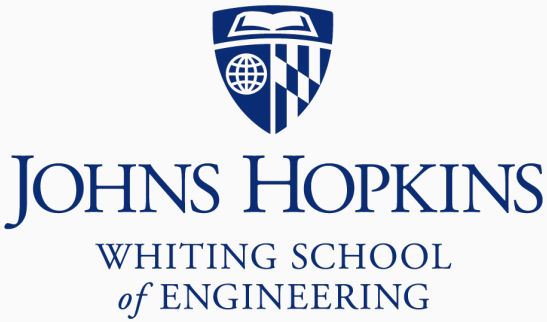
Binary I/O

Ben Langmead

ben.langmead@gmail.com

www.langmead-lab.org



Source markdown available at github.com/BenLangmead/c-cpp-notes

Binary I/O

When reading from a filehandle, we have used `fgets` and `fscanf`

These are for *parsing text*, i.e. turning strings like "0.435" and "999" into `floats`, `ints`, etc

But you can also read and write data (`floats`, `ints`, `structs` etc) as *binary* data, without turning them into strings first

For midterm project, we will read and write PPM files, using binary I/O

fwrite

```
size_t fwrite(const void *data, size_t size,
              size_t nitems, FILE *stream);
```

- `data`: pointer to data to be written
- `size`: size of 1 item
- `nitems`: # items to write
- `stream`: filehandle to write to
    - Must have been opened in "`wb`" mode

Returns the number of items successfully written

- If return value `!= nitems`, there was an error; check `ferror`

fread

```
size_t fread(void *data, size_t size,
             size_t nitems, FILE * stream);
```

- data: data should be copied to here
  - There needs to be enough space! `size * nitems` bytes
- `size`: size of 1 item
- `nitems`: # items to read
- `stream`: filehandle to read from
  - Must have been opened in "rb" mode

Returns the number of items successfully read

- If return value `!= nitems`, there was an error; check `ferror`

Binary I/O

```c
#include <stdio.h> // printf, fread & fwrite
#include <assert.h>

int main() {
    int evens[] = {2, 4, 6, 8}, odds[] = {1, 3, 5, 7};

    FILE *out = fopen("bio_eg1.bin", "wb");
    assert(out != NULL);
    size_t nwritten = 0;
    nwritten += fwrite(evens, sizeof(int), 4, out);
    nwritten += fwrite(odds, sizeof(int), 4, out);
    assert(nwritten == 8);
    fclose(out);

    FILE *in = fopen("bio_eg1.bin", "rb");
    assert(in != NULL);
    int buf1[4] = {0}, buf2[4] = {0};
    size_t nread = 0;
    nread += fread(buf1, sizeof(int), 4, in);
    nread += fread(buf2, sizeof(int), 4, in);
    assert(nread == 8);
    fclose(in);

    printf("Even: %d %d %d %d\n", buf1[0], buf1[1], buf1[2], buf1[3]);
    printf("Odds: %d %d %d %d\n", buf2[0], buf2[1], buf2[2], buf2[3]);
    return 0;
}
```

Binary I/O

```
$ gcc bio_eg1.c -std=c99 -pedantic -Wall -Wextra
$ ./a.out
Even: 2 4 6 8
Odds: 1 3 5 7
```