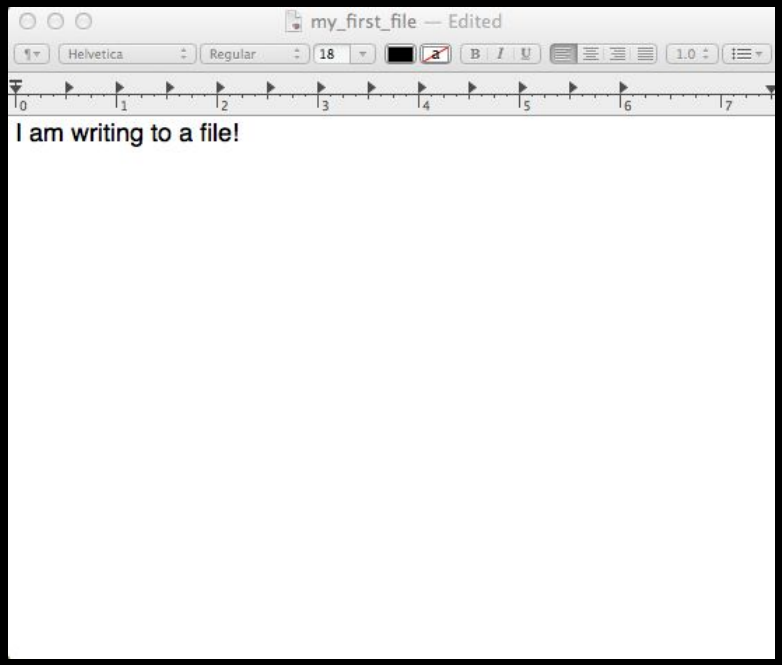
File I/O

We are used to reading from and writing to the terminal:

- read from `stdin`
- write to `stdout`



But we can also read from and write to files!

Step 1: Create a reference to the file

```
FILE* file;
```

Step 2: Open the file

```
file = fopen("file.txt", "r");
```

- 1st argument -- path to the file
- 2nd argument -- mode
  - "r" -- read, "w" -- write, "a" -- append

Step 3a: Read from the file

- `fgetc` -- returns the next character
- `fgets` -- returns a line of text
- `fread` -- reads a certain # of bytes and places them into an array
- `fseek` -- moves to a certain position

Step 3b: Write to the file

- `fputc` -- write a character
- `fputs` -- returns a line of text
- `fprintf` -- print a formatted output to a file
- `fwrite` -- write an array of bytes to a file

Step 4: Close the file

```
fclose(file);
```

Remember!

- Always open a file before reading from or writing to it
- Always close a file if you open it

Example #1
Writing to a file

```c
#include <stdio.h>

#define STUDENTS 3

int main(void)
{
    int scores[] = { 96, 90, 83 };
    FILE* file = fopen("database", "w");
    if (file != NULL)
    {
        for (int i = 0; i < STUDENTS; i++)
        {
            fprintf(file, "%i\n", scores[i]);
        }
        fclose(file);
    }
}
```

Example #2
What does this program do?

```c
#include <stdio.h>

int main(int argc, char* argv[])
{
    if (argc < 2)
    {
        printf("Usage: cat file [file ...]\n");
        return 1;
    }
    for (int i = 1; i < argc; i++)
    {
        FILE* file = fopen(argv[i], "r");
        if (file == NULL)
        {
            printf("cat: %s: No such file or directory\n", argv[i]);
            return 1;
        }
        for (int c = fgetc(file); c != EOF; c = fgetc(file))
        {
            putchar(c);
        }
        fclose(file);
    }
    return 0;
}
```