【Linux Programming】Day4

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

【Ch2】How does who do it?

Read the .h files

On most Unix machines, the header files for system information are stored in the directory called `/usr/include`. When the C compiler sees a line such as

```
#include <stdio.h>
```

it looks for that file in `/usr/include`.

Note: The `utmp.h` file on my machine is in the path `/usr/include/bits/utmp.h`

We use `more` to read the file:

```
$ more /usr/include/utmp.h
 ...
#define UTMP_FILE       "/var/adm/utmp"
#define WTMP_FILE       "/var/adm/wtmp"

#include <sys/types.h>  /* for pid_t, time_t */

/*
 * Structure of utmp and wtmp files.
 *
 * Assuming these numbers is unwise.
 */

#define ut_name ut_user                 /* compatibility */
struct utmp {
        char    ut_user[32];            /* User login name */
        char    ut_id[14];              /* /etc/inittab id- IDENT_LEN in
                                         * init */
        char    ut_line[32];            /* device name (console, lnxx) */
        short   ut_type;                /* type of entry */
        pid_t   ut_pid;                 /* process id */
        struct exit_status {
            short       e_termination;  /* Process termination status */
            short       e_exit;         /* Process exit status */
        } ut_exit;                      /* The exit status of a process
                                         * marked as DEAD_PROCESS.
                                         */
        time_t  ut_time;                /* time entry was made */
        char    ut_host[64];            /* host name same as
                                         * MAXHOSTNAMELEN */
};
/* Definitions for ut_type                                          */
utmp.h (60%)
```

Log-in records, it appears, consist of eight members.

- The `ut_user` array stores the user log-in name.
- The `ut_id` array stores the device, which means the terminal from which the user connected.
- A few lines later in the struct are `ut_time` to store the log-in time and `ut_host` to store the name of the remote computer.

This struct contains other members. These do not correspond to items who displays, but they may come in handy in some situations.

2.4.1 We now know how who works

By reading the manual on the topics of `who` and `utmp` and by reading the header file `/usr/include/utmp.h`, we learned how `who` works.

`who` reads structures from a file. The file contains one structure for each log-in session. We know the exact layout of the structure.

The flow of information is shown in the figure below：

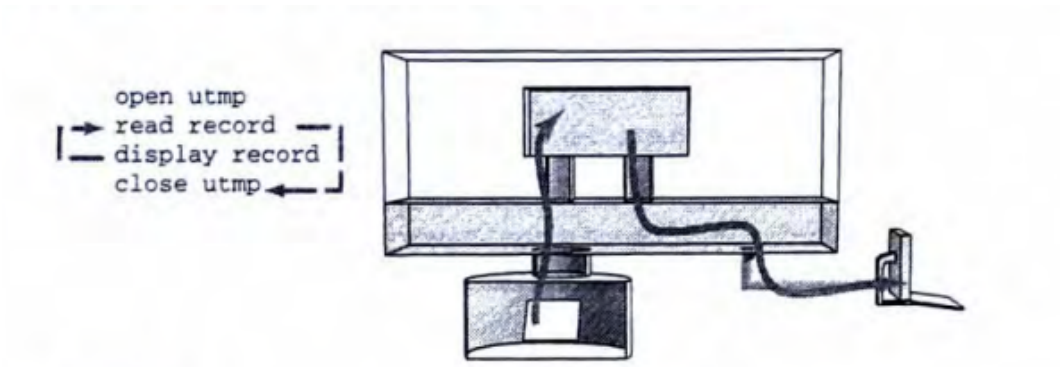

FIGURE 2.2

Data flow in the who command.

The file is an array, so who must read the records and print out the information. The simplest logic would be to read and print them one by one. *Could it be that easy?*

2.5 Question 3: Can I write who?

There are only two tasks we need to program:

- Read structs from a file
- Display the information stored in a struct

2.5.1 Question: How do I read structs from a file?

We use `getc` and `fgets` to read chars and lines from a file. *What about structs of raw data?*

Let's read the manual

We want to find manpages about `file` and `read`. The `-k` option accepts only one keyword, let's try file first:

```
$ man -k file
```

There are a lot of topics about files. There are 537 lines printed out.

We want to search those 537 lines for lines that contain the word "read." The Unix command called `grep` prints out lines that contain a specified pattern. We use grep in a pipeline as follows:

```
$ man -k file | grep read
```

```
$ man -k file | grep read
_llseek (2)           - reposition read/write file offset
fileevent (n)         - Execute a script when a channel becomes readable
                        or writable
gftype (1)            - translate a generic font file for humans to read
lseek (2)             - reposition read/write file offset
macsave (1)           - Save Mac files read from standard input
read (2)              - read from a file descriptor
readprofile (1)       - a tool to read kernel profiling information
scr_dump, scr_restore, scr_init, scr_set (3) - read (write) a curses
screen from (to) a file
tee (1)               - read from standard input and write to standard
                        output and files
$
```

The most promising item in the list is `read(2)`.

Look at the manpage in section 2 about `read`:

```
$ man 2 read
READ(2)                    System calls                    READ(2)

NAME
       read - read from a file descriptor

SYNOPSIS
       #include <unistd.h>

       ssize_t read(int fd, void *buf, size_t count);

DESCRIPTION
       read() attempts to read up to count bytes from file
       descriptor fd into the buffer starting at buf.
```



```
       If count is zero, read() returns zero and has no other
       results. If count is greater than SSIZE_MAX, the result
       is unspecified.

RETURN VALUE
       On success, the number of bytes read is returned (zero
       indicates end of file), and the file position is advanced
       by this number. It is not an error if this number is
       smaller than the number of bytes requested; this may hap-
       pen for example because fewer bytes are actually available
       right now (maybe because we were close to end-of-file, or
       because we are reading from a pipe, or from a terminal),
       or because read() was interrupted by a signal. On error,
       -1 is returned, and errno is set appropriately. In this
       case it is left unspecified whether the file position (if
       any) changes.
```

This system calls allows us to read a specified number of bytes from a file into a buffer.

We want to read one struct at a time, so we can use `sizeof(struct utmp)` to specify the number of bytes to read.

The file reads from a file descriptor. *How do we get one of those?*

The page contains a reference to `open(2)`. Running

```
$ man 2 open
```

reveals how `open` works. The page for open refers to close.