.2f\n ", p3.name,
p3.id, p3.age);
printf ("p4: name: %s\t id:
%ld\t age: %.2f\n ", p4.name,
p4.id, p4.age);
}

```

```

#include <stdio.h>
void main()
{
FILE* f = fopen( " myF ile.tx t", " w");
int res;
if (f == NULL){
printf ("Failed opening the
file. Exitin g! \n");
return;
}
fputs( " Hello World! \n", f);
fclose(f);
f = fopen( " myF ile.tx t", " -
a");
fputs( "And Good Mornin g! \n",
f);
fclose(f);
}

```

Reading file into struct

```

typedef struct person {
char name[20];
int age;
} person;
int main() {
person *people = NULL;
char name[20];
int size, age = 0;
FILE *f = fopen( " peo -
ple.tx t", " r");
while (!feof (f)){
scanf( "%s %d",
name, &age);
}
}

```



By Interesting

cheatography.com/interesting/

Not published yet.

Last updated 16th April, 2024.

Page 2 of 2.

Sponsored by [Readable.com](https://readable.com)

Measure your website readability!

<https://readable.com>