CH-230-A

Programming in C and C++

C/C++

Tutorial 8

Dr. Kinga Lipskoch

Fall 2021

Reading/Writing

Prototype	**Use**
`int getc(FILE *fp)`	Returns next `char` from `fp`
`int putc(int c, FILE *fp)`	Writes a `char` to `fp`
`int fscanf(FILE* fp, char * format, ...)`	Gets data from `fp` according to the format string
`int fprintf(FILE* fp, char * format, ...)`	Outputs data to `fp` according to the format string

Line Input and Line Output

```
char *fgets(char *line, int max, FILE *fp);
```

- Already seen with `stdin`
- Used for files as well

```
int fputs(char *line, FILE *fp);
```

- Outputs/writes a string to a file

Files: Example 1

```c
#include <stdio.h>
#include <stdlib.h>
int main() {
  char ch;
  FILE *fp;
  fp = fopen("file.txt", "r");
  if (fp == NULL) {
    printf("Cannot open file!\n");
    exit(1);
  }
  ch = getc(fp);
  while (ch != EOF) {
    putchar(ch);
    ch = getc(fp);
  }
  fclose(fp);
  return 0;
}
```

Files: Example 2

```
# include <stdio.h>
# include <stdlib.h>
int main () {
  char ch;
  FILE * fp;
  fp = fopen("file.txt", "r") ;
  if (fp == NULL) {
    printf("Cannot open file!\n");
    exit(1);
  }
  while((ch=getc(fp))!=EOF) {
    putchar(ch);
  }
  fclose(fp);
  return 0;
}
```

Files: Example 3

```c
# include <stdio.h>
# include <stdlib.h>
int main () {
  char ch;
  FILE * fp;
  fp = fopen("file.txt", "r") ;
  if (fp == NULL) {
    printf("Cannot open file!\n");
    exit(1);
  }
  while(!feof(fp)) {
    ch=getc(fp);
    if (ch!=EOF)
      putchar(ch);
  }
  fclose(fp);
  return 0;
}
```

`fflush()`, `feof()`, `ferror()`

- `int fflush(FILE *stream)` flushes the output buffer of a stream
    - `fflush_ex.c`
- `int feof(FILE *stream)` tests the end-of-file indicator for the given stream
    - `feof_ex.c`
    - `myfile.txt`
- `int ferror(FILE *stream)` tests the error indicator for the given stream
    - `ferror_ex.c`

`fseek()` and `ftell()`

- Enables to use a file just like an array and move directly to a specific byte in a file that has been opened via `fopen()`
- `ftell()` returns current position of file pointer as a `long` value

`fseek(fp, offset, mode)`

- `fp` is a file pointer, points to file via `fopen()`
- `offset` is how far to move (in bytes) from the reference point
- `mode` specifies the reference point

Mode	measure offset from
`SEEK_SET`	beginning of file
`SEEK_CUR`	current position
`SEEK_END`	end of file

Examples

- `fseek(fp, 0L, SEEK_END);`
  - set position to offset of 0 bytes from file end therefore set position to end of file
- `long last = ftell(fp);`
  - assigns to `last` the number of bytes from the beginning to end of file

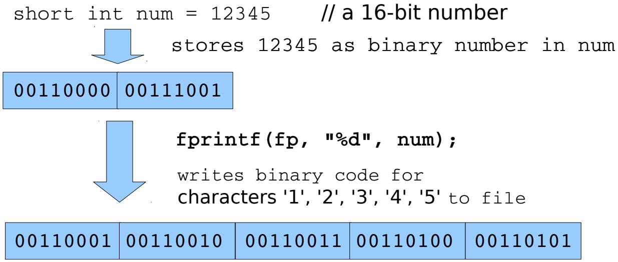
Binary I/O

- `fread()` and `fwrite()`
- Standard I/O is text-oriented
  - Characters and strings
- How to save a double
  - Possible as string but also other

    ```
    double num = 1/3.0;
    fprintf(fp, "%lf", num);
    ```
- Most accurate way would be to store the bit pattern that program internally uses
- Called binary when data is stored in representation the program uses

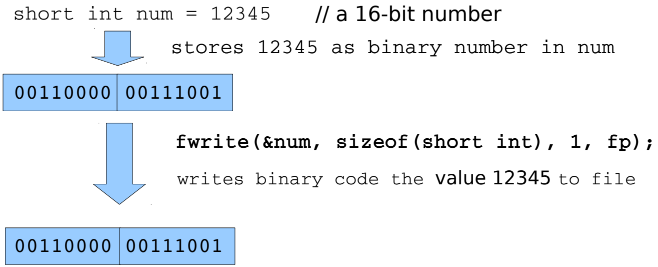
I/O as Text

- All data is stored in binary form
- But for text, data is interpreted as characters


short int num = 12345 // a 16-bit number
stores 12345 as binary number in num
00110000 00111001
fprintf(fp, "%d", num);
writes binary code for
characters '1', '2', '3', '4', '5' to file
00110001 00110010 00110011 00110100 00110101

I/O as Binary

If data is interpreted as numeric data in binary form, data is stored as binary


short int num = 12345 // a 16-bit number
stores 12345 as binary number in num
00110000 00111001
fwrite(&num, sizeof(short int), 1, fp);
writes binary code the value 12345 to file
00110000 00111001

fwrite() (1)

- `size_t fwrite(void *ptr, size_t size, size_t nmemb, FILE *fp)`
- Writes binary data to a file
- `size_t` - type is type that `sizeof()` returns, typically `unsigned int`
- `ptr` - address of chunk of data to be written
- `size` - size in bytes of one chunk
- `nmemb` - number of chunks to be written
- `fp` - file pointer to write to

fwrite() (2)

```c
char buffer[256];
fwrite(buffer, 256, 1, fp);
```

- Writes 256 of bytes to the file

```c
double price[10];
fwrite(price, sizeof(double), 10, fp);
```

- Writes data from the price array to the file in 10 chunks each of size `double`
- Return number of items successfully written, may be less if write error

fread()

- `size_t fread(void *ptr, size_t size, size_t nmemb, FILE *fp)`
- Takes same set of arguments that `fwrite()` does
- `ptr` pointer to which data is read to

```
double price[10];
fread(price, sizeof(double), 10, fp);
```

- Reads `10` size double values into the `price` array
- Returns number of items read, maybe less if read error or end of file reached