【Linux Programming】Day4(2)

Class	Understanding Linux/Unix Programming
Date	@March 15, 2022

【Ch2】Question 3: Can I write who?

2.5.2 Answer: Use open, read, and close

Opening a file: open

The open system call creates a connection between a process and a file. That connection is called a file descriptor and is depicted below as a tunnel from the process to the kernel.

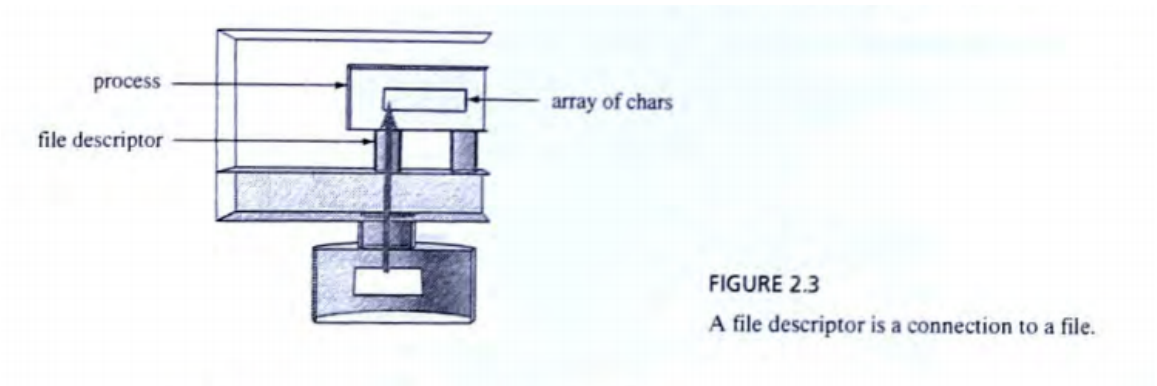

FIGURE 2.3
A file descriptor is a connection to a file.

The basic usage of the `open` is as follows:

open	
PURPOSE	Creates a connection to a file
INCLUDE	#include <fcntl.h>
USAGE	int fd = open(char *name, int how)
ARGS	name name of file how O_RDONLY, O_WRONLY, or O_RDWR
RETURNS	-1 on error int on success

To open a file, specify the name of the file and type of connection we want.

The three types are a connection for reading, a connection for writing, and a connection for reading and writing. The header file `/usr/include/fcntl.h` contains the definitions of `O_RDONLY`, `O_WRONLY`, and O_RDWR

Opening a file is a kernel service. The open system call is a request from our program to the kernel. If the kernel detects an error, it returns the value -1. There are many sorts of errors:

- The file might not exist
- It might exist, but we might not have permission to read it
- It might be in a directory to which we do not have access to

If things go well, the kernel returns a small positive integer. The number is called a file descriptor and is the identifier for the connection.

We use the file descriptor for all operations on the connection.

Reading Data from a File: read

We can read data from a file descriptor into a process:

read	
PURPOSE	Transfer up to qty bytes from fd to buf
INCLUDE	#include <unistd.h>
USAGE	ssize_t numread = read(int fd, void *buf, size_t qty)
ARGS	fd source of data buf destination for data qty number of bytes to transfer
RETURNS	-1 on error numread on success

The read system call asks the kernel to transfer `qty` bytes of data from the file descriptor `fd` to the array `buf` in the memory space of the calling process.

The kernel does what it can and returns a result. It the request fails, `read` returns -1. Otherwise, the call returns the number of bytes transferred.

We might get fewer bytes than the number being asked. This is because the file might not have as many bytes as we ask.

Closing a File: close

When we are done reading or writing data to a file descriptor, we close it. The `close` call is as follows:

close	
PURPOSE	Closes a file
INCLUDE	#include <unistd.h>
USAGE	int result = close(int fd)
ARGS	fd file descriptor
RETURNS	-1 on error 0 on success

The `close` system call hangs up the connection specified by file descriptor `fd`. `close` returns -1 on error. For example, trying to close a file descriptor that does not refer to an open file is an error. Other errors are described in the manpage.

2.5.3 Write who.c

```c
/* who1.c  - a first version of the who program
 *              open, read UTMP file, and show results
 */
#include        <stdio.h>
#include        <utmp.h>
#include        <fcntl.h>
#include        <unistd.h>

#define SHOWHOST        /* include remote machine on output */

int main()
{
        struct utmp     current_record; /* read info into here       */
        int             utmpfd;         /* read from this descriptor */
        int             reclen = sizeof(current_record);

        if ( (utmpfd = open(UTMP_FILE, O_RDONLY)) == -1 ){
                perror( UTMP_FILE );    /* UTMP_FILE is in utmp.h    */
                exit(1);
        }
        while ( read(utmpfd, &current_record, reclen) == reclen )
                show_info(&current_record);
        close(utmpfd);
        return 0;                       /* went ok */
}
```