Unix File I/O

Dr. Kerry Veenstra

CSE 13S

May 19, 2023

Using System Calls for File I/O

- See Section 2 of the man Pages
  - For example: **`$ man 2 write`**
- We won't be using Section 3 functions like fopen() or printf().
  - Why? We want to control the buffering.

Comparison: Buffered I/O vs. Unbuffered I/O

```
FILE *f = fopen(filename, mode);
```

```
FILE *f = fopen("file.txt", "r");
```

vs.

```
int fd = open(filename, flags, permissions);
```

```
int fd = open("file.txt", RD_RDONLY, 0);
```

```
int res = read(int fd, char *buf, int n);
```

- **`fd`** = file descriptor
  - `0` (**`STDIN_FILENO`**)
  - result of **`open()`** or **`creat()`**
- **`buf`** = pointer to a buffer
- **`n`** = number of bytes to read
- **`res`** = number of bytes actually read (**`res < n`** is possible)
  - or `0` to indicate an end-of-file (EOF)
  - or `-1` to indicate an error (see errno)

```
int res = write(int fd, char *buf, int n);
```

- **fd** = file descriptor
  - **1** (**STDOUT_FILENO**) or **2** (**STDERR_FILENO**)
  - result of **open()** or **creat()**
- **buf** = pointer to a buffer
- **n** = number of bytes to write
- **res** = number of bytes actually written (**res < n** is possible)
  - or **-1** to indicate an error (see errno)

```
int fd = open(char *name, int flags, int perms);
```

- **`name`** = file to open (file must exist, but see **`creat()`**)
- **`flags`** = O_RDONLY, O_WRONLY, or O_RDWR
- **`perms`** = permissions code (usually `0`)
- **`fd`** = file descriptor
  - or `-1` to indicate an error (see errno)

```
int fd = creat(char *name, int perms);
```

- **`name`** = file to open (overwrites if it exists)
  - Equivalent to **`open(name, O_WRONLY|O_CREAT|O_TRUNC, perms)`**
- **`perms`** = permissions code (also see **`man chmod`**)
- **`fd`** = file descriptor
  - or **`-1`** to indicate an error (see errno)

```
int res = close(int fd);
```

- **`fd`** = file descriptor
  - result of **`open()`** or **`creat()`**
- **`res`** = return code
  - **`0`** to indicate success
  - **`-1`** to indicate an error (see errno)

```
int res = unlink(char *name);
```

- **`name`** = name of file to delete
- **`res`** = return code
  - `0` to indicate success
  - `-1` to indicate an error (see errno)

```
int res = lseek(int fd, long offset, int origin);
```

- **`fd`** = file descriptor
  - result of **`open()`** or **`creat()`**
- **`offset`** = offset in bytes relative to the origin
- **`origin`**
  - **`SEEK_SET = 0`** = beginning of file
  - **`SEEK_CUR = 1`** = current position
  - **`SEEK_END = 2`** = end of file
- **`res`** = return code
  - `0` to indicate success
  - `-1` to indicate an error (see errno)

Buffering Example

```
#include <stdio.h>
#include <unistd.h>
/*
* Demonstration that combining buffered I/O like fprintf()
* and unbuffered I/O like write() can lead to out-of-order
* data in the output file.
*/
int main(void) {
  FILE *f = fopen("out.txt", "w");
  int fd = fileno(f);
  fprintf(f, "1\n");
  write(fd, "2\n", 2);
  fprintf(f, "3\n");
  write(fd, "4\n", 2);
  fclose(f);
  return 0;
}
```

Buffering Example

```
#include <stdio.h>
#include <unistd.h>
/*
* Demonstration that combining buffered I/O like fprintf()
* and unbuffered I/O like write() can lead to out-of-order
* data in the output file.
*/
int main(void) {
  FILE *f = fopen("out.txt", "w");
  int fd = fileno(f);
  fprintf(f, "1\n");
  write(fd, "2\n", 2);
  fprintf(f, "3\n");
  write(fd, "4\n", 2);
  fclose(f);
  return 0;
}
```

Buffering Example

```
#include <stdio.h>
#include <unistd.h>
/*
* Demonstration that combining buffered I/O like fprintf()
* and unbuffered I/O like write() can lead to out-of-order
* data in the output file.
*/
int main(void) {
  FILE *f = fopen("out.txt", "w");
  int fd = fileno(f);
  fprintf(f, "1\n");
  write(fd, "2\n", 2);
  fprintf(f, "3\n");
  write(fd, "4\n", 2);
  fclose(f);
  return 0;
}
```

Buffering Example

```
veenstra@arm128:~/130-CSE$ clang -o buffering bufferi
ng.c
veenstra@arm128:~/130-CSE$ ./buffering
veenstra@arm128:~/130-CSE$ cat out.txt
2
4
1
3
veenstra@arm128:~/130-CSE$
```

Buffering Example

```
veenstra@arm128:~/130-CSE$ clang -o buffering bufferi
ng.c
veenstra@arm128:~/130-CSE$ ./buffering
veenstra@arm128:~/130-CSE$ cat out.txt
2
4
1
3
veenstra@arm128:~/130-CSE$
```