Systems I/O

Input/Output

Drawing: Threads and Synchronization

- Take three minutes to draw “threads and synchronization of threads”

- Some reminders
  - Shared data (code, heap, data, file meta data)
  - Thread-local data (stack, registers)
  - Threads vs processes (creation cost and shared data)
  - Invalid data state and race conditions (instructions and preemption)
  - Locks (spinning), semaphores (sleeping), condition variables (sleeping)
  - Atomic operations (cannot be interrupted; e.g., the `xchg` instruction)
  - Critical sections (mutating/writing to shared data)
  - Reading is safe; writing is unsafe

System I/O as a Uniform Interface

Operating systems use a uniform system I/O interface for all I/O devices

Commands to read and write to a file descriptor are the same no matter what type of "file"

Types of files include

- File (input/output)
- Keyboard (input)
- Screen (output)
- Pipe (input/output)
- Network (input/output
- Etc.

https://en.cppreference.com/w/c/io

C API (Not Systems Calls)

```c
// Defined in header <stdio.h>

#define stdin  /* implementation-defined */
#define stdout /* implementation-defined */
#define stderr /* implementation-defined */

FILE *fopen( const char *restrict filename, const char *restrict mode );
int fclose( FILE *stream );

size_t fread( void *restrict buffer, size_t size, size_t count, FILE *restrict stream );
size_t fwrite( const void *restrict buffer, size_t size, size_t count, FILE *restrict stream );

int fscanf( FILE *restrict stream, const char *restrict format, ... );
int fprintf( FILE *restrict stream, const char *restrict format, ... );

int printf( const char *restrict format, ... );
```

https://en.cppreference.com/w/c/io

C API (Not Systems Calls)

```
// Defined in header <stdio.h>

#define stdin  /* implementation-defined */
#define stdout /* implementation-defined */
#define stderr /* implementation-defined */

FILE *fopen( const char *restrict filename, const char *restrict mode );
int fclose( FILE *stream );

size_t fread( void *restrict buffer, size_t size, size_t count, FILE *restrict stream );
size_t fwrite( const void *restrict buffer, size_t size, size_t count, FILE *restrict stream );

int fscanf( FILE *restrict stream, const char *restrict format, ... );
int fprintf( FILE *restrict stream, const char *restrict format, ... );

int printf( const char *restrict format, ... );
```

```
fprintf(ppm_ptr, "P6 %d %d 255 ", image_width, image_height);
```
Assignment 1

```
fprintf(stderr, "job '%s' complete\n", j->command);
```
Assignment 8

Our First I/O System: File Systems

Long-term information storage goals

- Should be able to store large amounts of information
- Information must survive processes, power failures, etc.
- Processes must be able to find information
- Needs to support concurrent accesses by multiple processes

Solution: the file system abstraction

- Interface that provides operations involving: Files and Directories
  - Directories are just a special kind of file

The File System Abstraction

Interface that provides operations on data stored long-term on disk

A file is a named sequence of stored bytes

- Name is defined on creation
- Processes use name to subsequently access that file

A file comprises two parts:

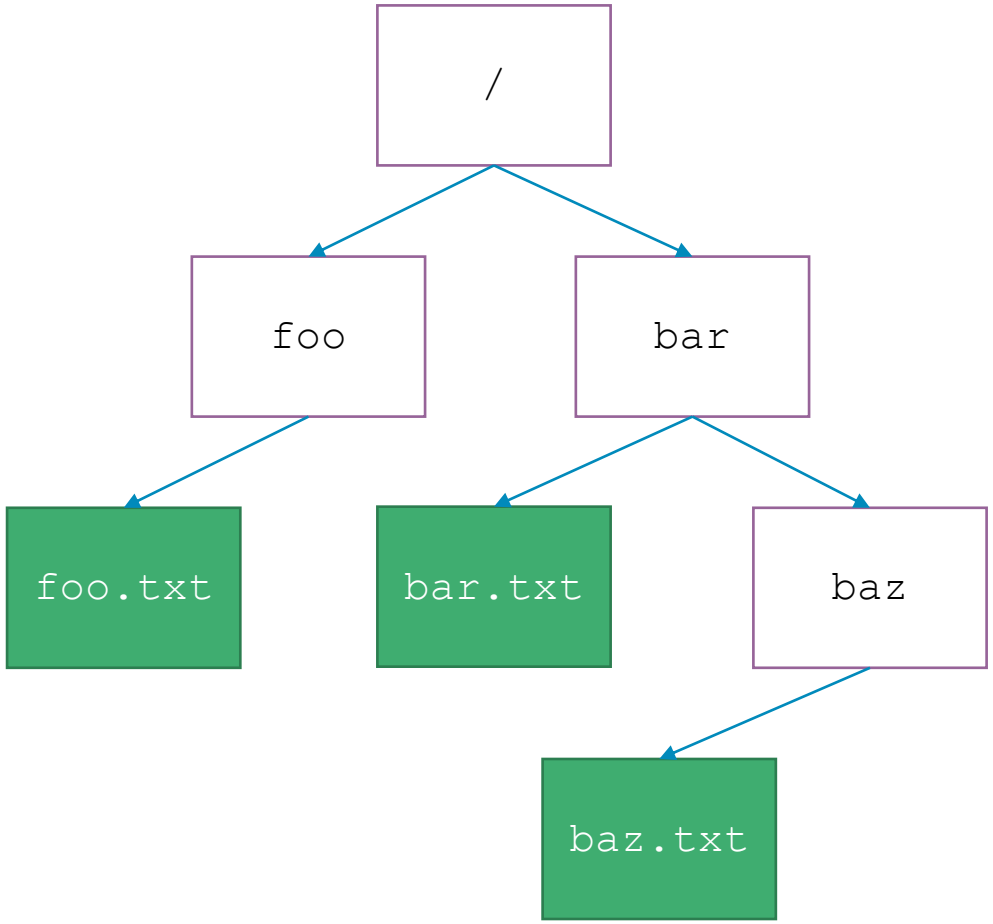
- Data: information a user or application puts in a file (an array of untyped bytes)
- Metadata: information added and managed by the OS (e.g., size, owner, security info, modification time)

Two types of files

- Normal files: data is an arbitrary sequence of bytes
- Directories: a special type of file that provides mappings from human-readable names to low-level names (i.e., File numbers)

Path Names

- Each path from root is a name for a leaf
  - `/foo/foo.txt`
  - `/bar/baz/baz.txt`
- Each UNIX directory contains 2 special entries
  - " . " = this directory
  - " . . " = parent directory
- Absolute paths: path of file from the root directory
- Relative paths: path from current working directory (stored as part of a process's data)



Practice with Path Names

Assume I created a file named `example1.txt` in `/data/`

How would you

- Specify an absolute path to the file `example1.txt`
- Specify a relative path to the file `example1.txt` from **your home directory**

How would you create a file named `example2.txt` in your home directory? And then

- Specify an absolute path to the file `example2.txt`
- Specify a relative path to the file `example2.txt` from **your home directory**

- Hint: you can always get back to your home directory with `cd ~`
- Hint: the name of your home directory is your username

Practice with Path Names

Assume I created a file named `example1.txt` in `/data/`

How would you

- Specify an absolute path to the file `example1.txt`
- Specify a relative path to the file `example1.txt` from **your home directory**

How would you create a file named `example2.txt` in your home directory? And then

- Specify an absolute path to the file `example2.txt`
- Specify a relative path to the file `example2.txt` from **your home directory**

- Hint: you can always get back to your home directory with `cd ~`
- Hint: the name of your home directory is your username

```
/data/example1.txt
```

```
../../data/example1.txt
```

```
touch ~/example2.txt
```

```
/home/ajcd2020/example1.txt
```

```
./example1.txt
```

Basic File System Operations

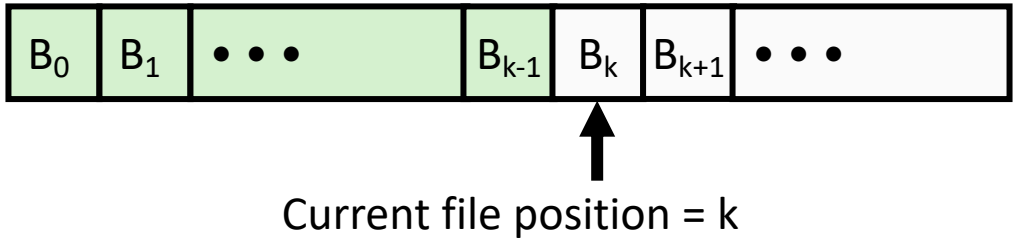
- Create a file
- Delete a file
- Write to a file
- Read from a file
- Seek to somewhere in a file

How should the OS implement this functionality?

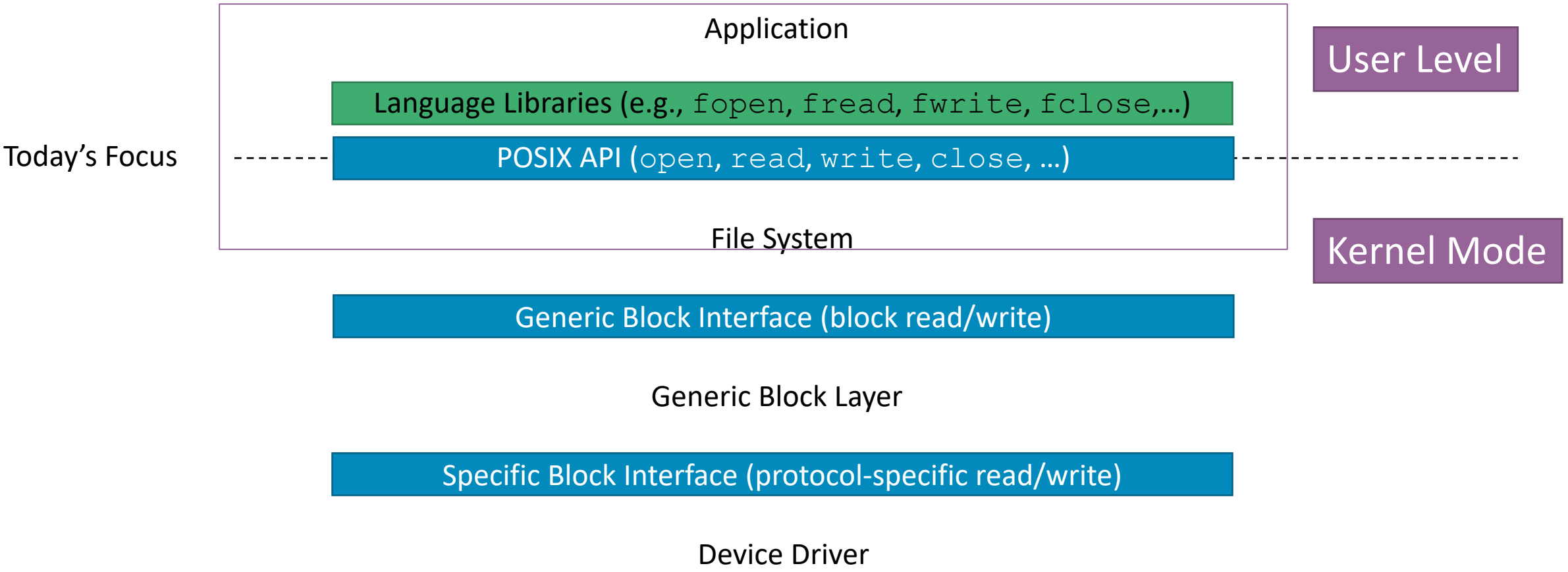
Unix I/O Interface (System Calls)

The mapping of files to devices allows kernel to export simple interface:

- Opening a file
  - **`open()`** and **`close()`**
- Reading and writing a file
  - **`read()`** and **`write()`**
- Changing the current **file position** (seek)
  - indicates next offset into file to read or write
  - **`lseek()`**



The File System Stack



[ajcd2020@itbdcv-lnx04p 11SystemIO]$ strace cat readwrite.c > newfile.txt

```
execve("/usr/bin/cat", ["cat", "readwrite.c"], 0x7ffc92b9bfa8 /* 37 vars */) = 0
brk(NULL)                               = 0x55a92c569000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffc060259e0) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fbf2b459000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=36571, ...}) = 0
mmap(NULL, 36571, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fbf2b450000
close(3)                                = 0
openat(AT_FDCWD, "/lib64/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\25\3\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2089152, ...}) = 0
lseek(3, 808, SEEK_SET)                 = 808
read(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 32) = 32
mmap(NULL, 3950400, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fbf2ae69000
mprotect(0x7fbf2b025000, 2093056, PROT_NONE) = 0
mmap(0x7fbf2b224000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bb000) = 0x7fbf2b224000
mmap(0x7fbf2b22a000, 14144, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fbf2b22a000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fbf2b44e000
arch_prctl(ARCH_SET_FS, 0x7fbf2b45a680) = 0
mprotect(0x7fbf2b224000, 16384, PROT_READ) = 0
mprotect(0x55a92b636000, 4096, PROT_READ) = 0
mprotect(0x7fbf2b45b000, 4096, PROT_READ) = 0
munmap(0x7fbf2b450000, 36571)           = 0
brk(NULL)                               = 0x55a92c569000
brk(0x55a92c58a000)                     = 0x55a92c58a000
brk(NULL)                               = 0x55a92c58a000
openat(AT_FDCWD, "/usr/lib/locale/locale-archive", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/share/locale/locale.alias", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=2997, ...}) = 0
read(3, "# Locale name alias data base.\n#"..., 4096) = 2997
read(3, "", 4096)                       = 0
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/locale/en_US.UTF-8/LC_IDENTIFICATION", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/lib/locale/en_US.utf8/LC_IDENTIFICATION", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=368, ...}) = 0
mmap(NULL, 368, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fbf2b458000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib64/gconv/gconv-modules.cache", O_RDONLY) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=26998, ...}) = 0
mmap(NULL, 26998, PROT_READ, MAP_SHARED, 3, 0) = 0x7fbf2b451000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/locale/en_US.UTF-8/LC_MEASUREMENT", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/lib/locale/en_US.utf8/LC_MEASUREMENT", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=23, ...}) = 0
mmap(NULL, 23, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fbf2b450000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/locale/en_US.UTF-8/LC_TELEPHONE", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/lib/locale/en_US.utf8/LC_TELEPHONE", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=59, ...}) = 0
mmap(NULL, 59, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fbf2b44d000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/locale/en_US.UTF-8/LC_ADDRESS", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/lib/locale/en_US.utf8/LC_ADDRESS", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=167, ...}) = 0
mmap(NULL, 167, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fbf2b44c000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/locale/en_US.UTF-8/LC_NAME", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/lib/locale/en_US.utf8/LC_NAME", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=77, ...}) = 0
mmap(NULL, 77, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fbf2b44b000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/locale/en_US.UTF-8/LC_PAPER", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/lib/locale/en_US.utf8/LC_PAPER", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=34, ...}) = 0
mmap(NULL, 34, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fbf2b44a000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/locale/en_US.UTF-8/LC_MESSAGES", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/lib/locale/en_US.utf8/LC_MESSAGES", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFDIR|0755, st_size=29, ...}) = 0
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/locale/en_US.utf8/LC_MESSAGES/SYS_LC_MESSAGES", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=57, ...}) = 0
mmap(NULL, 57, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fbf2b449000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/locale/en_US.UTF-8/LC_MONETARY", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/lib/locale/en_US.utf8/LC_MONETARY", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=286, ...}) = 0
mmap(NULL, 286, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fbf2b448000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/locale/en_US.UTF-8/LC_COLLATE", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/lib/locale/en_US.utf8/LC_COLLATE", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=2586930, ...}) = 0
mmap(NULL, 2586930, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fbf2abf1000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/locale/en_US.UTF-8/LC_TIME", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/lib/locale/en_US.utf8/LC_TIME", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=3316, ...}) = 0
mmap(NULL, 3316, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fbf2b447000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/locale/en_US.UTF-8/LC_NUMERIC", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/lib/locale/en_US.utf8/LC_NUMERIC", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=54, ...}) = 0
mmap(NULL, 54, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fbf2b446000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/locale/en_US.UTF-8/LC_CTYPE", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/lib/locale/en_US.utf8/LC_CTYPE", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=337024, ...}) = 0
mmap(NULL, 337024, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fbf2b3f3000
close(3)                                = 0
fstat(1, {st_mode=S_IFREG|0644, st_size=0, ...}) = 0
openat(AT_FDCWD, "readwrite.c", O_RDONLY) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=242, ...}) = 0
fadvise64(3, 0, 0, POSIX_FADV_SEQUENTIAL) = 0
mmap(NULL, 139264, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fbf2b3d1000
read(3, "#include <fcntl.h>\n#include <stdi"..., 131072) = 242
write(1, "#include <fcntl.h>\n#include <std"..., 242) = 242
read(3, "", 131072)                     = 0
munmap(0x7fbf2b3d1000, 139264)          = 0
close(3)                                = 0
close(1)                                = 0
close(2)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

```
[ajcd2020@itbdcv-lnx04p 09-solution]$ ./virtmem
```

	Heap Sort						Quick Sort					
	Random			LRU			Random			LRU		
N	Miss	Access	Rate	Miss	Access	Rate	Miss	Access	Rate	Miss	Access	Rate
2	0	6	0.00000	0	6	0.00000	0	11	0.00000	0	11	0.00000
4	0	44	0.00000	0	44	0.00000	0	42	0.00000	0	42	0.00000
8	0	154	0.00000	0	154	0.00000	0	139	0.00000	0	139	0.00000
16	0	430	0.00000	0	430	0.00000	0	378	0.00000	0	378	0.00000
32	0	1164	0.00000	0	1164	0.00000	0	1079	0.00000	0	1079	0.00000
64	0	3070	0.00000	0	3070	0.00000	0	2795	0.00000	0	2795	0.00000
128	0	7324	0.00000	0	7324	0.00000	0	7188	0.00000	0	7188	0.00000
256	0	17222	0.00000	0	17222	0.00000	0	18522	0.00000	0	18522	0.00000
512	0	39530	0.00000	0	39530	0.00000	0	38994	0.00000	0	38994	0.00000
1024	0	89858	0.00000	0	89858	0.00000	0	93004	0.00000	0	93004	0.00000
2048	0	200220	0.00000	0	200220	0.00000	0	202388	0.00000	0	202388	0.00000
4096	0	441126	0.00000	0	441126	0.00000	0	431784	0.00000	0	431784	0.00000
8192	0	965550	0.00000	0	965550	0.00000	0	1000443	0.00000	0	1000443	0.00000

```
Segmentation fault (core dumped)
```

```
[ajcd2020@itbdcv-lnx04p 09-solution]$ strace ./virtmem
…
openat(AT_FDCWD, "files/00000000000000000000000000000008.pg", O_WRONLY|O_CREAT|O_TRUNC, 0666) = -1 ENOENT (No such file or directory)
--- SIGSEGV {si_signo=SIGSEGV, si_code=SEGV_MAPERR, si_addr=NULL} ---
+++ killed by SIGSEGV (core dumped) +++
Segmentation fault (core dumped)
```

(Side note: use the makefiles!)

Opening Files

- Opening a file informs the kernel that you are getting ready to access that file

```
int fd;    /* file descriptor */

if ((fd = open("/etc/hosts", O_RDONLY)) < 0) {
   perror("open");
   exit(1);
}
```

- Returns a small identifying integer file descriptor
  - `fd == -1` indicates that an error occurred

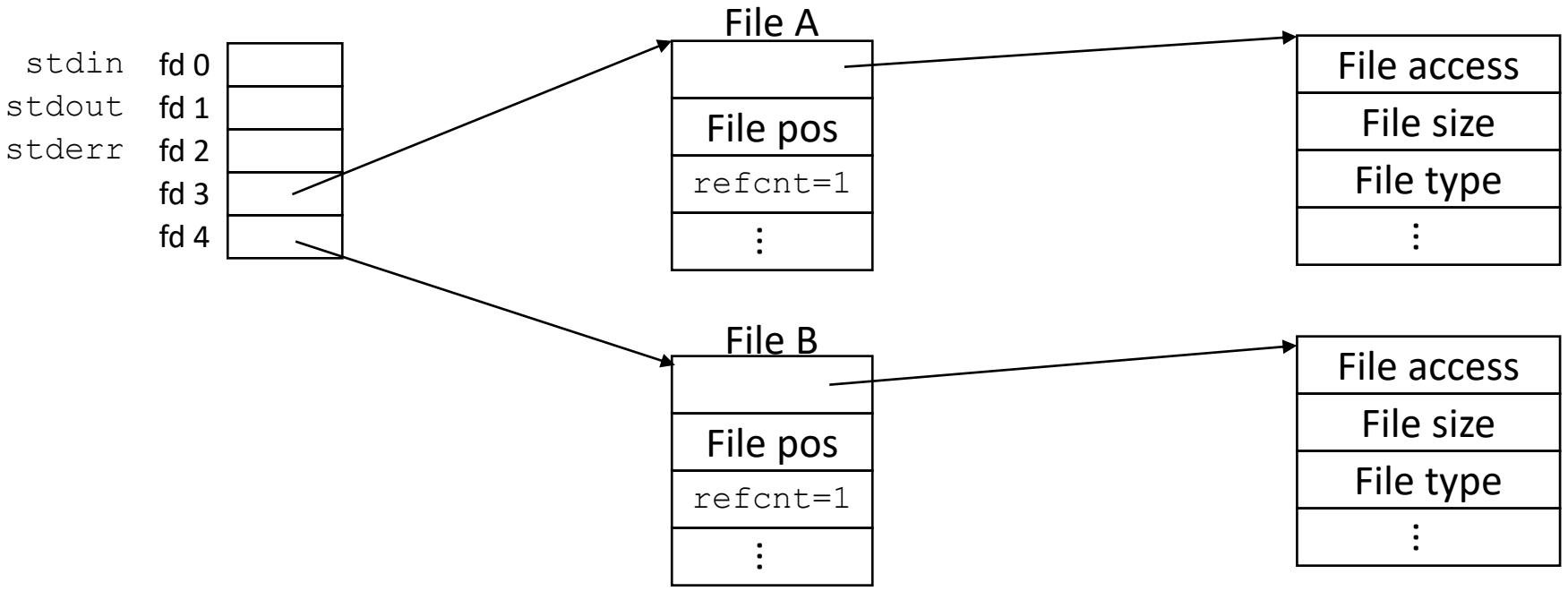
- Each process created by a Linux shell begins life with three open files associated with a specific terminal:
  - 0: standard input (`stdin`)
  - 1: standard output (`stdout`)
  - 2: standard error (`stderr`)

Kernel Data Structures

Descriptor table
(table created on `fork`,
one table per process)

Open file table
(entry created on `open`,
shared by all processes)

v-node table
(one per file,
shared by all processes)



Closing Files

- Closing a file informs the kernel that you are finished accessing that file

```
int fd;      /* file descriptor */
int retval; /* return value */

if ((retval = close(fd)) < 0) {
   perror("close");
   exit(1);
}
```

- Closing an already closed file is a recipe for disaster in threaded programs
- Moral: Always check return codes, even for seemingly benign functions such as `close()`

Reading Files

- Reading a file copies bytes from the current file position to memory, and then updates file position

```
char buf[512];
int fd;       /* file descriptor */
int nbytes;   /* number of bytes read */

/* Open file fd ...  */
/* Then read up to 512 bytes from file fd */
if ((nbytes = read(fd, buf, sizeof(buf))) < 0) {
   perror("read");
   exit(1);
}
```

- Returns number of bytes read from file `fd` into `buf`
  - Return type `ssize_t` is signed integer
  - `nbytes < 0` indicates that an error occurred
  - Short counts (`nbytes < sizeof(buf)`) are possible and are not errors!

Writing Files

- Writing a file copies bytes from memory to the current file position, and then updates current file position

```
char buf[512];
int fd;        /* file descriptor */
int nbytes;    /* number of bytes read */

/* Open the file fd ... */
/* Then write up to 512 bytes from buf to file fd */
if ((nbytes = write(fd, buf, sizeof(buf)) < 0) {
   perror("write");
   exit(1);
}
```

- Returns number of bytes written from `buf` to file `fd`
  - `nbytes < 0` indicates that an error occurred
  - As with reads, short counts are possible and are not errors!

On Short Counts

- Short counts can occur in these situations:
  - Encountering (end-of-file) EOF on reads
  - Reading text lines from a terminal

- Short counts never occur in these situations:
  - Reading from disk files (except for EOF)
  - Writing to disk files

- Best practice is to always allow for short counts.

Buffered Reads/Writes

- Stream data is stored in a kernel buffer and returned to the application on request

- Enables same system call interface to handle both streaming reads (e.g., Keyboard) and block reads (e.g., Disk)

Practice with Reading and Writing

Assume the file `foobar.txt` consists of the six characters: `foobar`

What gets printed when the following program is run?

What system calls are made while running the program?

```
int main(int argc, char ** argv){

    int fd1 = open("foobar.txt", O_RDONLY);
    int fd2 = open("foobar.txt", O_RDONLY);

    char c;
    read(fd1, &c, 1);
    read(fd2, &c, 1);

    printf("c = %c\n", c);

    return 0;
}
```

Practice with Reading and Writing

Assume the file `foobar.txt` consists of the six characters: `foobar`

What gets printed when the following program is run?

What system calls are made while running the program?

```
int main(int argc, char ** argv){

    int fd1 = open("foobar.txt", O_RDONLY);
    int fd2 = open("foobar.txt", O_RDONLY);

    char c;
    read(fd1, &c, 1);
    read(fd2, &c, 1);

    printf("c = %c\n", c);

    return 0;
}
```

f

f

```
open
open

read
read

write

close?
exit
```

```
[ajcd2020@itbdcv-lnx04p 11SystemIO]$ ll
total 8
-rw-r--r--. 1 ajcd2020 domain users   7 Nov 13 19:50 foobar.txt
-rw-r--r--. 1 ajcd2020 domain users 242 Nov 13 19:51 readwrite.c
[ajcd2020@itbdcv-lnx04p 11SystemIO]$ bat foobar.txt
```

```
       File: foobar.txt
   1   foobar
```

```
[ajcd2020@itbdcv-lnx04p 11SystemIO]$ bat readwrite.c
```

```
       File: readwrite.c
   1   #include <fcntl.h>
   2   #include <stdio.h>
   3   #include <unistd.h>
   4
   5   int main() {
   6
   7       int fd1 = open("foobar.txt", O_RDONLY);
   8       int fd2 = open("foobar.txt", O_RDONLY);
   9
  10       char c;
  11       read(fd1, &c, 1);
  12       read(fd2, &c, 1);
  13
  14       printf("c = %c\n", c);
  15       return 0;
  16   }
  17
```

```
[ajcd2020@itbdcv-lnx04p 11SystemIO]$ gcc -o readwrite readwrite.c
[ajcd2020@itbdcv-lnx04p 11SystemIO]$ ./readwrite
c = f
[ajcd2020@itbdcv-lnx04p 11SystemIO]$
```

```
[ajcd2020@itbdcv-lnx04p 11SystemIO]$ strace ./readwrite
execve("./readwrite", ["./readwrite"], 0x7fffde0a5f30 /* 37 vars */) = 0
brk(NULL)                               = 0x24e7000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe205c1960) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ff749fe1000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=36571, ...}) = 0
mmap(NULL, 36571, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ff749fd8000
close(3)                                = 0
openat(AT_FDCWD, "/lib64/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\256\3\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2089152, ...}) = 0
lseek(3, 808, SEEK_SET)                 = 808
read(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32) = 32
mmap(NULL, 3950400, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7ff7499f1000
mprotect(0x7ff749bad000, 2093056, PROT_NONE) = 0
mmap(0x7ff749dac000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bb000) = 0x7ff749dac000
mmap(0x7ff749db2000, 14144, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7ff749db2000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ff749fd6000
arch_prctl(ARCH_SET_FS, 0x7ff749fe2600) = 0
mprotect(0x7ff749dac000, 16384, PROT_READ) = 0
mprotect(0x600000, 4096, PROT_READ)     = 0
mprotect(0x7ff749fe3000, 4096, PROT_READ) = 0
munmap(0x7ff749fd8000, 36571)           = 0
openat(AT_FDCWD, "foobar.txt", O_RDONLY) = 3
openat(AT_FDCWD, "foobar.txt", O_RDONLY) = 4
read(3, "f", 1)                         = 1
read(4, "f", 1)                         = 1
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x5), ...}) = 0
brk(NULL)                               = 0x24e7000
brk(0x2508000)                          = 0x2508000
brk(NULL)                               = 0x2508000
write(1, "c = f\n", 6c = f
)                  = 6
exit_group(0)                           = ?
+++ exited with 0 +++
```

```
[ajcd2020@itbdcv-lnx04p 11SystemIO]$ strace ./readwrite
execve("./readwrite", ["./readwrite"], 0x7fffde0a5f30 /* 37 vars */) = 0
brk(NULL)                               = 0x24e7000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe205c1960) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ff749fe1000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=36571, ...}) = 0
mmap(NULL, 36571, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ff749fd8000
close(3)                                = 0
openat(AT_FDCWD, "/lib64/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\256\3\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2089152, ...}) = 0
lseek(3, 808, SEEK_SET)                 = 808
read(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32) = 32
mmap(NULL, 3950400, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7ff7499f1000
mprotect(0x7ff749bad000, 2093056, PROT_NONE) = 0
mmap(0x7ff749dac000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bb000) = 0x7ff749dac000
mmap(0x7ff749db2000, 14144, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7ff749db2000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ff749fd6000
arch_prctl(ARCH_SET_FS, 0x7ff749fe2600) = 0
mprotect(0x7ff749dac000, 16384, PROT_READ) = 0
mprotect(0x600000, 4096, PROT_READ)     = 0
mprotect(0x7ff749fe3000, 4096, PROT_READ) = 0
munmap(0x7ff749fd8000, 36571)           = 0
openat(AT_FDCWD, "foobar.txt", O_RDONLY) = 3
openat(AT_FDCWD, "foobar.txt", O_RDONLY) = 4
read(3, "f", 1)                         = 1
read(4, "f", 1)                         = 1
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x5), ...}) = 0
brk(NULL)                               = 0x24e7000
brk(0x2508000)                          = 0x2508000
brk(NULL)                               = 0x2508000
write(1, "c = f\n", 6c = f
)                  = 6
exit_group(0)                           = ?
+++ exited with 0 +++
```

```
[ajcd2020@itbdcv-lnx04p 11SystemIO]$ strace ./readwrite
execve("./readwrite", ["./readwrite"], 0x7fffde0a5f30 /* 37 vars */) = 0
brk(NULL)                               = 0x24e7000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe205c1960) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ff749fe1000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=36571, ...}) = 0
mmap(NULL, 36571, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ff749fd8000
close(3)                                = 0
openat(AT_FDCWD, "/lib64/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\256\3\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2089152, ...}) = 0
lseek(3, 808, SEEK_SET)                 = 808
read(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32) = 32
mmap(NULL, 3950400, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7ff7499f1000
mprotect(0x7ff749bad000, 2093056, PROT_NONE) = 0
mmap(0x7ff749dac000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bb000) = 0x7ff749dac000
mmap(0x7ff749db2000, 14144, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7ff749db2000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ff749fd6000
arch_prctl(ARCH_SET_FS, 0x7ff749fe2600) = 0
mprotect(0x7ff749dac000, 16384, PROT_READ) = 0
mprotect(0x600000, 4096, PROT_READ)     = 0
mprotect(0x7ff749fe3000, 4096, PROT_READ) = 0
munmap(0x7ff749fd8000, 36571)           = 0
openat(AT_FDCWD, "foobar.txt", O_RDONLY) = 3
openat(AT_FDCWD, "foobar.txt", O_RDONLY) = 4
read(3, "f", 1)                         = 1
read(4, "f", 1)                         = 1
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x5), ...}) = 0
brk(NULL)                               = 0x24e7000
brk(0x2508000)                          = 0x2508000
brk(NULL)                               = 0x2508000
write(1, "c = f\n", 6c = f
)                  = 6
exit_group(0)                           = ?
+++ exited with 0 +++
```

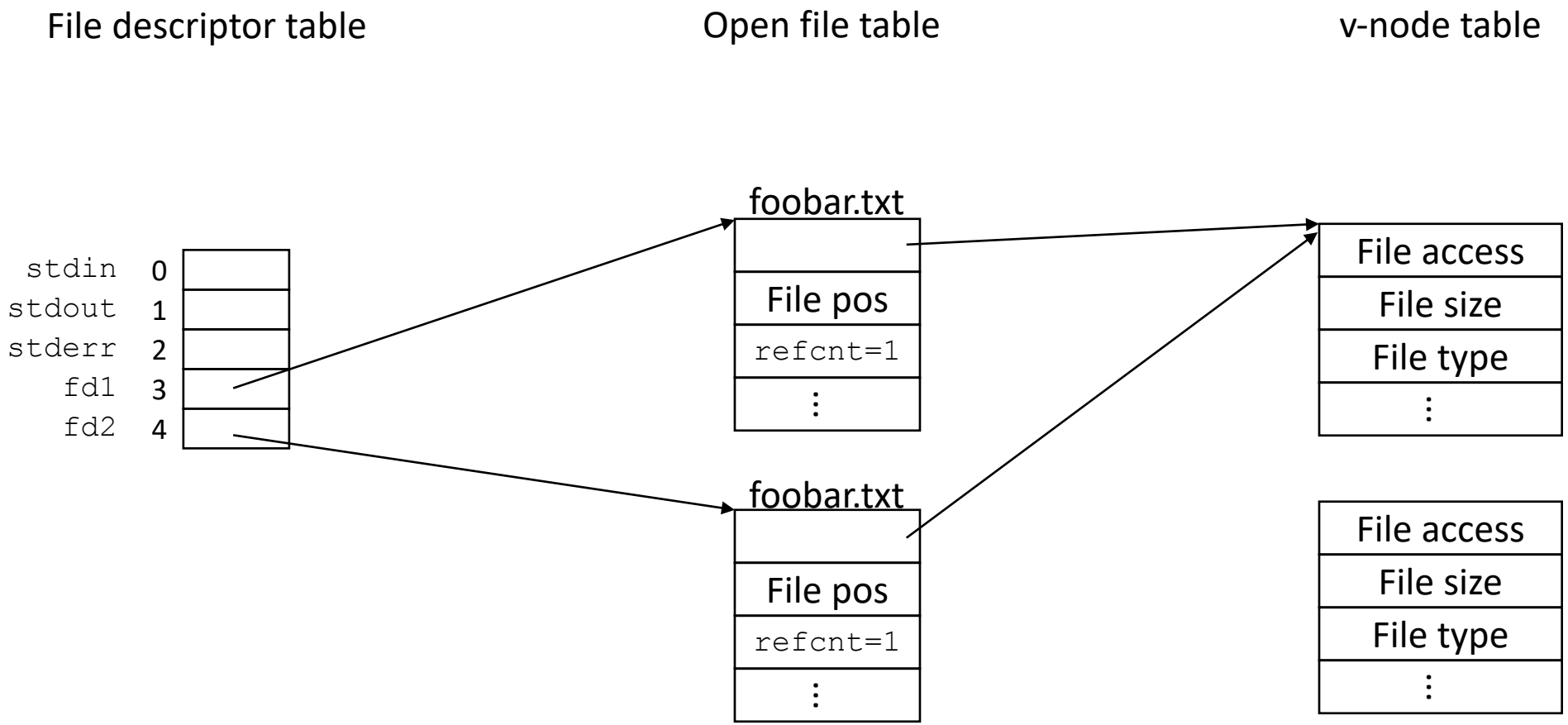
```
[ajcd2020@itbdcv-lnx04p 11SystemIO]$ strace ./readwrite
execve("./readwrite", ["./readwrite"], 0x7fffde0a5f30 /* 37 vars */) = 0
brk(NULL)                               = 0x24e7000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe205c1960) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ff749fe1000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=36571, ...}) = 0
mmap(NULL, 36571, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ff749fd8000
close(3)                                = 0
openat(AT_FDCWD, "/lib64/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\256\3\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2089152, ...}) = 0
lseek(3, 808, SEEK_SET)                 = 808
read(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32) = 32
mmap(NULL, 3950400, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7ff7499f1000
mprotect(0x7ff749bad000, 2093056, PROT_NONE) = 0
mmap(0x7ff749dac000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bb000) = 0x7ff749dac000
mmap(0x7ff749db2000, 14144, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7ff749db2000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ff749fd6000
arch_prctl(ARCH_SET_FS, 0x7ff749fe2600) = 0
mprotect(0x7ff749dac000, 16384, PROT_READ) = 0
mprotect(0x600000, 4096, PROT_READ)     = 0
mprotect(0x7ff749fe3000, 4096, PROT_READ) = 0
munmap(0x7ff749fd8000, 36571)           = 0
openat(AT_FDCWD, "foobar.txt", O_RDONLY) = 3
openat(AT_FDCWD, "foobar.txt", O_RDONLY) = 4
read(3, "f", 1)                         = 1
read(4, "f", 1)                         = 1
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x5), ...}) = 0
brk(NULL)                               = 0x24e7000
brk(0x2508000)                          = 0x2508000
brk(NULL)                               = 0x2508000
write(1, "c = f\n", 6c = f
)                  = 6
exit_group(0)                           = ?
+++ exited with 0 +++
```

```
[ajcd2020@itbdcv-lnx04p 11SystemIO]$ strace ./readwrite
execve("./readwrite", ["./readwrite"], 0x7fffde0a5f30 /* 37 vars */) = 0
brk(NULL)                               = 0x24e7000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe205c1960) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ff749fe1000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=36571, ...}) = 0
mmap(NULL, 36571, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ff749fd8000
close(3)                                = 0
openat(AT_FDCWD, "/lib64/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\256\3\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2089152, ...}) = 0
lseek(3, 808, SEEK_SET)                 = 808
read(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32) = 32
mmap(NULL, 3950400, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7ff7499f1000
mprotect(0x7ff749bad000, 2093056, PROT_NONE) = 0
mmap(0x7ff749dac000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bb000) = 0x7ff749dac000
mmap(0x7ff749db2000, 14144, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7ff749db2000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ff749fd6000
arch_prctl(ARCH_SET_FS, 0x7ff749fe2600) = 0
mprotect(0x7ff749dac000, 16384, PROT_READ) = 0
mprotect(0x600000, 4096, PROT_READ)     = 0
mprotect(0x7ff749fe3000, 4096, PROT_READ) = 0
munmap(0x7ff749fd8000, 36571)           = 0
openat(AT_FDCWD, "foobar.txt", O_RDONLY) = 3
openat(AT_FDCWD, "foobar.txt", O_RDONLY) = 4
read(3, "f", 1)                                 = 1
read(4, "f", 1)                                 = 1
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x5), ...}) = 0
brk(NULL)                               = 0x24e7000
brk(0x2508000)                          = 0x2508000
brk(NULL)                               = 0x2508000
write(1, "c = f\n", 6c = f
)                  = 6
exit_group(0)                           = ?
+++ exited with 0 +++
```

```
[ajcd2020@itbdcv-lnx04p 11SystemIO]$ strace ./readwrite
execve("./readwrite", ["./readwrite"], 0x7fffde0a5f30 /* 37 vars */) = 0
brk(NULL)                               = 0x24e7000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe205c1960) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ff749fe1000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=36571, ...}) = 0
mmap(NULL, 36571, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ff749fd8000
close(3)                                = 0
openat(AT_FDCWD, "/lib64/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\256\3\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2089152, ...}) = 0
lseek(3, 808, SEEK_SET)                 = 808
read(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32) = 32
mmap(NULL, 3950400, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7ff7499f1000
mprotect(0x7ff749bad000, 2093056, PROT_NONE) = 0
mmap(0x7ff749dac000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bb000) = 0x7ff749dac000
mmap(0x7ff749db2000, 14144, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7ff749db2000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ff749fd6000
arch_prctl(ARCH_SET_FS, 0x7ff749fe2600) = 0
mprotect(0x7ff749dac000, 16384, PROT_READ) = 0
mprotect(0x600000, 4096, PROT_READ)     = 0
mprotect(0x7ff749fe3000, 4096, PROT_READ) = 0
munmap(0x7ff749fd8000, 36571)           = 0
openat(AT_FDCWD, "foobar.txt", O_RDONLY) = 3
openat(AT_FDCWD, "foobar.txt", O_RDONLY) = 4
read(3, "f", 1)                         = 1
read(4, "f", 1)                         = 1
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x5), ...}) = 0
brk(NULL)                               = 0x24e7000
brk(0x2508000)                          = 0x2508000
brk(NULL)                               = 0x2508000
write(1, "c = f\n", 6c = f
)                       = 6
exit_group(0)                           = ?
+++ exited with 0 +++
```

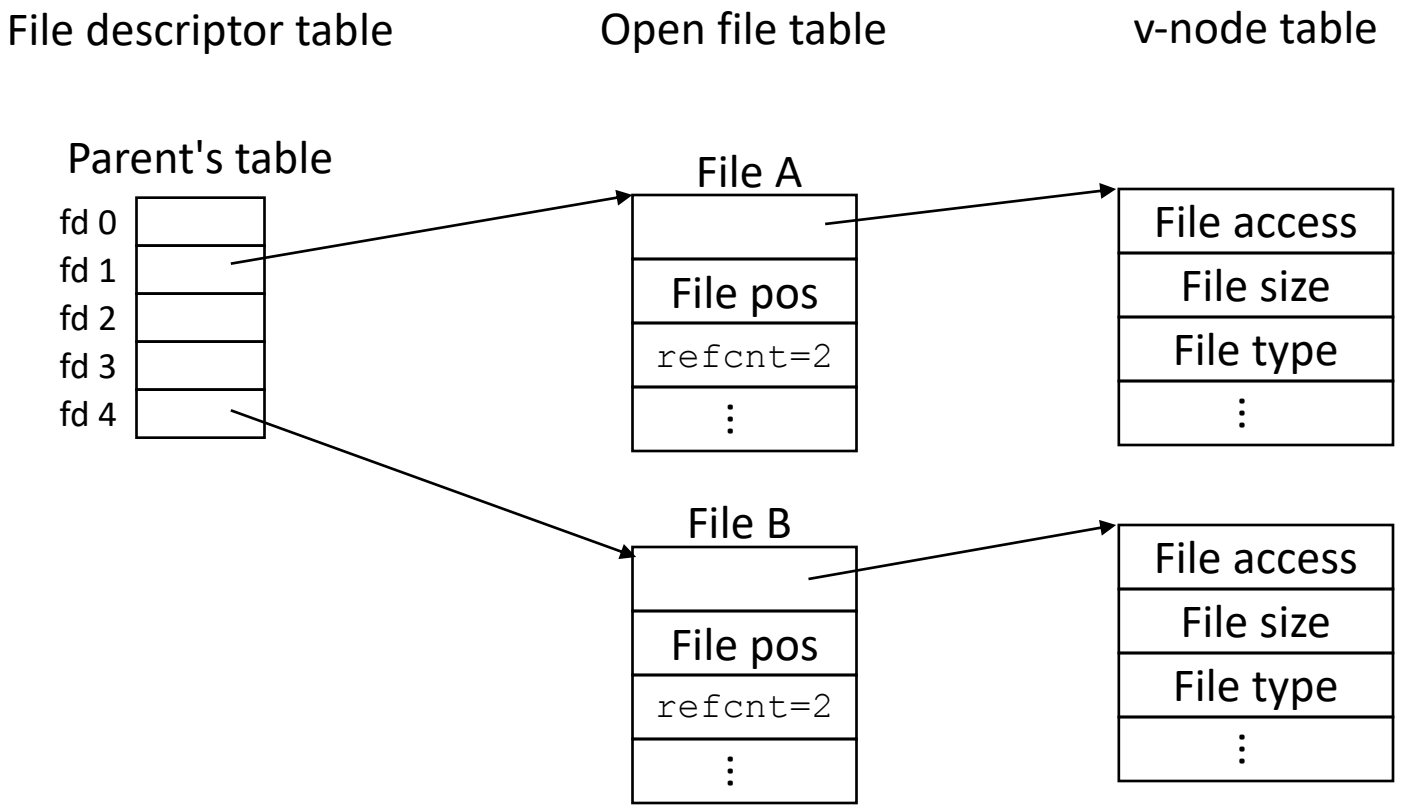
```
[ajcd2020@itbdcv-lnx04p 11SystemIO]$ strace ./readwrite
execve("./readwrite", ["./readwrite"], 0x7fffde0a5f30 /* 37 vars */) = 0
brk(NULL)                               = 0x24e7000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe205c1960) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ff749fe1000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=36571, ...}) = 0
mmap(NULL, 36571, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ff749fd8000
close(3)                                = 0
openat(AT_FDCWD, "/lib64/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\256\3\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2089152, ...}) = 0
lseek(3, 808, SEEK_SET)                 = 808
read(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32) = 32
mmap(NULL, 3950400, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7ff7499f1000
mprotect(0x7ff749bad000, 2093056, PROT_NONE) = 0
mmap(0x7ff749dac000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bb000) = 0x7ff749dac000
mmap(0x7ff749db2000, 14144, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7ff749db2000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ff749fd6000
arch_prctl(ARCH_SET_FS, 0x7ff749fe2600) = 0
mprotect(0x7ff749dac000, 16384, PROT_READ) = 0
mprotect(0x600000, 4096, PROT_READ)     = 0
mprotect(0x7ff749fe3000, 4096, PROT_READ) = 0
munmap(0x7ff749fd8000, 36571)           = 0
openat(AT_FDCWD, "foobar.txt", O_RDONLY) = 3
openat(AT_FDCWD, "foobar.txt", O_RDONLY) = 4
read(3, "f", 1)                         = 1
read(4, "f", 1)                         = 1
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x5), ...}) = 0
brk(NULL)                               = 0x24e7000
brk(0x2508000)                          = 0x2508000
brk(NULL)                               = 0x2508000
write(1, "c = f\n", 6c = f
)                 = 6
exit_group(0)                                      = ?
+++ exited with 0 +++
```

Practice with Reading and Writing



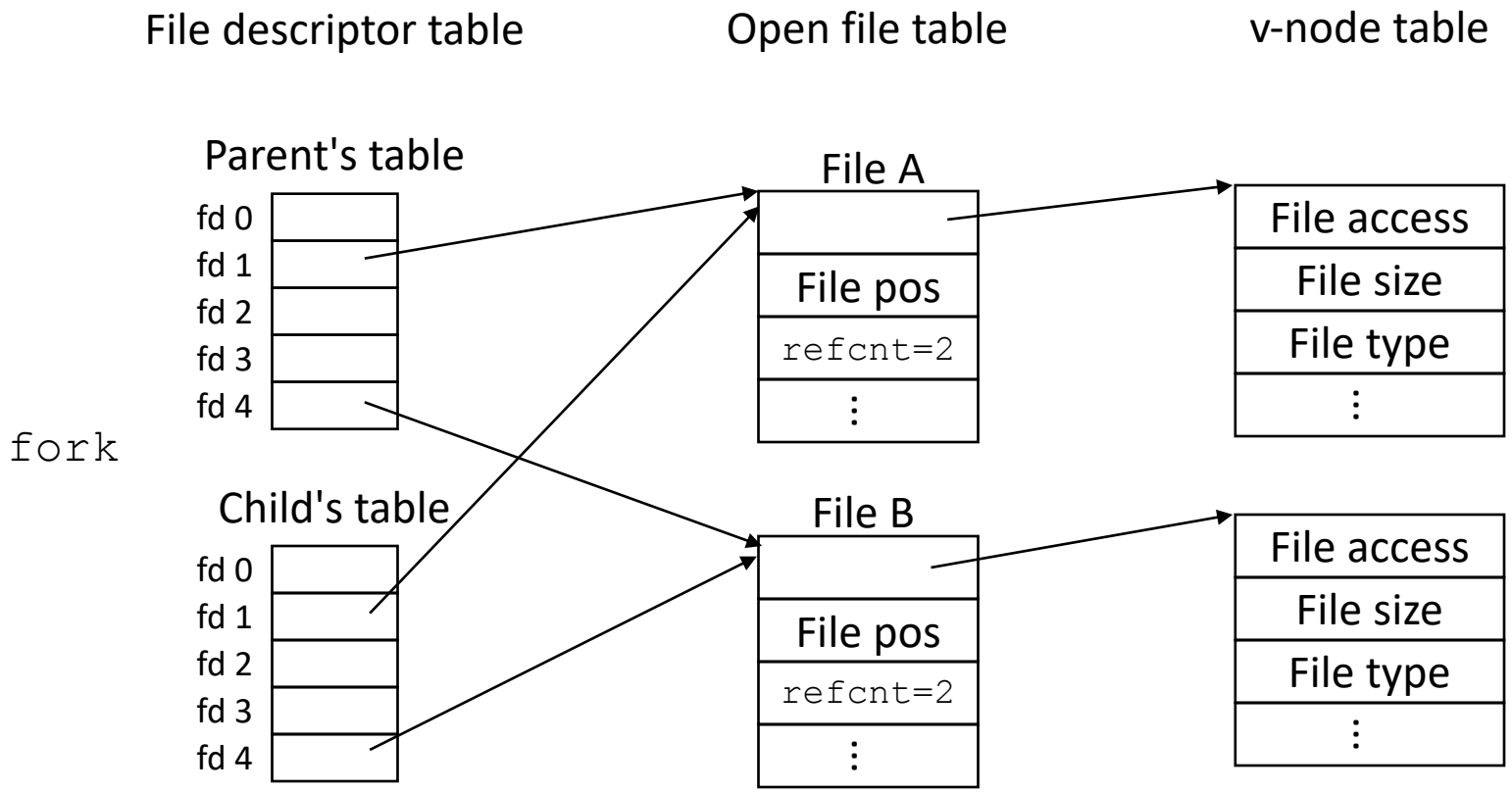
Processes and Files

- A child process inherits all file descriptors from its parent on `fork`



Processes and Files

- A child process inherits all file descriptors from its parent on `fork`



Practice with Processes and Files

Suppose the file `foobar.txt` consists of the six characters: `foobar`

What is printed when the following program is run?

```
int main(int argc, char ** argv) {
    char c;

    int fd1 = open("foobar.txt", O_RDONLY);

    if(fork() == 0){
        read(fd, &c, 1);
        return 0;
    } else {
        wait();
        read(fd, &c, 1);
        printf("c = %c\n", c);
        return 0;
    }
}
```

Practice with Processes and Files

Suppose the file `foobar.txt` consists of the six characters: `foobar`

What is printed when the following program is run?

```
int main(int argc, char ** argv) {
    char c;

    int fd1 = open("foobar.txt", O_RDONLY);

    if(fork() == 0){
        read(fd, &c, 1);
        return 0;
    } else {
        wait();
        read(fd, &c, 1);
        printf("c = %c\n", c);
        return 0;
    }
}
```

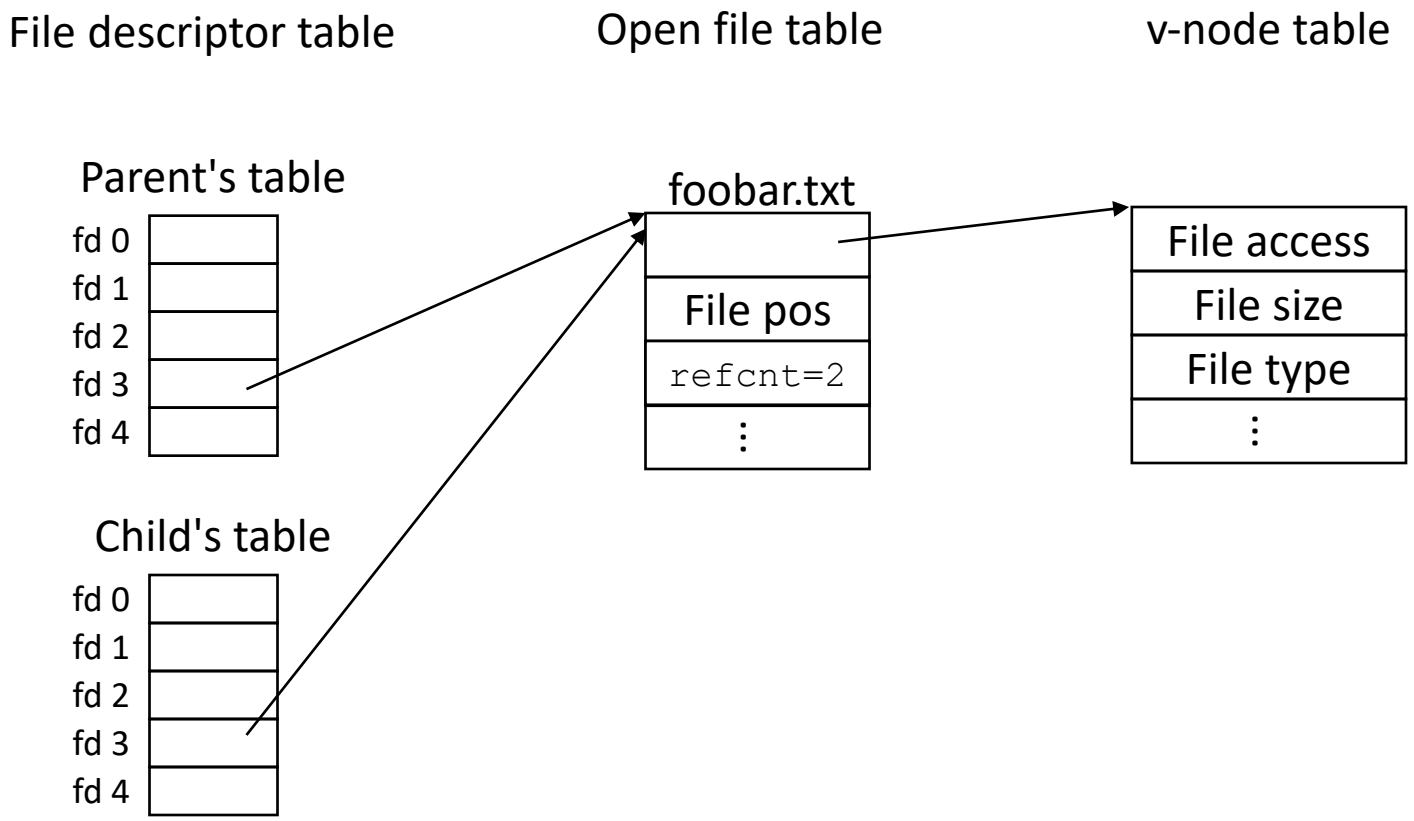
Child

Parent

f

f

Practice with Processes and Files



I/O Redirection

DESCRIPTION
The `dup()` system call creates a copy of the file descriptor old `fd`, using the Lowest-numbered unused file descriptor for the new descriptor.

After a successful return, the old and new file descriptors may be used interchangeably.

Examples of I/O redirection

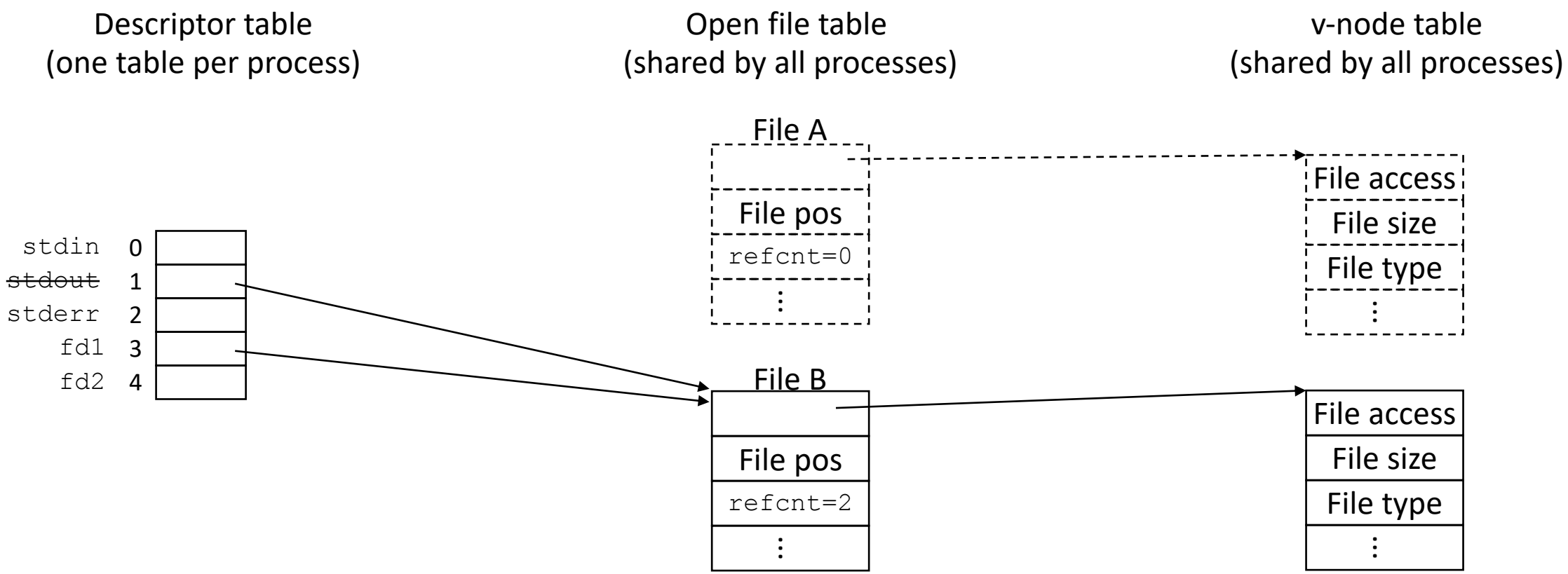
- You can redirect `stdout` to a file: `./ringbuf 4 > testout.txt`
- You can redirect a file to `stdin`: `./ringbuf 4 < testin.txt`
- You can pipe `stdout` to `stdin`: `cpp file.c | cparse | cgen | as > file.o`

- I/O redirection uses a function called dup2

```
int dup2(int oldfd, int newfd);
```

- Returns file descriptor if OK, -1 on error

I/O Redirection



```
int dup2(int oldfd, int newfd);
```

Practice with I/O Redirection

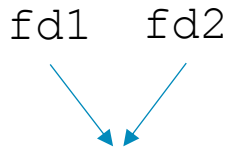
Suppose the file `foobar.txt` consists of the six characters: `foobar`

What is printed when the following program is run?

```
int main(){
  char c;

  int fd1 = open("foobar.txt", O_RDONLY);
  int fd2 = open("foobar.txt", O_RDONLY);

  read(fd2, &c, 1);
  dup2(fd2, fd1);
  read(fd1, &c, 1);

  printf("c = %c\n", c);

  return 0;
}
```

```
int dup2(int oldfd, int newfd);
```

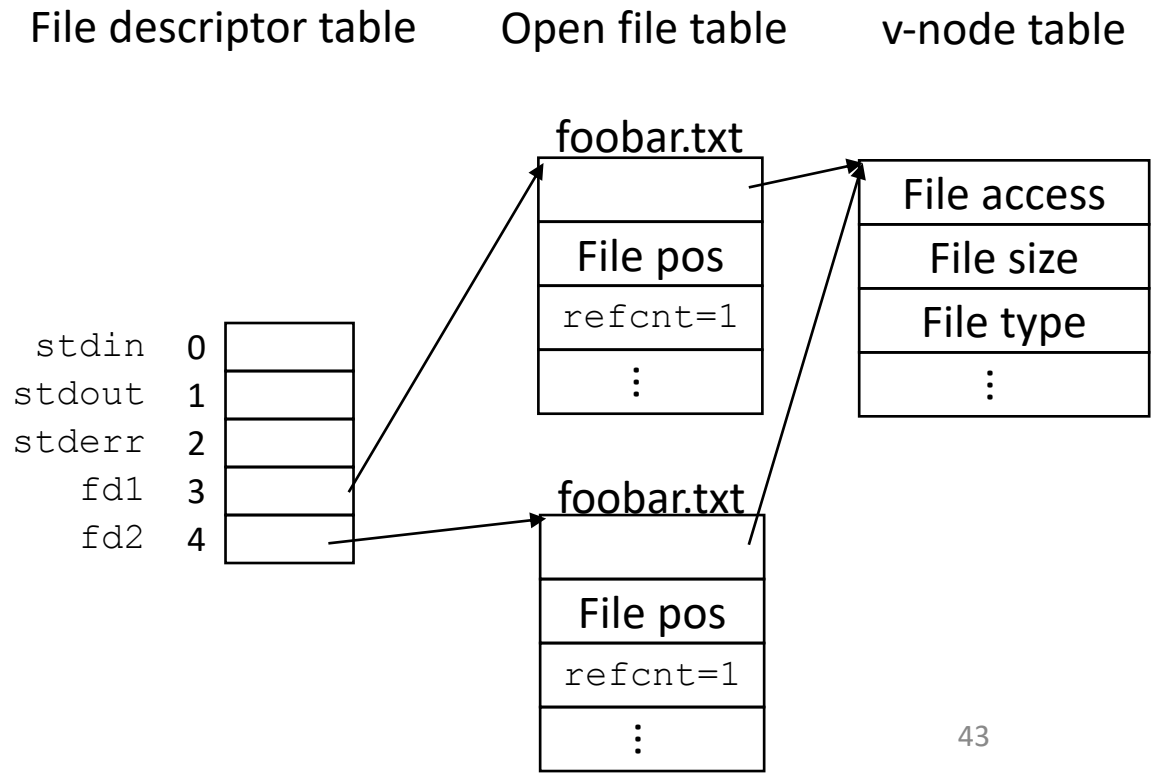
Practice with I/O Redirection



Suppose the file `foobar.txt` consists of the six characters: `foobar`

What is printed when the following program is run?

```c
int main(){
  char c;

  int fd1 = open("foobar.txt", O_RDONLY);
  int fd2 = open("foobar.txt", O_RDONLY);

  read(fd2, &c, 1);
  dup2(fd2, fd1);
  read(fd1, &c, 1);

  printf("c = %c\n", c);

  return 0;
}
```



```
int dup2(int oldfd, int newfd);
```

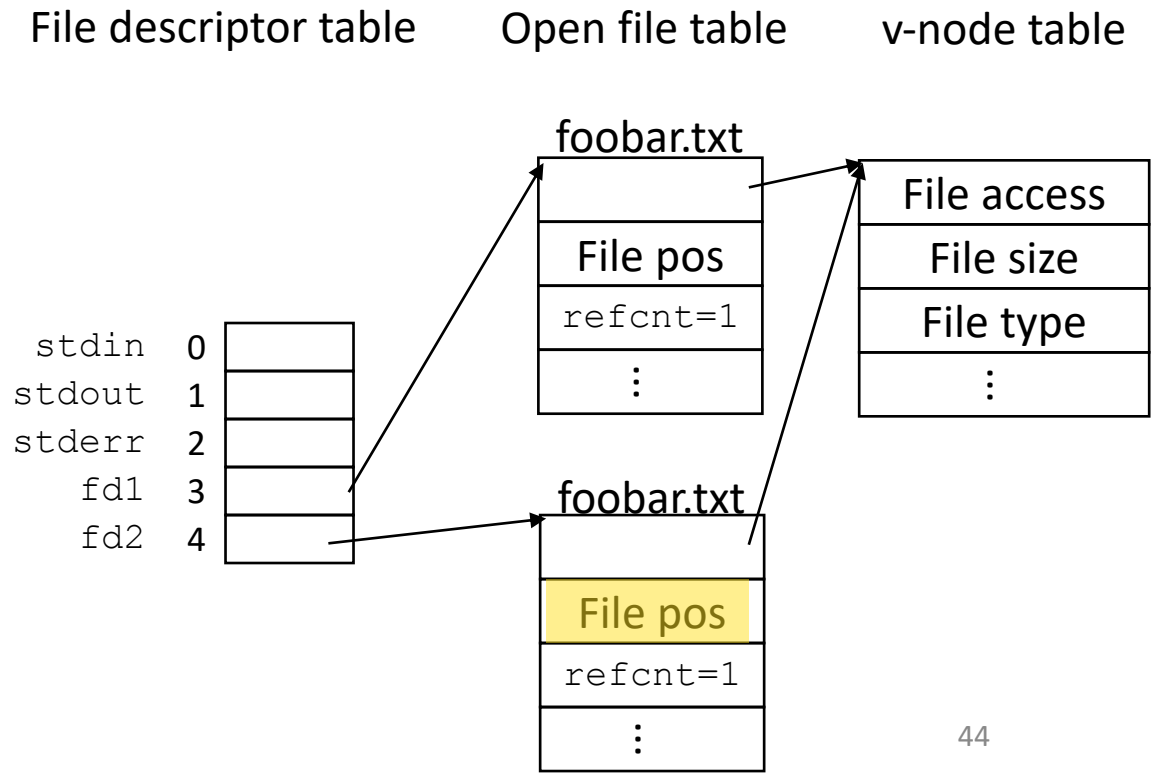
Practice with I/O Redirection

fd1 fd2

Suppose the file `foobar.txt` consists of the six characters: `foobar`

What is printed when the following program is run?

```c
int main(){
  char c;

  int fd1 = open("foobar.txt", O_RDONLY);
  int fd2 = open("foobar.txt", O_RDONLY);

  read(fd2, &c, 1);
  dup2(fd2, fd1);
  read(fd1, &c, 1);

  printf("c = %c\n", c);

  return 0;
}
```



```
int dup2(int oldfd, int newfd);
```

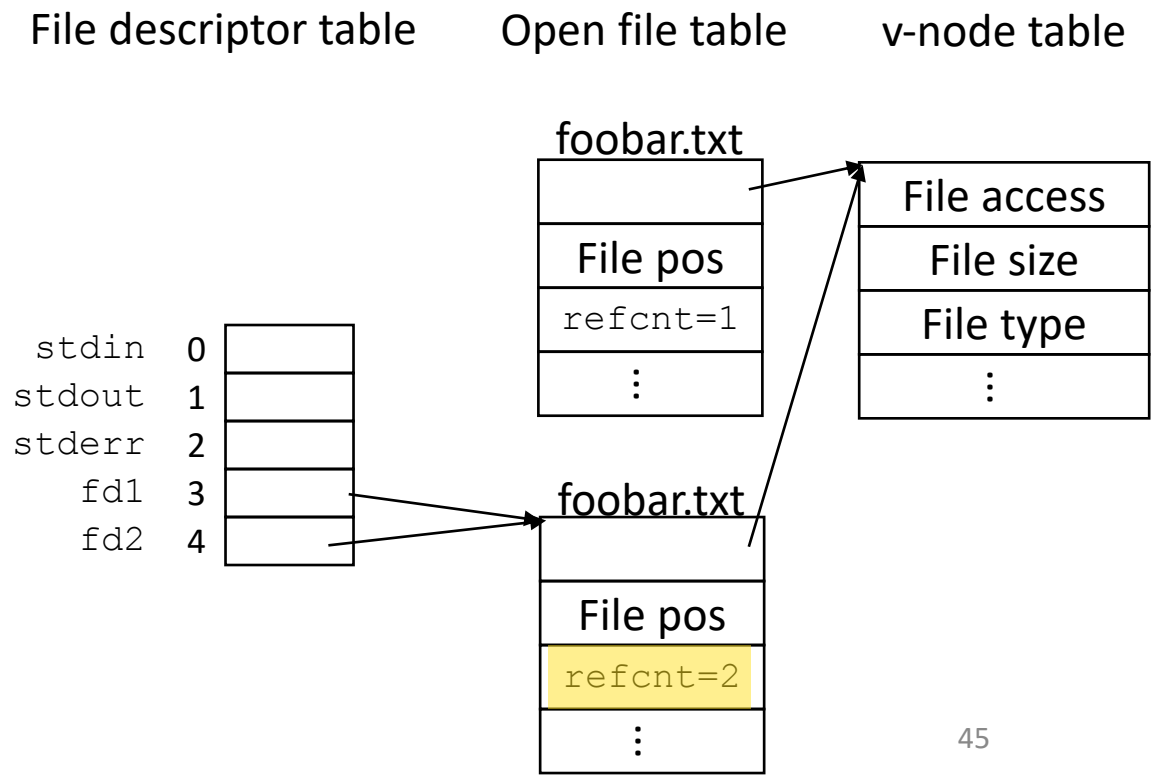
Practice with I/O Redirection

fd1
fd2

Suppose the file `foobar.txt` consists of the six characters: `foobar`

What is printed when the following program is run?

```c
int main(){
  char c;

  int fd1 = open("foobar.txt", O_RDONLY);
  int fd2 = open("foobar.txt", O_RDONLY);

  read(fd2, &c, 1);
  dup2(fd2, fd1);
  read(fd1, &c, 1);

  printf("c = %c\n", c);

  return 0;
}
```



```
int dup2(int oldfd, int newfd);
```

Practice with I/O Redirection

fd1
fd2

Suppose the file `foobar.txt` consists of the six characters: `foobar`

What is printed when the following program is run?

```c
int main(){
  char c;

  int fd1 = open("foobar.txt", O_RDONLY);
  int fd2 = open("foobar.txt", O_RDONLY);

  read(fd2, &c, 1);
  dup2(fd2, fd1);
  read(fd1, &c, 1);

  printf("c = %c\n", c);

  return 0;
}
```