

Input, Output, and the American Way

- The Levels of IO
- System IO
- Standard IO
- A few extras



Two Levels of Input and Output

Be careful! Standard IO is implemented in terms of System IO.
If you mix the two types, your computer will get angry and join a cult.

System IO

- open
- close
- read
- write

Standard IO

- fopen
- fclose
- fread
- fwrite
- fprintf
- fscanff

Using System IO

```
int open(const char *path, int flags);
```

Returns the file integer on success, or -1 on failure.

The location of the file.

The following flags can be OR-ed:

Access Type Flags (1 Required):

O_RDONLY
O_WRONLY
O_RDWR

Useful Extras:

O_CREAT
O_APPEND
O_TRUNC

Using System IO

```
int close(int file);
```

-1 on
Failure

A purple rectangular box containing the text "-1 on Failure". A yellow arrow points from the top-right corner of this box to the parameter 'file' in the function signature 'int close(int file);'.

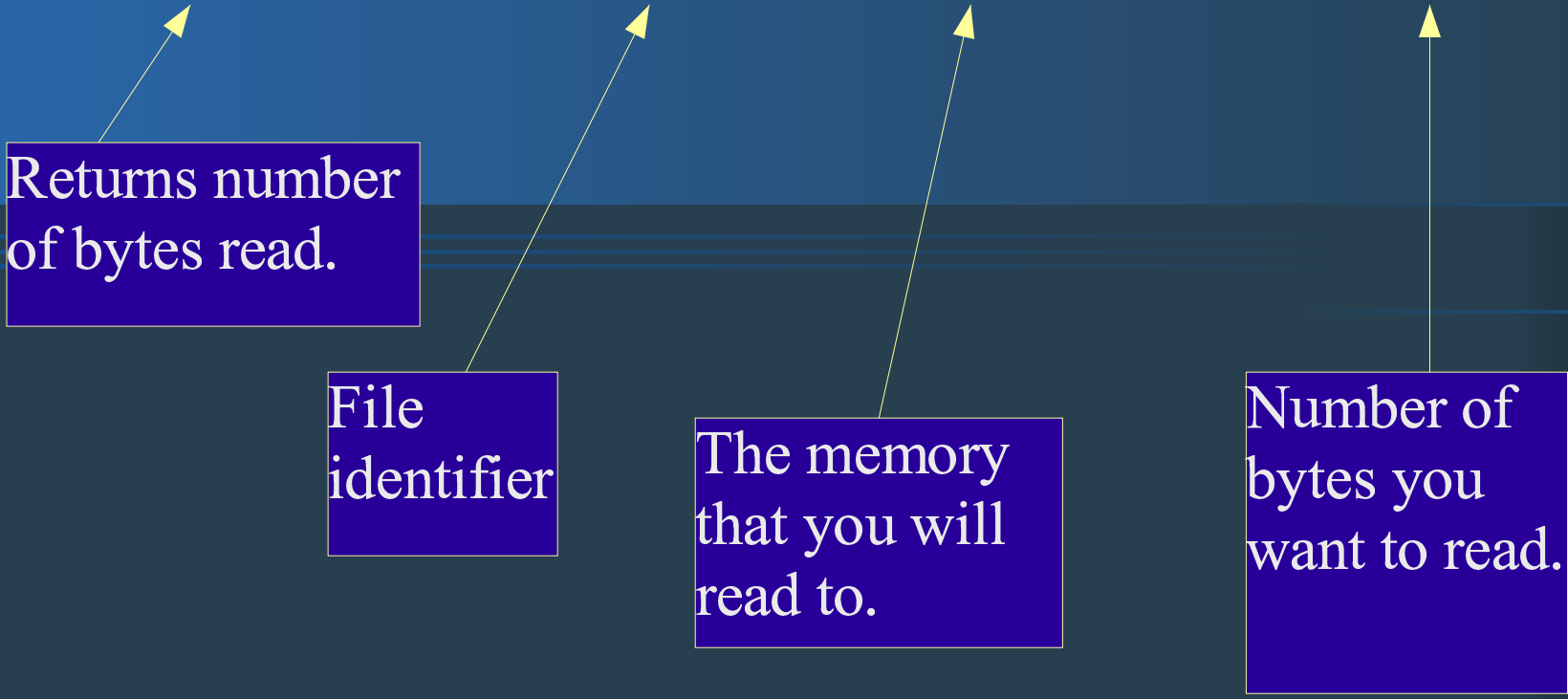
The file
descriptor.

A purple rectangular box containing the text "The file descriptor.". A yellow arrow points from the top-left corner of this box to the parameter 'file' in the function signature 'int close(int file);'.

Using System IO

```
size_t read(int file, void *buf, size_t count);
```

Returns number
of bytes read.

A diagram consisting of four yellow arrows pointing upwards from explanatory boxes to the function signature. The first arrow points from the 'Returns number of bytes read.' box to the 'size_t' return type. The second arrow points from the 'File identifier' box to the 'int file' parameter. The third arrow points from the 'The memory that you will read to.' box to the 'void *buf' parameter. The fourth arrow points from the 'Number of bytes you want to read.' box to the 'size_t count' parameter.

File
identifier

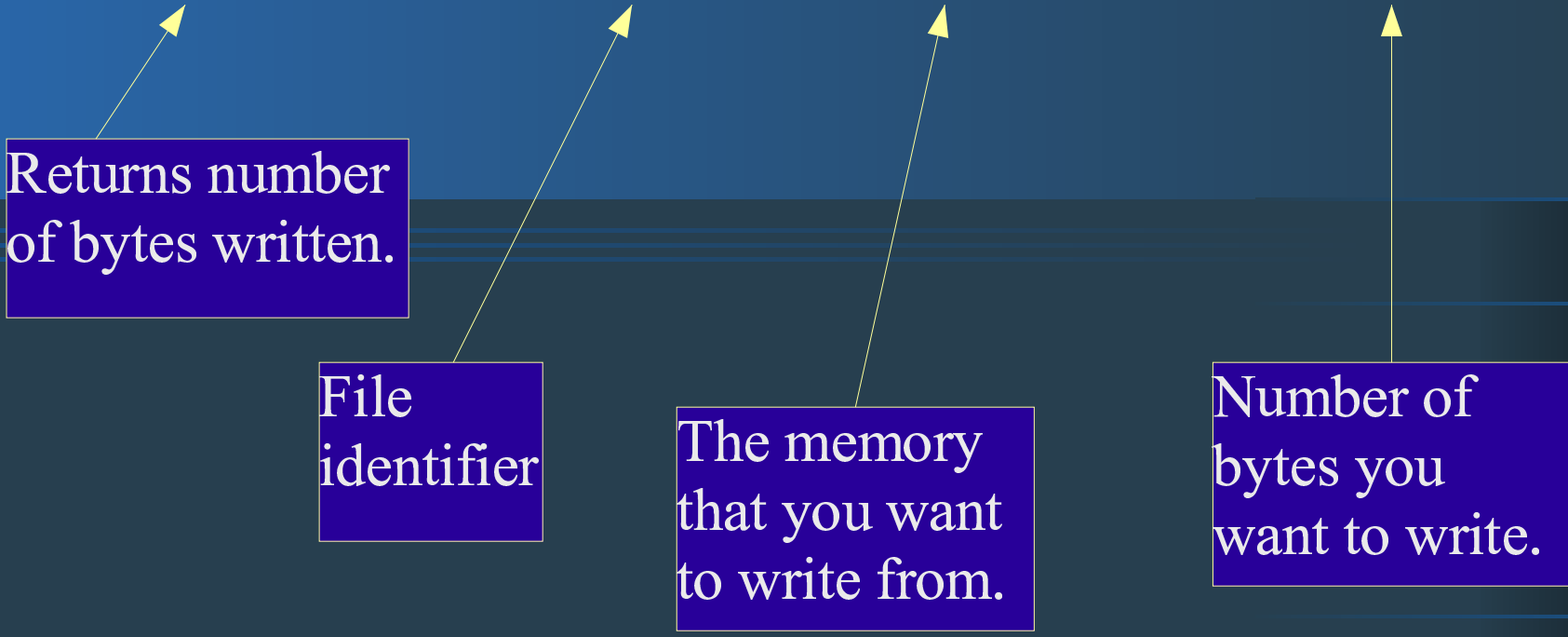
The memory
that you will
read to.

Number of
bytes you
want to read.

Using System IO

```
size_t write(int file, void *buf, size_t count);
```

Returns number
of bytes written.

A diagram consisting of four yellow arrows pointing upwards from explanatory text boxes to the corresponding parts of the function signature. The first arrow points from the return type 'size_t' to the text 'Returns number of bytes written.'. The second arrow points from the parameter 'int file' to the text 'File identifier'. The third arrow points from the parameter 'void *buf' to the text 'The memory that you want to write from.'. The fourth arrow points from the parameter 'size_t count' to the text 'Number of bytes you want to write.'.

File
identifier

The memory
that you want
to write from.

Number of
bytes you
want to write.

Using Standard IO

```
FILE *fopen(const char *path, const char *mode);
```

Returns a file pointer on success, or NULL on failure

The location of the file.

Available modes:

"r" : Read only

"w" : Create file for write only. Overwrites previous file.

"a" : Append to file

"r+" : Open for read-write

"w+" : Create new file for read-write

"a+" : Read from beginning of file, write to end.

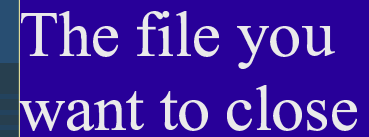
Using Standard IO

```
int fclose(FILE *file);
```

-1 on
Failure

A yellow arrow points from the top-right corner of the callout box to the space between the opening curly brace and the asterisk in the function signature.


The file you
want to close

A yellow arrow points from the top-left corner of the callout box to the asterisk in the function signature.


Using Standard IO

```
int fprintf(FILE *file, const char *format, ...);
```



The number
of characters
written



The file you
want to
write to




The printf() parameters
that you've already
come to know and love!




Using Standard IO

```
int fscanf(FILE *file, const char *format, ...);
```



The number
of characters
read



The file you
want to read
from



The scanf() parameters
that you've already
come to know and love!




Deleting Files

```
int unlink(const char *path);
```

-1 on
Failure



The location of
the file you
want to delete.



Obtaining File Information

```
int stat(const char *path, struct stat *buf);
```