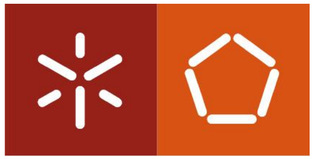
Operating Systems

(Sistemas Operativos)

Guide 1: Files

University of Minho

2024 - 2025

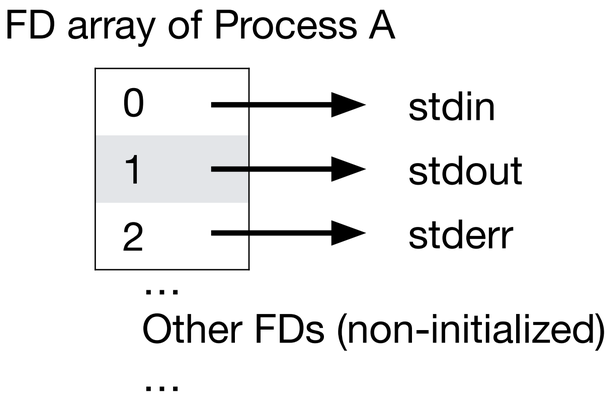


File Interface

The File Descriptor (FD)

- **File Descriptor (FD)**
  - integer that represents files
  - used for reading/writing files, pipes, or I/O devices

- **Array of file descriptors** (or file descriptor table)
  - Maintained by each process
  - Starts with three **open standard FDs**
    - **0 - Standard input:** read from keyboard
    - **1 - Standard output:** write to screen
    - **2 - Standard error**: write error messages



File Interface

Reading and Writing

#include <unistd.h>

- *ssize_t* ***read****(int fd, void *buf, size_t nbyte)*
  - **fd:** the file descriptor
  - **buf:** buffer from where the content is read
  - **nbyte:** max number of bytes to read[1]
  - Returns: **number of bytes read** (or -1 on errors)

For more information: *$ man 2 read*

#include <unistd.h>

- *ssize_t* ***write****(int fd, const void *buf, size_t nbyte)*
  - **fd:** the file descriptor
  - **buf:** buffer with content to be written
  - **nbyte:** number of bytes to write from the buffer[1]
  - Returns: **number of bytes written** (or -1 on errors)

For more information: *$ man 2 write*

[1] **Buffer overrun:** number of bytes to read/write may not be larger than the memory allocated for the buffer!

File Interface

Opening files

#include <fcntl.h>

- *int* ***open****(const char *pathname, int oflag [, mode])*
  - **pathname:** absolute or relative pathname
  - **oflag** - opening mode(s)
    - *O_WRONLY, O_RDONLY, O_RDWR* - open for writing, reading, or both, respectively
    - *O_CREAT* - create file if it does not exist
    - *O_TRUNC* - truncate file to zero length
    - *O_APPEND* - write data to the end of the file
  - **mode** - file permissions (required with *O_CREAT*)
    - *0600* - owner of the file can read/write
  - Returns: **file descriptor** (or -1 on errors)

For more information: *$ man 2 open*

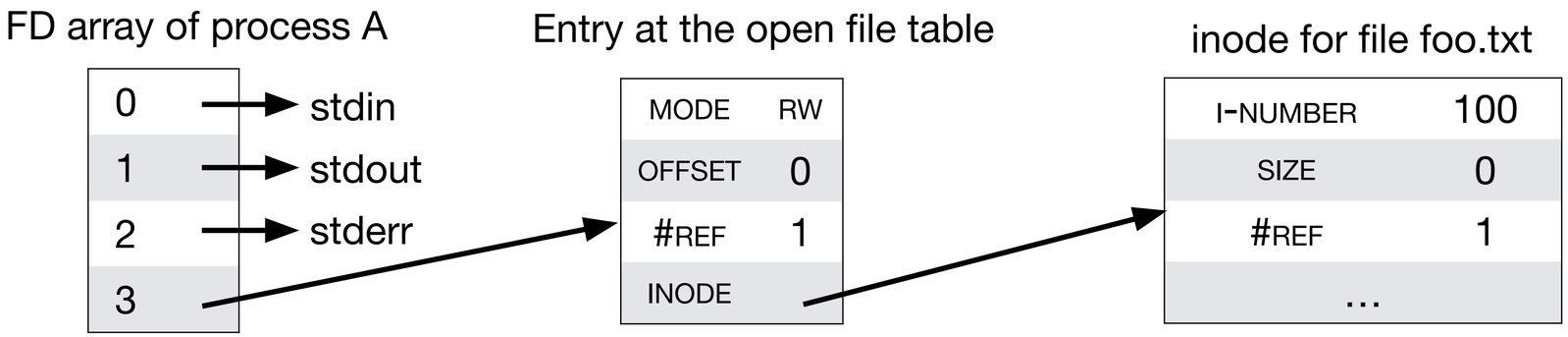
Note: Do not forget to close file descriptors when these are no longer needed (see Slide 15)

File Interface

Example: opening a file

int fd = ***open****(“foo.txt”, O_CREAT | O_TRUNC | O_RDWR, 0600)*

fd = 3

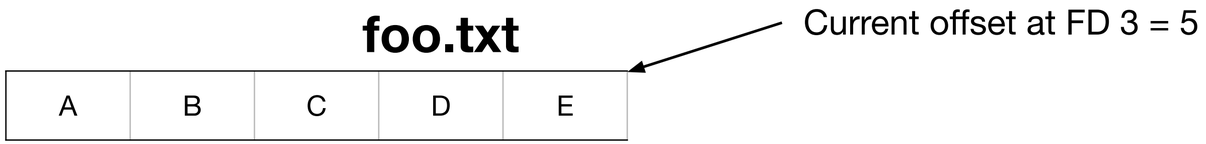


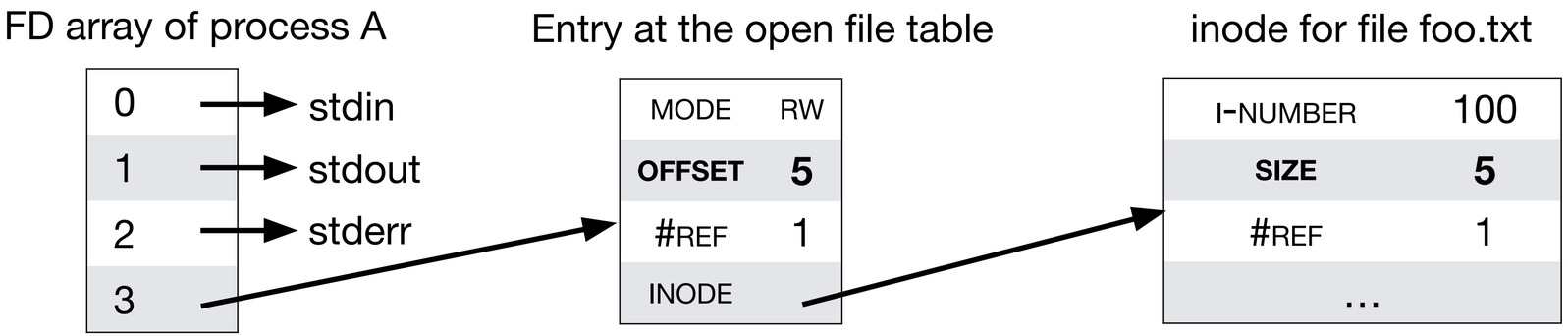
File Interface

Example: writing a file

ssize_t bytes_written = ***write****(fd, “abcde”, 5)*

bytes_written = 5



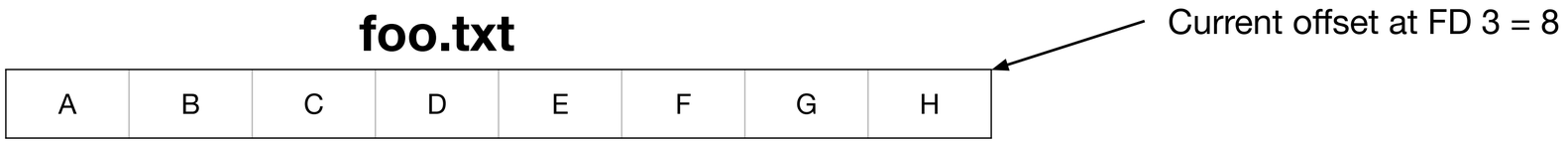


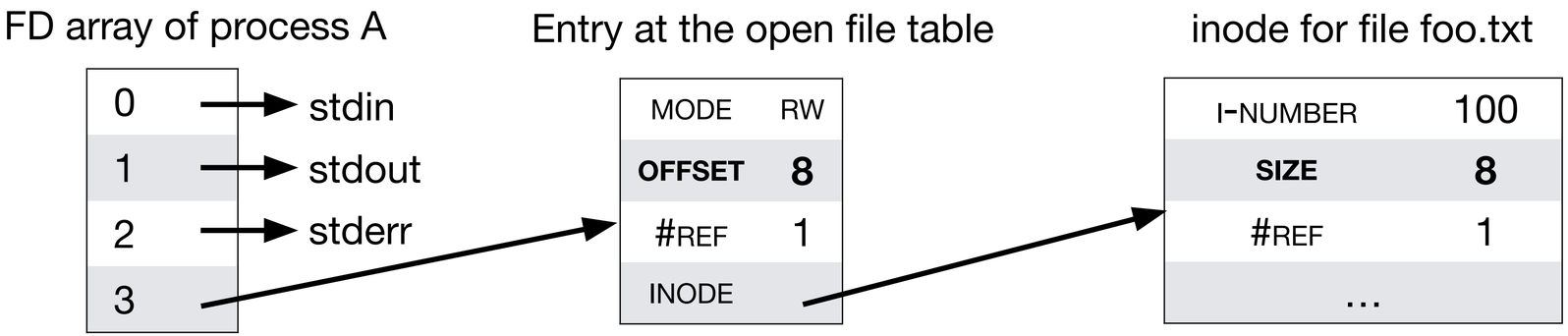
File Interface

Example: writing a file

*bytes_written = **write**(fd, "fgh", 3)*

bytes_written = 3



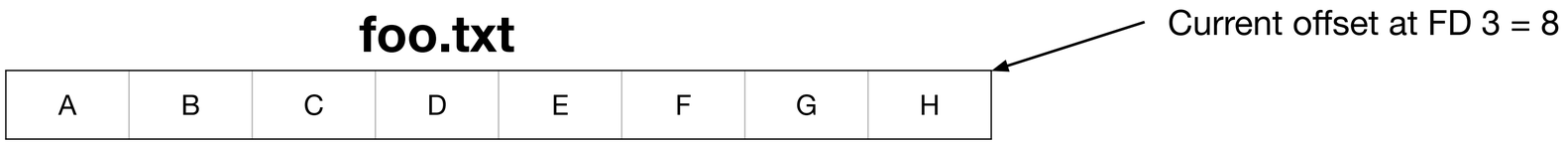


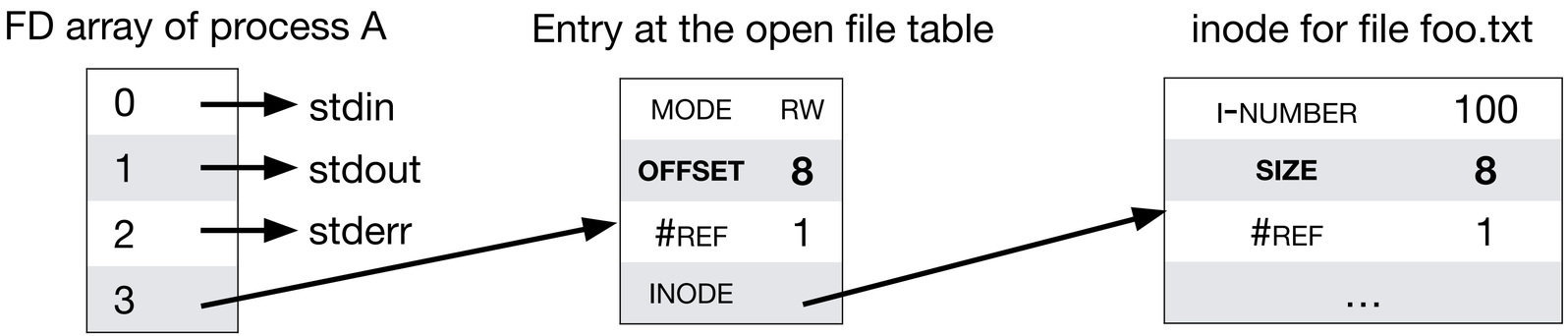
File Interface

Example: reading a file

bytes_read = ***read****(fd, buf, 8)*

bytes_read = ?



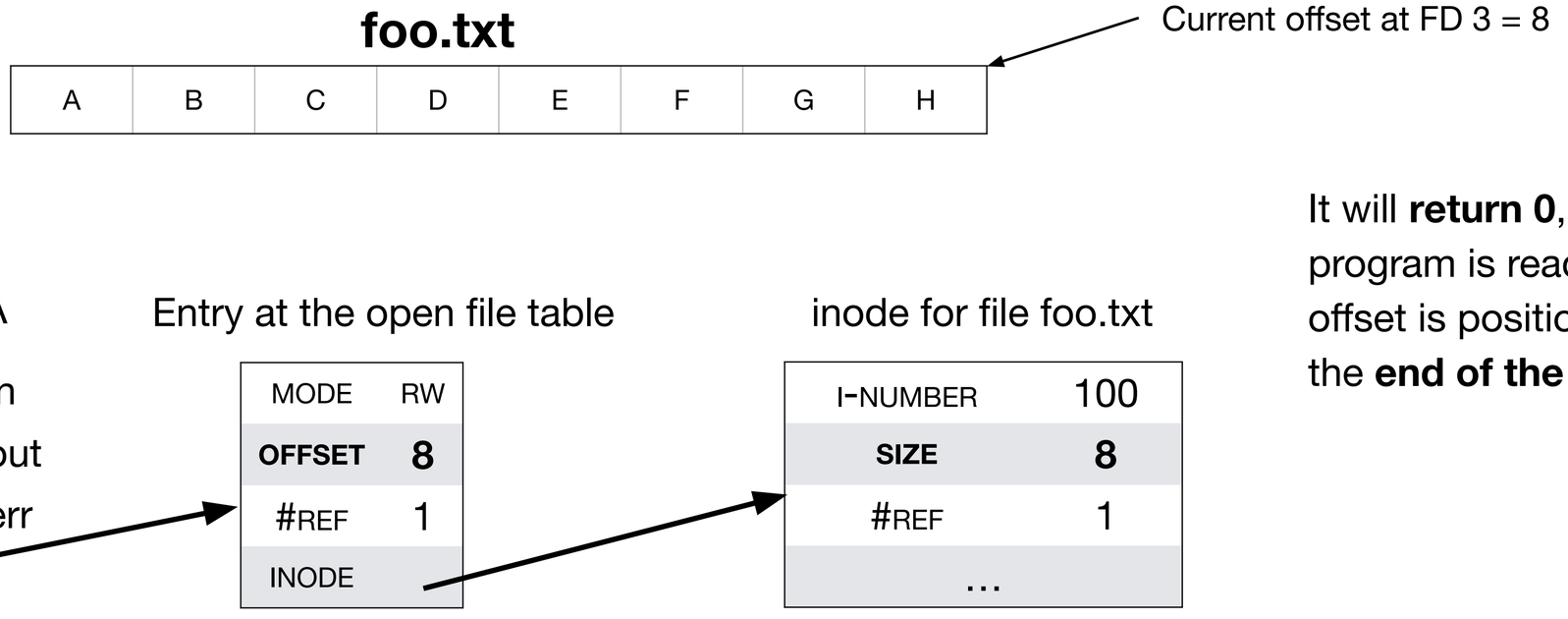


File Interface

Example: reading a file

bytes_read = ***read****(fd, buf, 8)*

bytes_read = 0



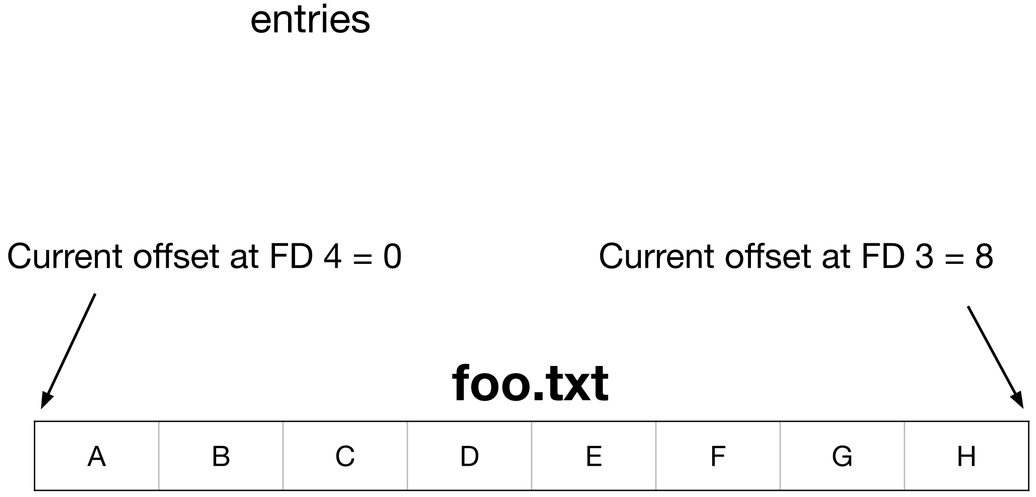
It will **return 0**, the program is reading (the offset is positioned) at the **end of the file**!

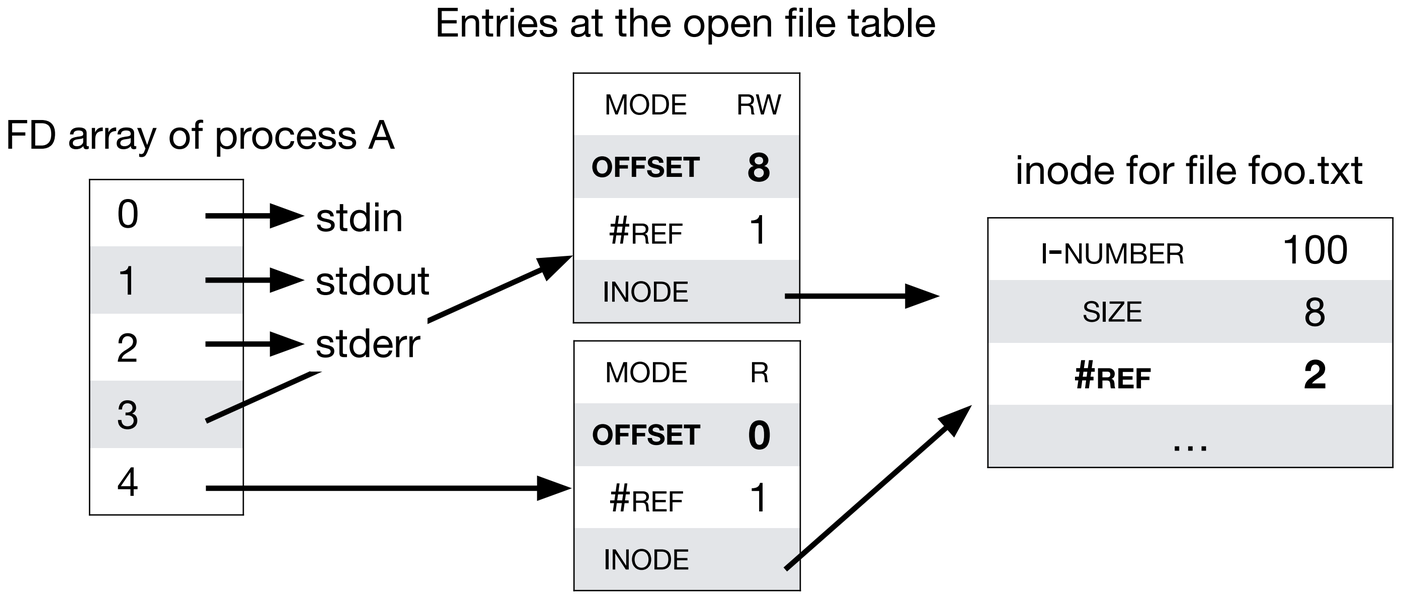
File Interface

Multiple file descriptors

- **Option 1:** Using another descriptor

  int fd1 = open("foo.txt", O_RDONLY)
  - new entry at the **open file table,** with offset = 0
  - *read(fd1, …)*
  - Note: the **inode is shared** across the two file table entries





File Interface

Random file access

- **Option 2:** Do a random access to the descriptor's offset 0

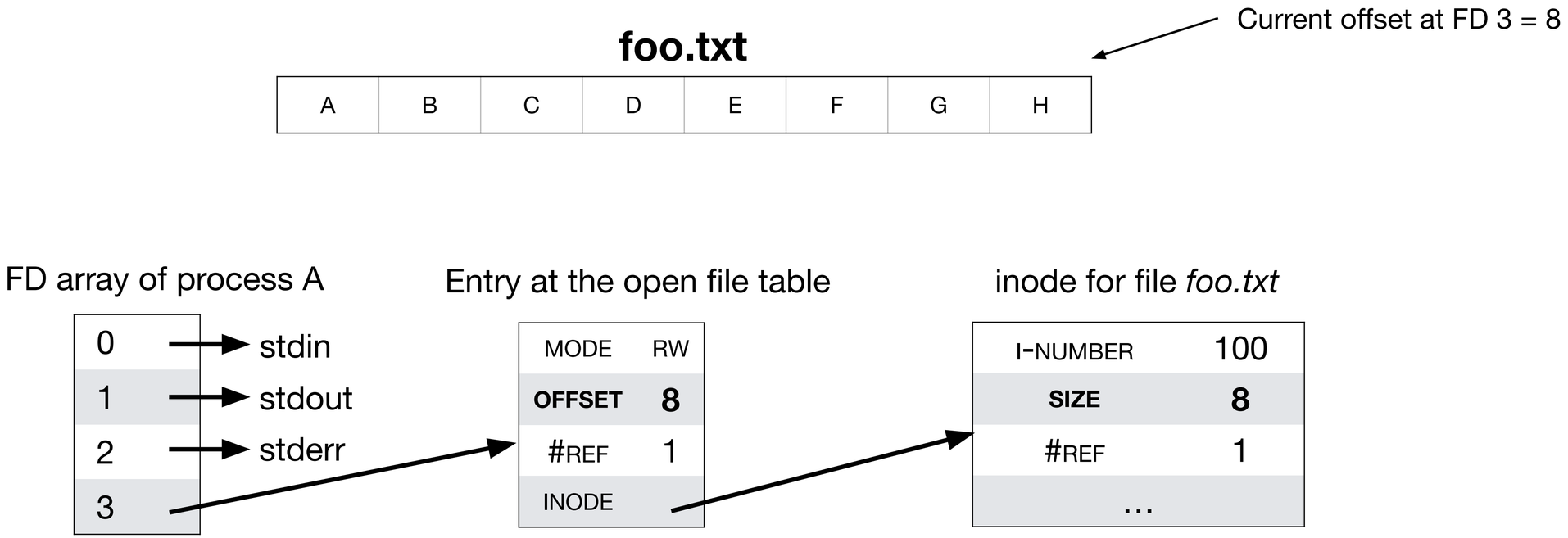
#include <unistd.h>

- *off_t* ***lseek****(int fd, off_t offset, int whence)*
  - **fd:** the file descriptor
  - **offset:** the number of bytes to move forward or backward (it can be negative)
  - **whence:** from where to move:
    - SEEK_SET - from the beginning of the file
    - SEEK_END - from the end of the file
    - SEEK_CUR - from the current offset
  - Returns: the **resulting offset at the file** or -1 on **error**

For more information: *$ man 2 lseek*

File Interface

Example: seeking and reading a file

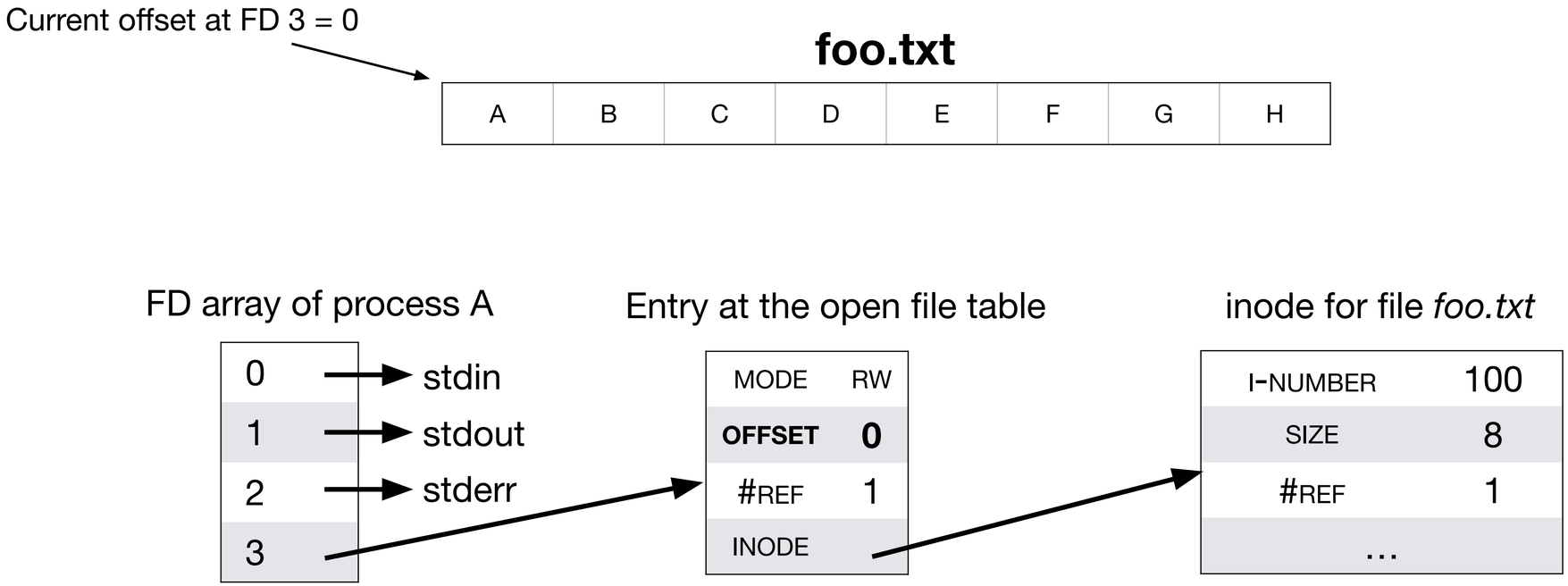


File Interface

Example: seeking and reading a file

off_t res_off = ***lseek****(fd, SEEK_SET, 0)*

res_off = 0

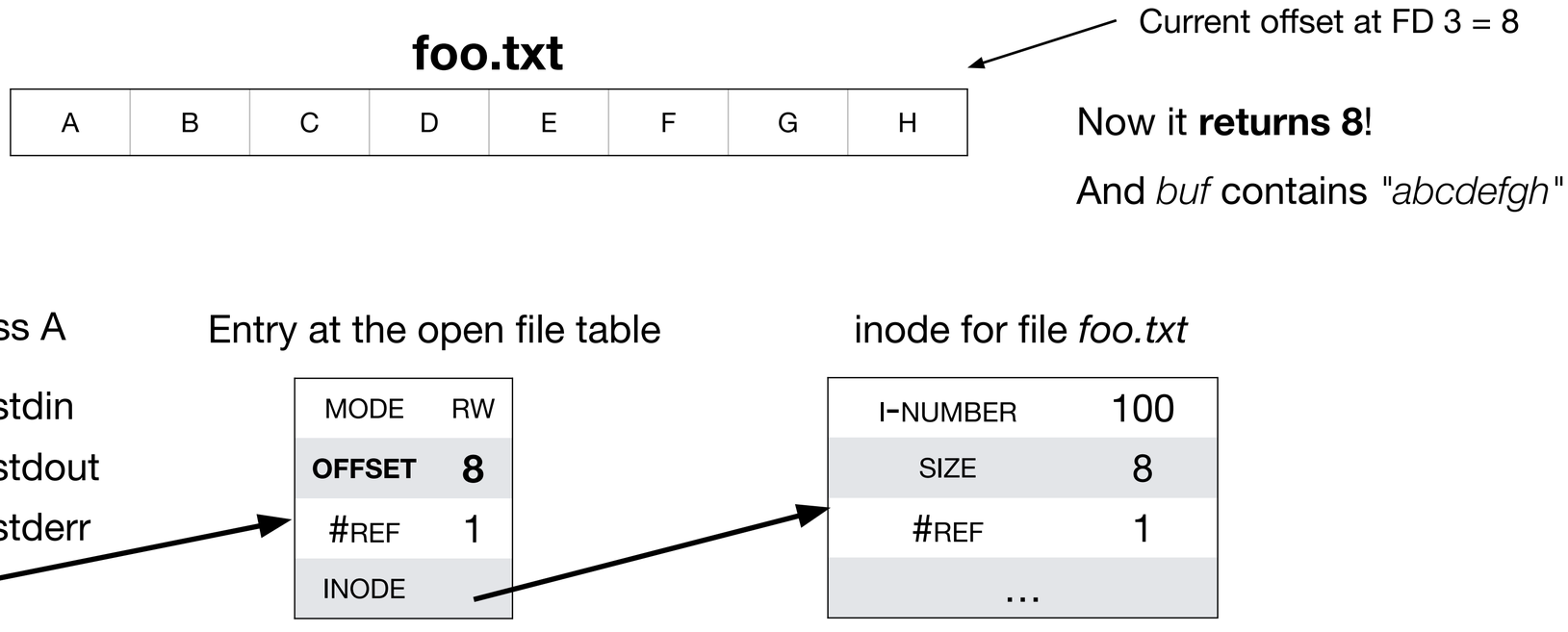


File Interface

Example: seeking and reading a file

*bytes_read = **read**(fd, buf, 8)*

bytes_read = 8



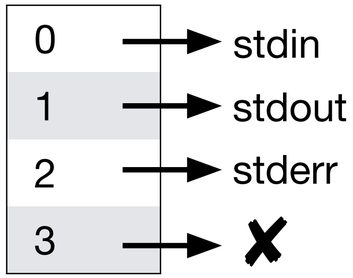
File Interface

Example: closing and deleting files

#include <unistd.h>

- *int* ***close****(int fd)*
  - **fd:** the file descriptor
  - Returns: *0* on success or *-1* on error

FD array of process A



inode for file *foo.txt*

I-NUMBER	100
SIZE	8
#REF	0
…	

inode is not deleted!

For more information: *$ man 2 close*

#include <unistd.h>

- *int* ***unlink****(const char* pathname)*
  - **pathname:** absolute or relative pathname
  - Returns: *0* on success or *-1* on error

deletes the inode when its *#ref* reaches 0

- Remember that multiple unrelated processes may have the same file opened

For more information: *$ man 2 unlink*