This is CS50

Week 4

Yuliia Zhukovets

Preceptor

yuliia@cs50.harvard.edu

Agenda

- Pointers
- File I/O

Pointers

Variables

```
int calls = 4;
```

Variables

```
int calls = 4;
```

name

calls

Variables

```
int calls = 4;
```

type

calls

Variables

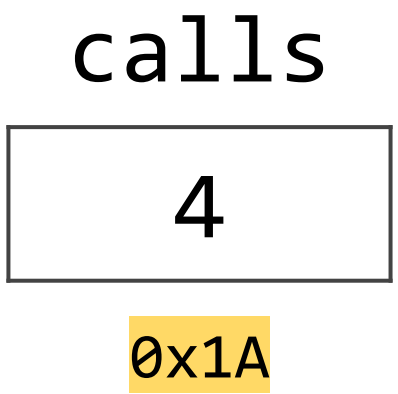
```
int calls = 4;
```

value

calls

Variables

```
int calls = 4;
```



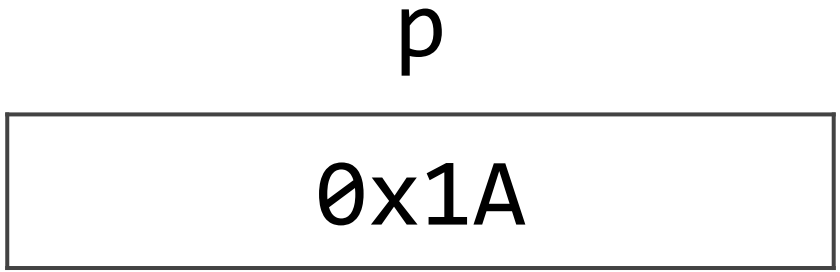
Pointers

```
int *p = 0x1A;
```

Pointers

```
int *p = 0x1A;
```

name



Pointers

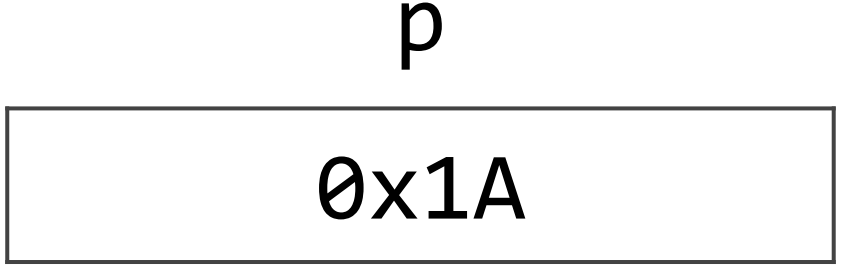
```
int *p = 0x1A;
```

type

Pointers

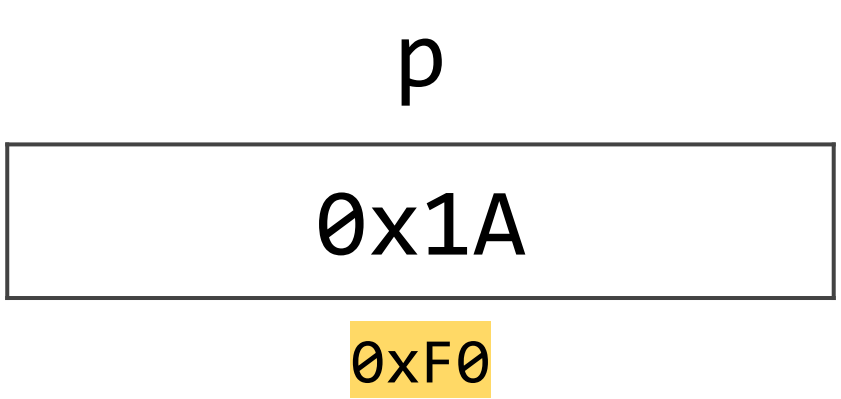
```
int *p = 0x1A;
```

value



Pointers

```
int *p = 0x1A;
```



Pointer Syntax

`calls;`

"value of"

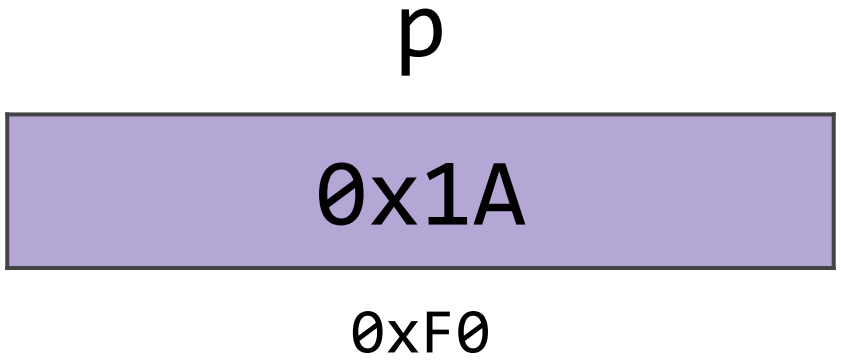
`calls`

`0x1A`

Pointer Syntax

```
p;
```

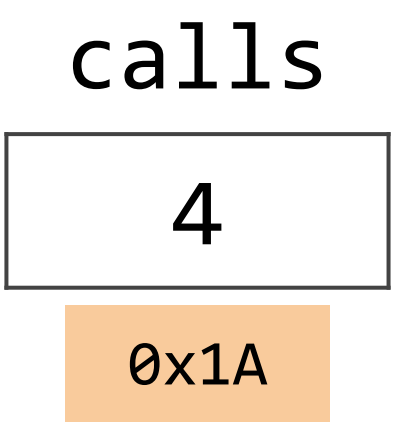
"value of"



Pointer Syntax

`&calls;`

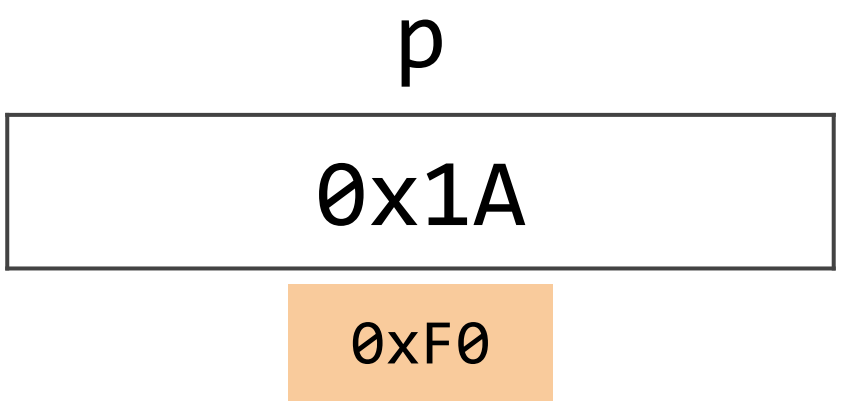
"address of"



Pointer Syntax

`&p;`

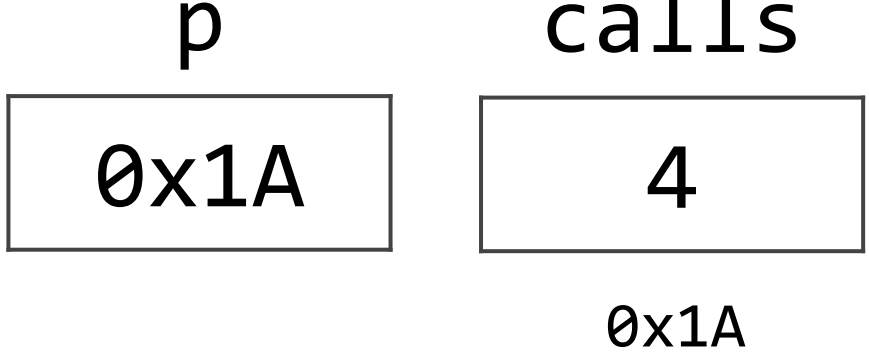
"address of"



Pointer Syntax

```
*p;
```

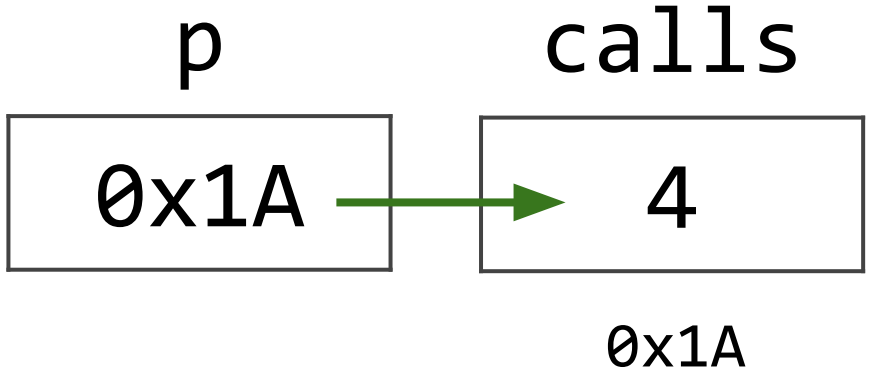
"**go to** the value at address stored in p"



Pointer Syntax

```
*p;
```

"**go to** the value at address stored in p"



`type *` is a pointer that stores the address of a **`type`**.

`*x` takes a pointer **`x`** and goes to the address stored at that pointer.

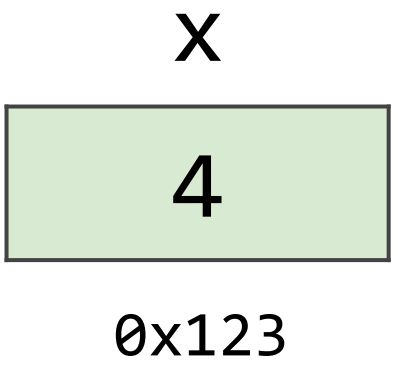
`&x` takes **`x`** and gets its address.

Pointers practice

```
int x = 4;
```

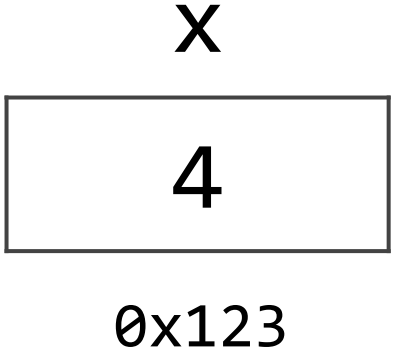
Pointers practice

```
int x = 4;
```



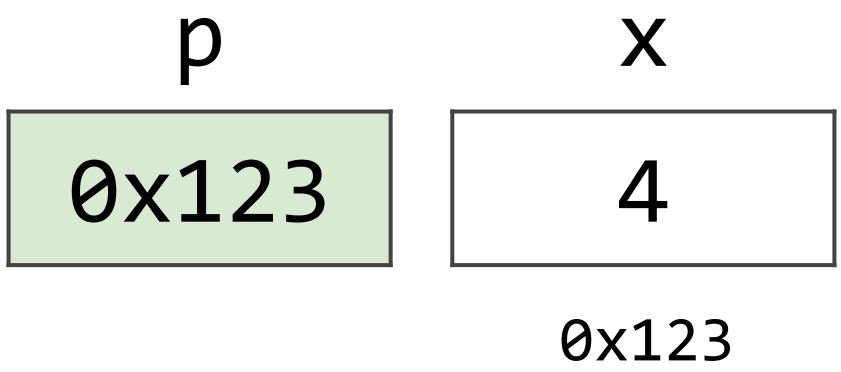
Pointers practice

```
int x = 4;

int *p = &x;
```



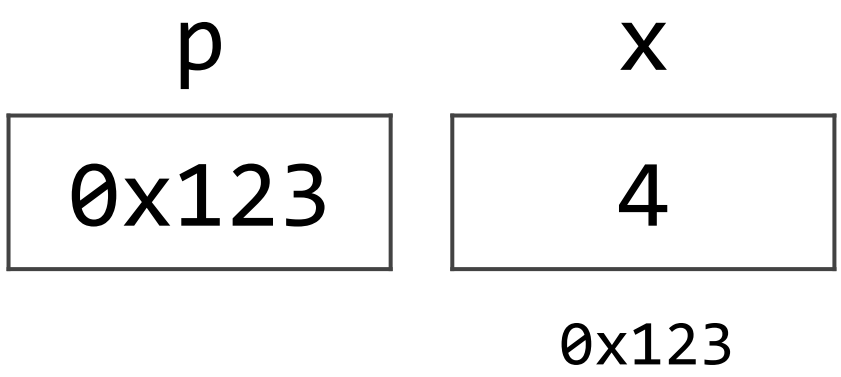
Pointers practice

```
int x = 4;

int *p = &x;
```



Pointers practice

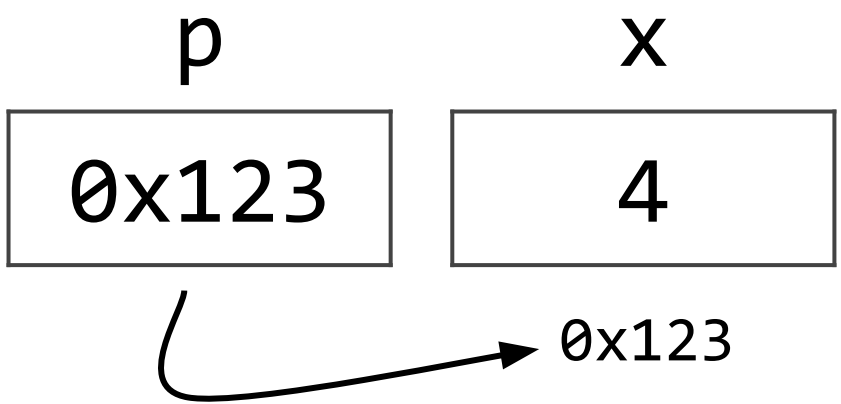
```
int x = 4;
int *p = &x;
printf("%i\n", *p);
```



Terminal:

Pointers practice

```
int x = 4;
int *p = &x;
printf("%i\n", *p);
```

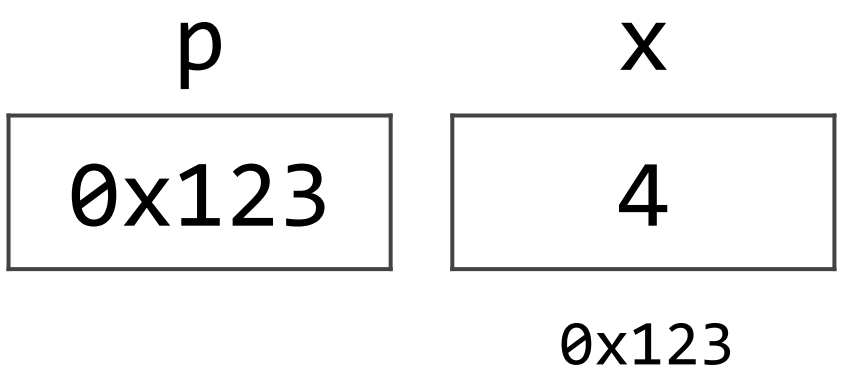


Terminal:

```
4
```

Pointers practice

```
int x = 4;

int *p = &x;

printf("%i\n", *p);
```

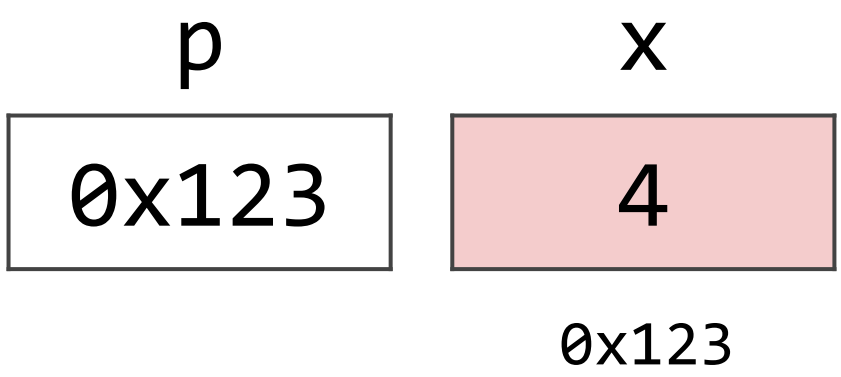



```
*p = 2;
```

```
Terminal:
4
```

Pointers practice

```
int x = 4;

int *p = &x;

printf("%i\n", *p);
```



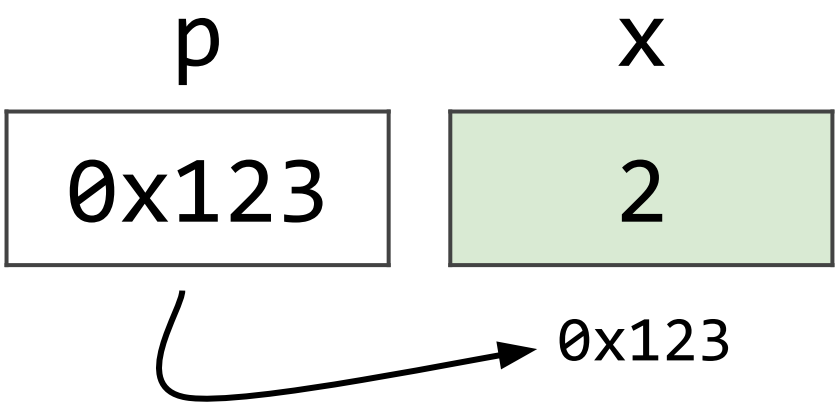

```
*p = 2;
```

```
Terminal:
4
```

Pointers practice

```
int x = 4;

int *p = &x;

printf("%i\n", *p);
```

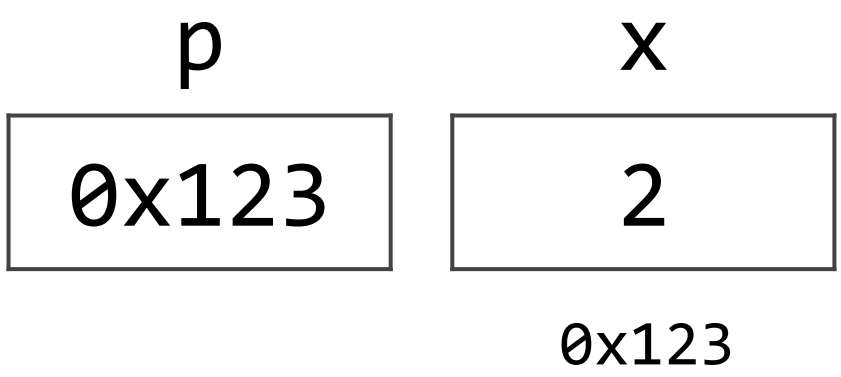
```
*p = 2;
```



Terminal:
4

Pointers practice

```
int x = 4;

int *p = &x;

printf("%i\n", *p);
```

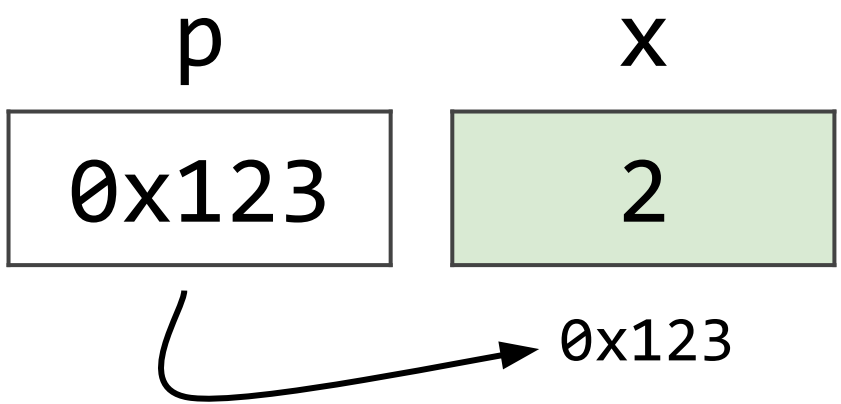



```
*p = 2;

printf("%i\n", *p - 2);
```

Terminal:
4

Pointers practice

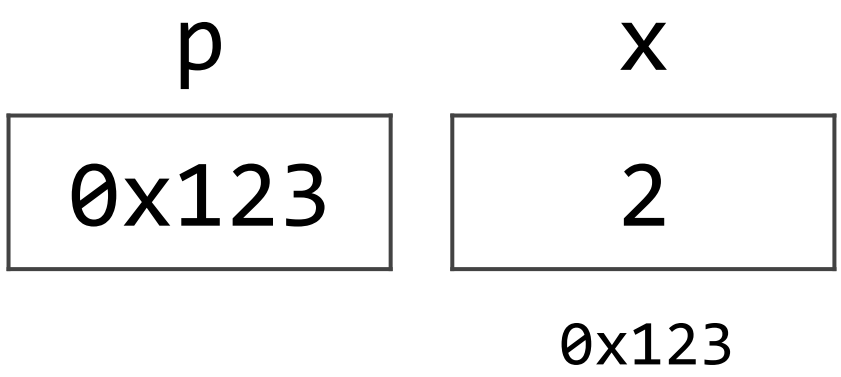
```
int x = 4;

int *p = &x;

printf("%i\n", *p);


*p = 2;

printf("%i\n", *p - 2);
```




```
Terminal:
4
0
```

Pointers practice

```
int x = 4;

int *p = &x;

printf("%i\n", *p);
```

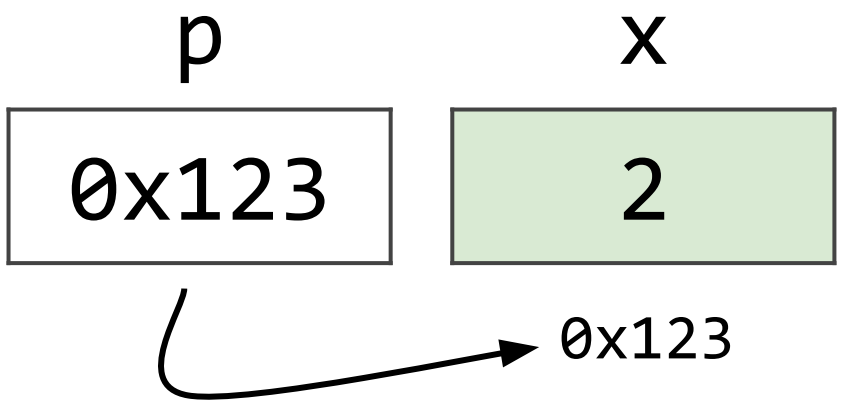



```
*p = 2;

printf("%i\n", *p - 2);

printf("%i\n", x);
```

```
Terminal:
4
0
```

Pointers practice

```
int x = 4;

int *p = &x;

printf("%i\n", *p);
```



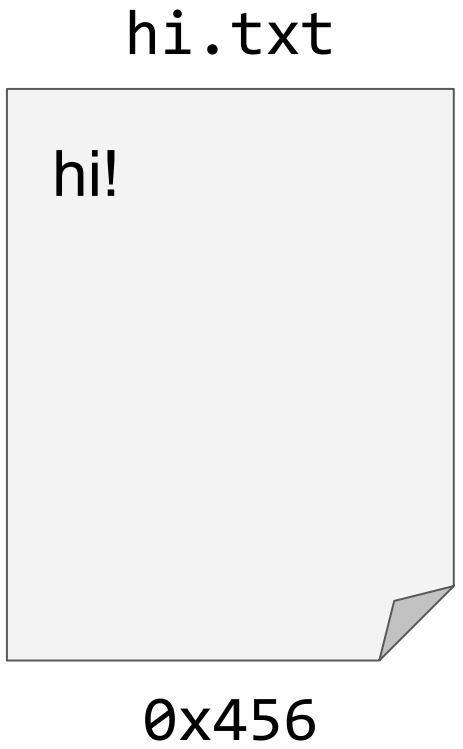

```
*p = 2;

printf("%i\n", *p - 2);

printf("%i\n", x);
```

```
Terminal:
4
0
2
```

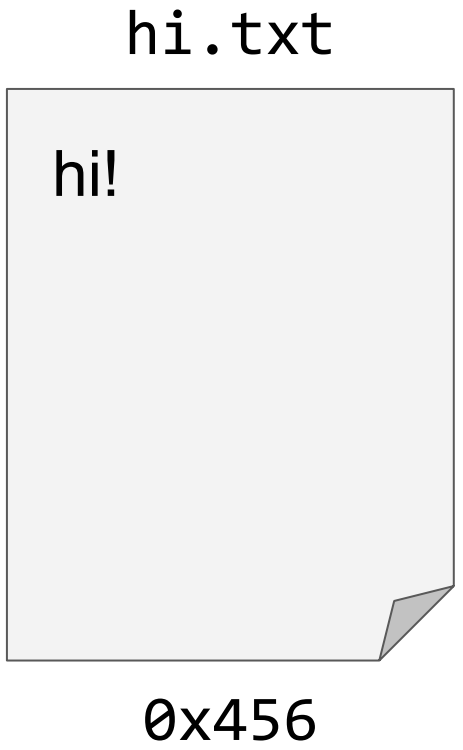
File I/O

- **`fopen`** open a file for future reading/writing
- **`fclose`** closes a file

Always **`fclose`** all the files you **`fopend`**'ed


hi.txt
hi!
0x456

```
FILE *input = fopen("hi.txt", "r");
```

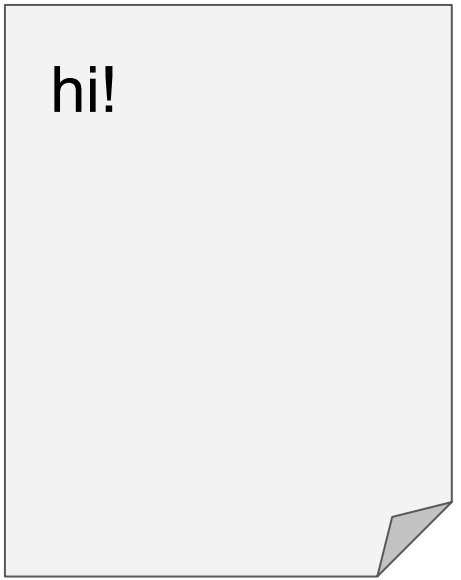


```
FILE *input = fopen("hi.txt", "r");
```

name

input

hi.txt



0x456

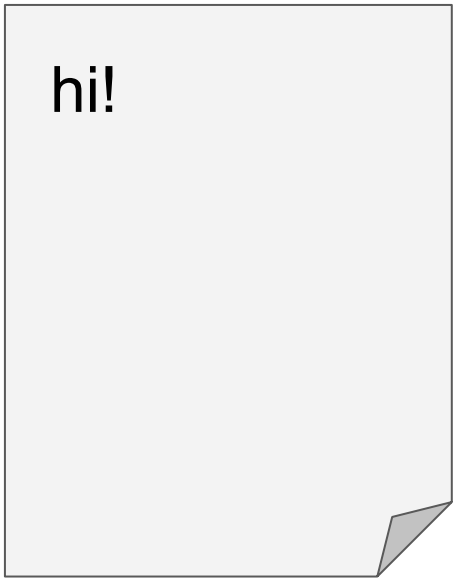
```
FILE *input = fopen("hi.txt", "r");
```

type

input

?

hi.txt



0x456

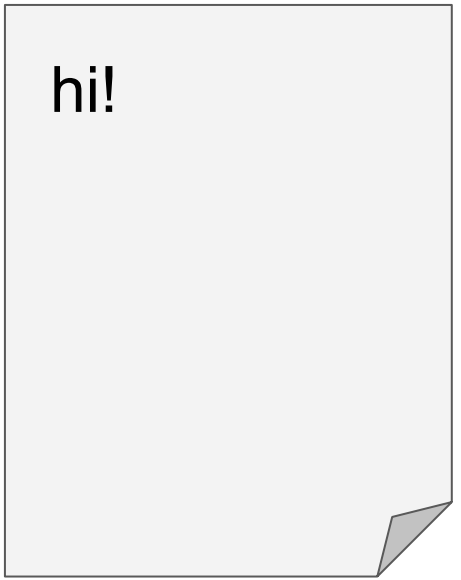
```
FILE *input = fopen("hi.txt", "r");
```

value

hi.txt

input

0x456



0x456

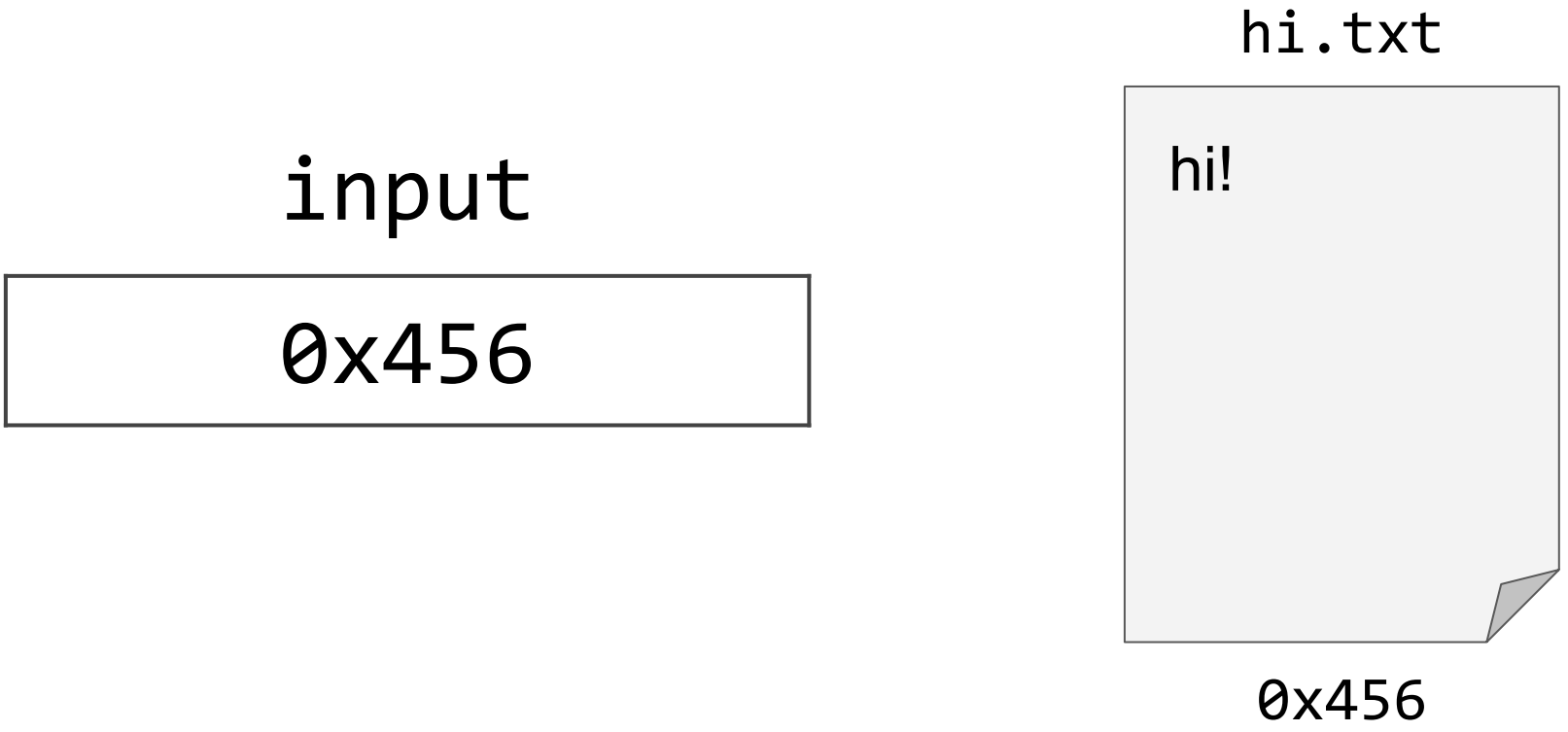
```
FILE *input = fopen("hi.txt", "r");
```

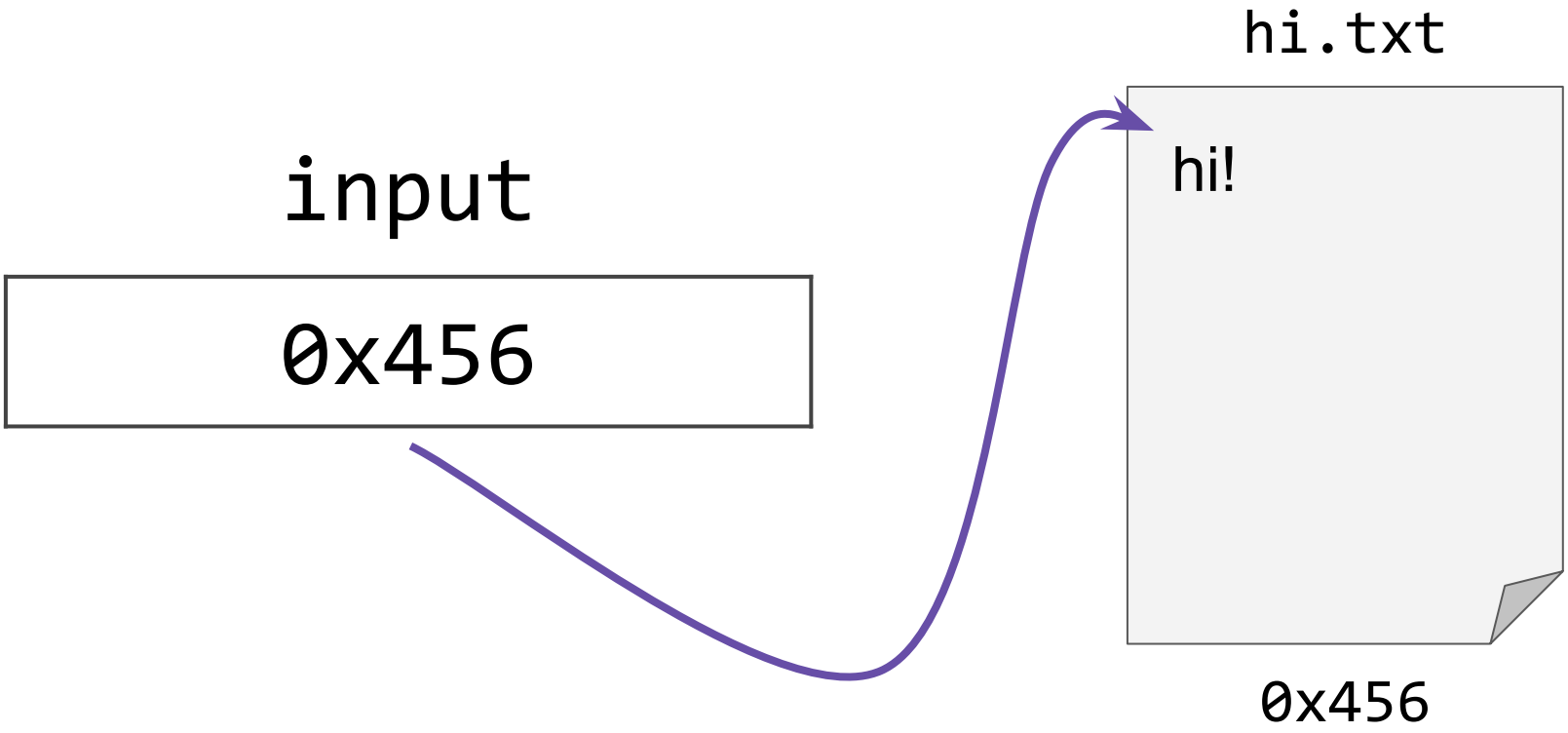
file name

```
FILE *input = fopen("hi.txt", "r");
```

mode

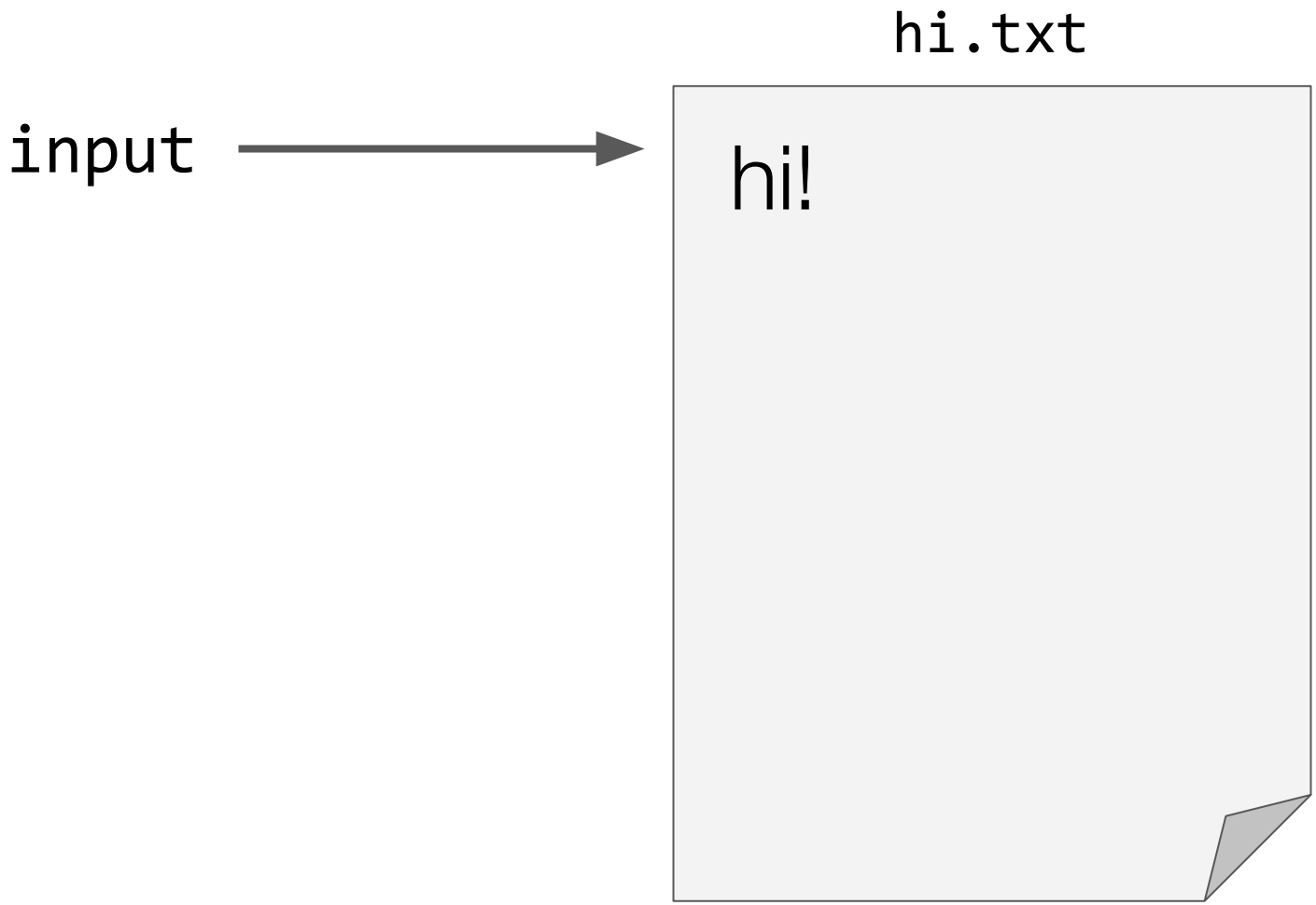
```
FILE *input = fopen("hi.txt", "r");
```

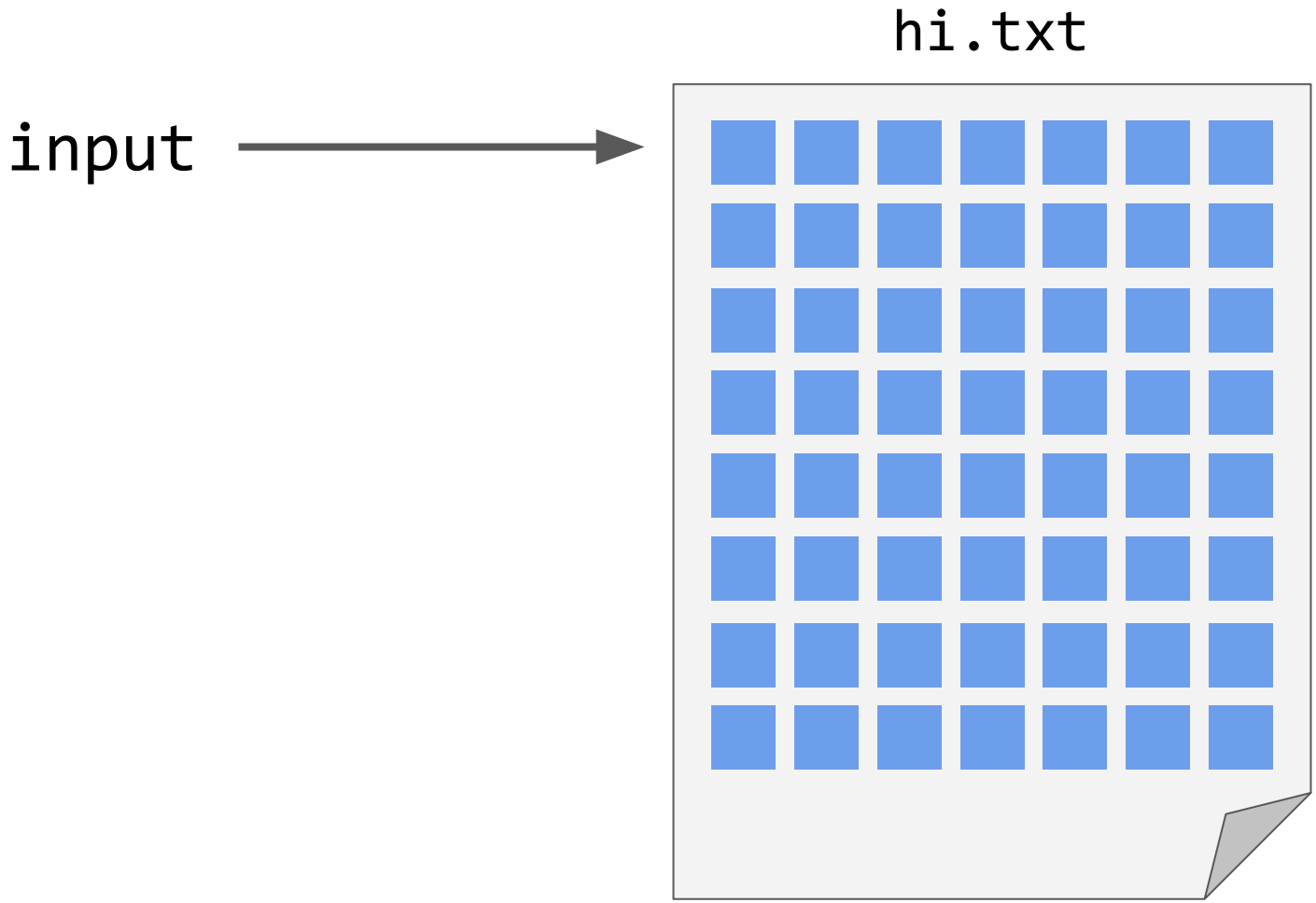



hi.txt
input
hi!
0x456
0x456

- **`fread`** reads data from a file into a buffer
- **`fwrite`** write data from a buffer to a file

A buffer is a chunk of memory that can temporarily store some data from the file.

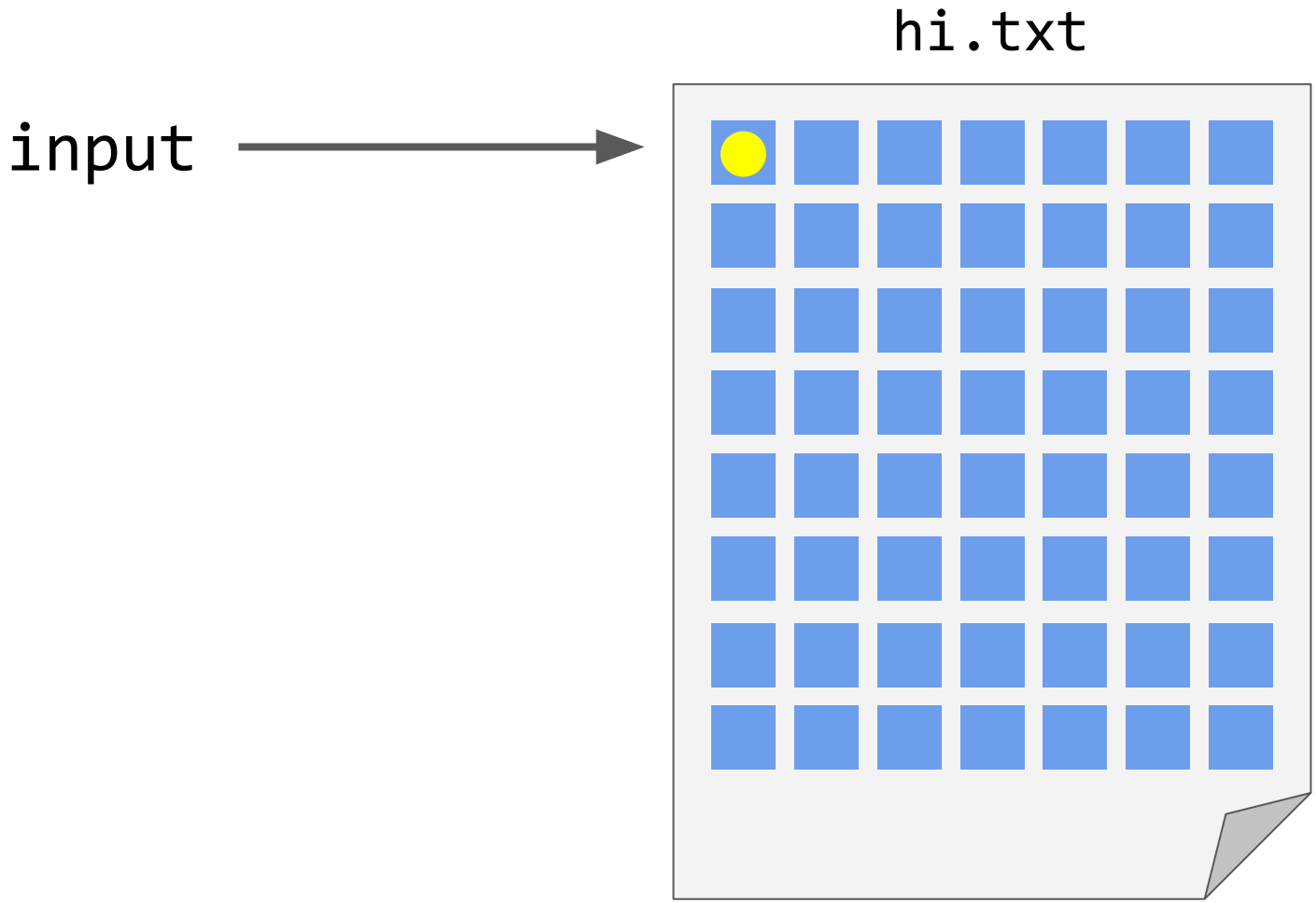

hi.txt
input
hi!


hi.txt
input

```
fread(buffer, 1, 4, input);
```

```
fread(buffer, 1, 4, input);
```

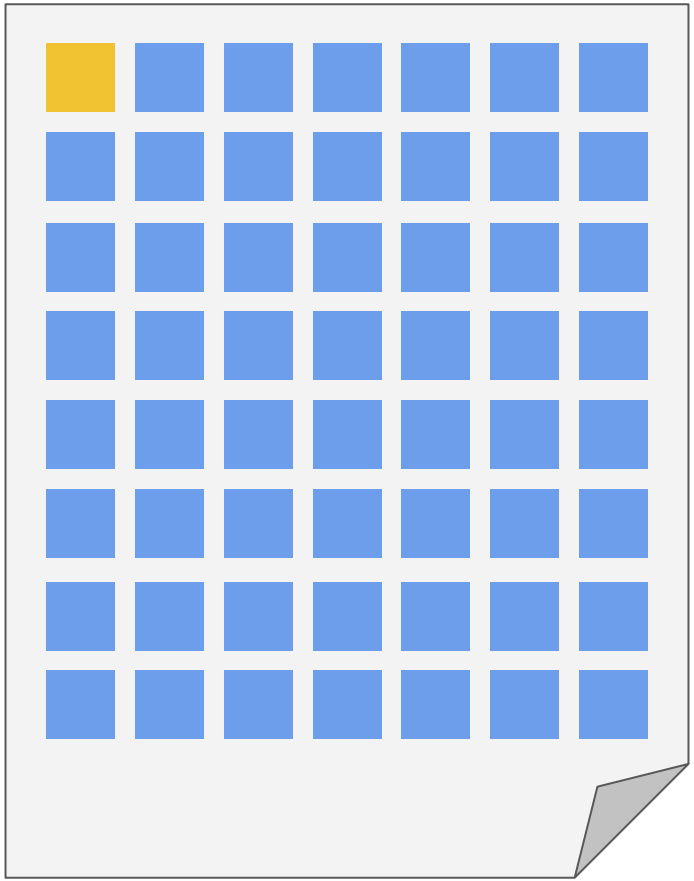
Location to read from


hi.txt
input

```
fread(buffer, 1, 4, input);
```

Size of blocks to read (in bytes)

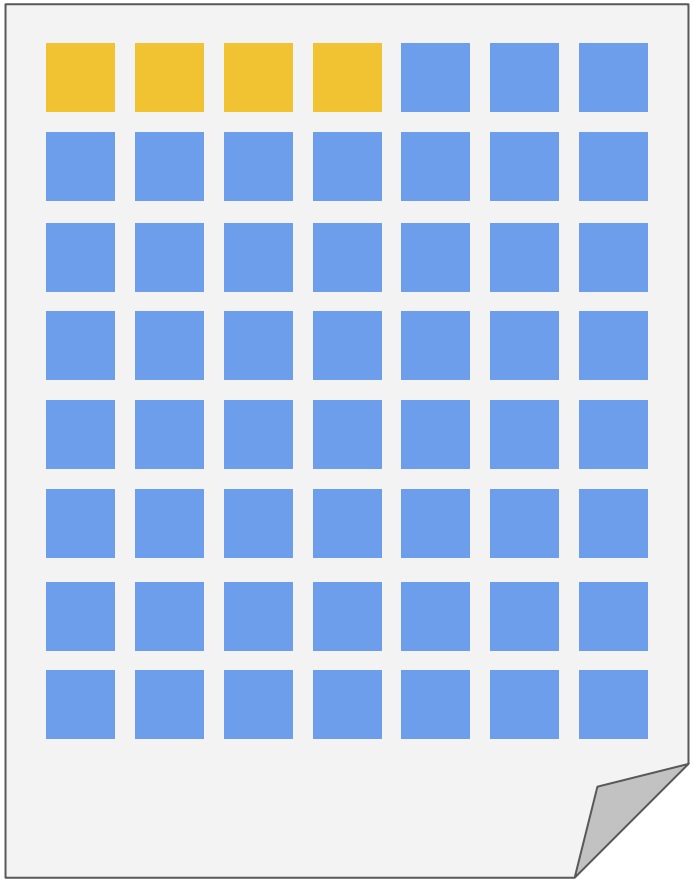
hi.txt



```
fread(buffer, 1, 4, input);
```

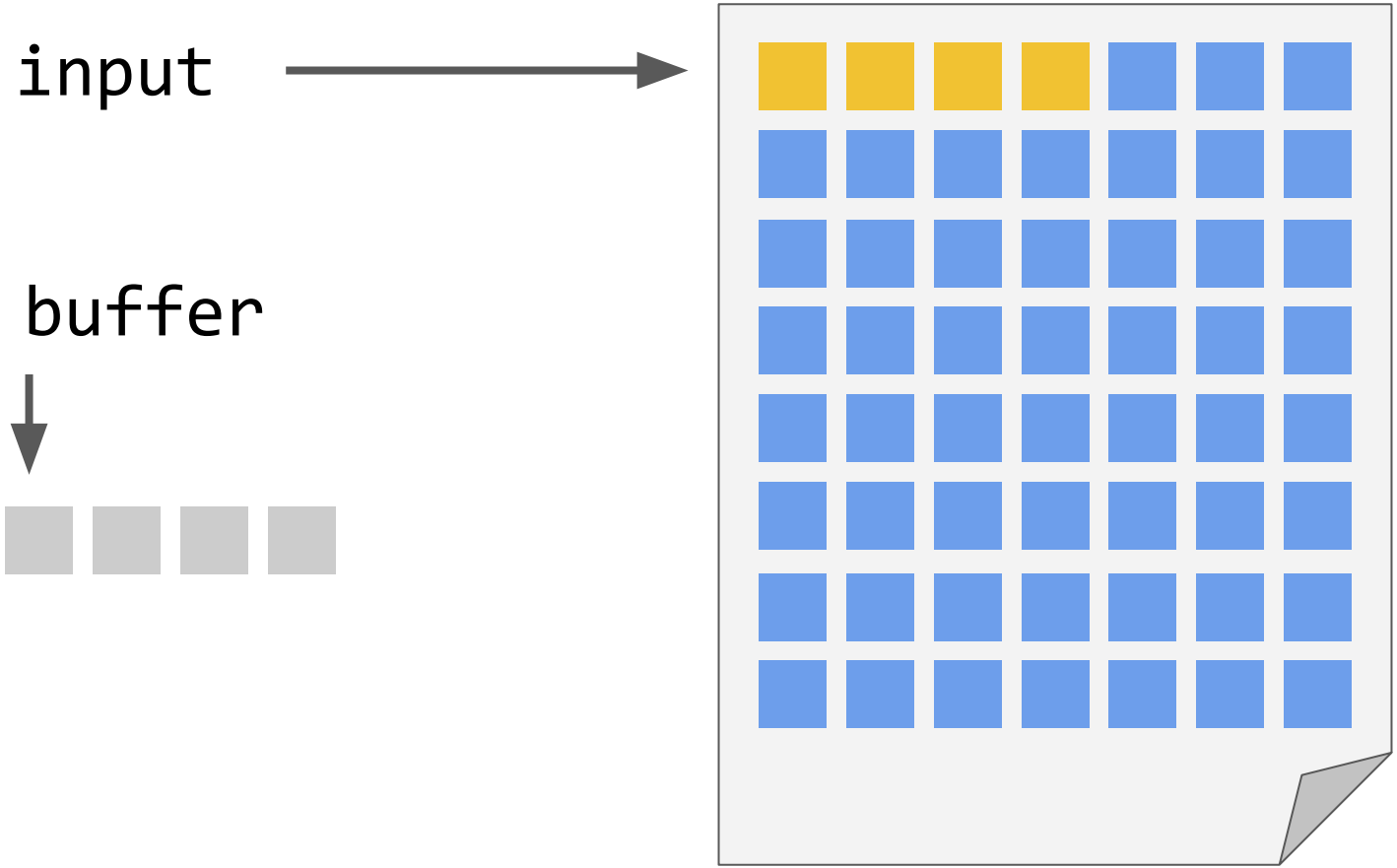
How many blocks to read

hi.txt



```
fread(buffer, 1, 4, input);
```

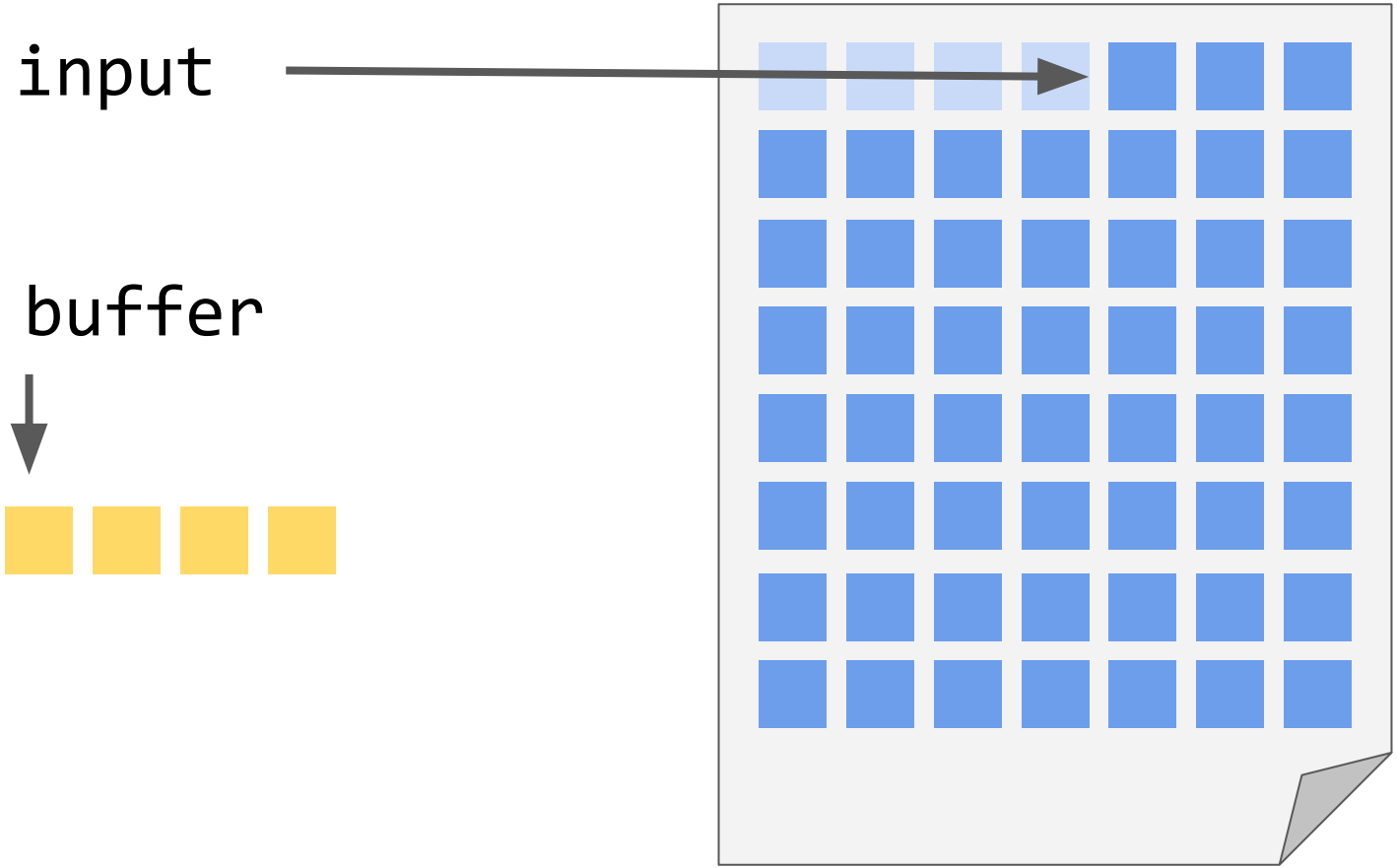
Location to store blocks


input
buffer

```
fread(buffer, 1, 4, input);
```

`fread`

- To where?
  - into **`buffer`**
- How many and what size?
  - **`4`** blocks of size **`1`**
- From where?
  - from **`input`**

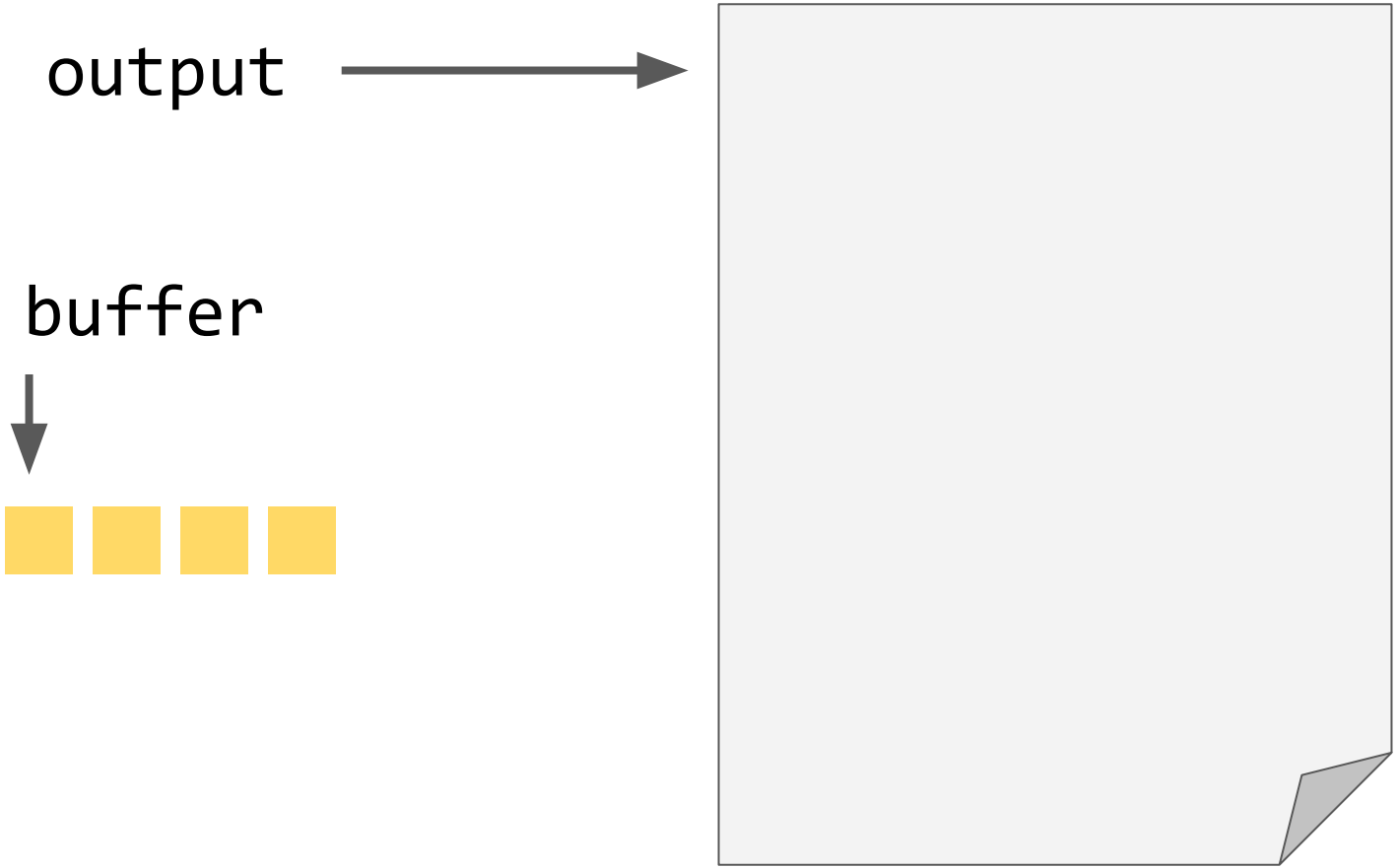

input
buffer

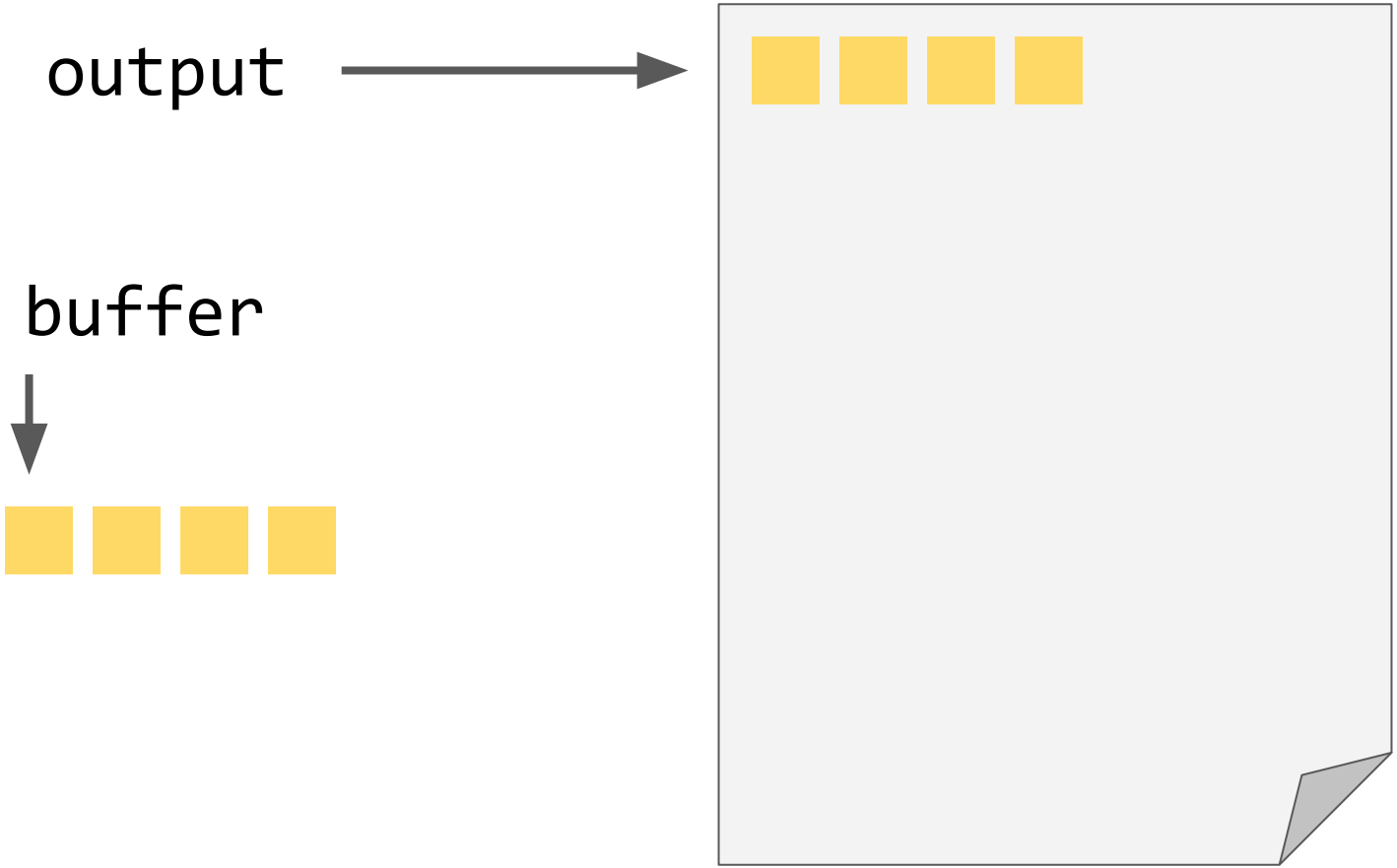
Why are we using
buffer in the first place?

```
fwrite(buffer, 1, 4, output);
```

`fwrite`

- From where?
  - from **`buffer`**
- How many and what size?
  - **`4`** blocks of size **`1`**
- To where?
  - to **`output`**


output
buffer


output
buffer

File Reading Exercise

Create a program, **pdf.c**, that checks whether a file, passed in as a command-line argument, is a PDF. All PDFs will begin with a four byte sequence:

```
0x25 0x50 0x44 0x46
```

For example: `./pdf test.pdf` should print "yes", while `./pdf test.jpg` should print "no".

This is CS50

Week 4