Con trỏ file

Trong các bài trước, chúng ta đã được giới thiệu các system call I/O quan trọng trong thao tác với file: open(), close(), read() và write(). Nhắc lại một chút về read() như sau:

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

System call read() đọc từ file có mô tả là "fd" một số lượng "count" byte và lưu vào vùng nhớ có địa chỉ là "buf". Câu hỏi đặt ra là, nếu bạn muốn đọc toàn bộ nội dung của file nhưng độ dài của file lớn hơn kích cỡ của "buf" thì làm sao để đọc tiếp nội dung file đó? Hoặc phải làm thế nào nếu bạn muốn đọc file từ byte thứ "n" bất kỳ trở đi thay vì đọc từ đầu file? Các vấn đề đó sẽ được giải quyết bằng một khái niệm "con trỏ file" (file offset hay file pointer).

Với mỗi file đang mở, kernel duy trì một giá trị gọi là file offset, còn được gọi là read-write offset. File offset là vị trí hiện tại của file mà lời gọi read() và write() tiếp theo sẽ bắt đầu. Ví dụ, nếu giá trị của file offset là 8, thì khi gọi read() hoặc write(), nó sẽ đọc hoặc ghi từ byte thứ 8 tính từ đầu file. Byte đầu tiên của file có giá trị offset là 0.

Khi mở một file, file offset sẽ trỏ đến đầu file và sẽ tự động cập nhật sau mỗi lần gọi read() hoặc write() để trỏ vào byte ngay sau các byte đã được đọc hoặc ghi. Vì vậy, các lệnh gọi read() và write() tiếp theo sẽ đọc từ byte tiếp theo cho đến khi hết file đó.

System call thay đổi file offset

Trong Linux, system call lseek() được dùng để thay đổi giá trị file offset của một file đang mở. Prototype của lseek() như sau:

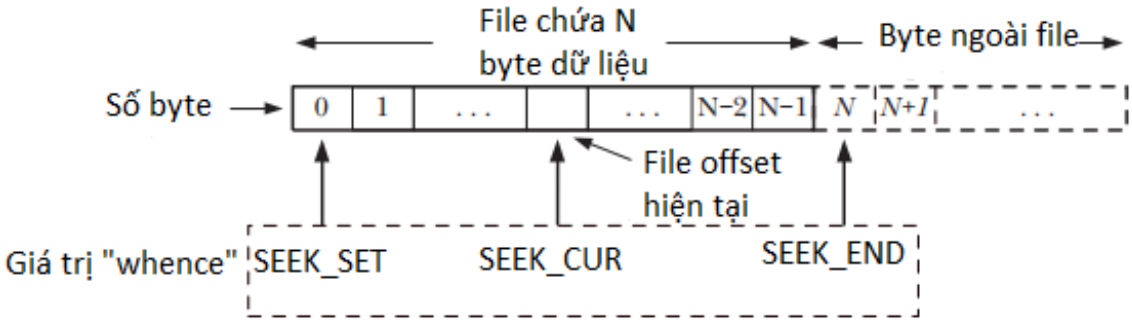
```
#include <unistd.h>

off_t lseek(int fd, off_t offset, int whence);
```

lseek() thay đổi file offset của một file có mô tả file "fd" một giá trị "offset" byte, tính từ điểm "whence". Whence có thể là một trong 3 giá trị sau:

- SEEK_SET: Bắt đầu file
- SEEK CUR: Giá trị file offset hiện tại của file
- SEEK_END: Điểm cuối cùng của file

Hình vẽ dưới đây sẽ mô tả rõ ràng hơn về giá trị "whence" cũng như file offset:



Hình 1: Vị trí các đối số SEEK_X trong lseek()

Nếu giá trị “whence” là SEEK_CUR hoặc SEEK_END, offset truyền vào có thể là số âm hoặc dương set con trỏ để tiến hoặc lùi so với vị trí hiện tại; nếu là SEEK_SET thì giá trị truyền vào phải là một số dương.

System call lseek() trả về giá trị offset mới của file. Ta có thể dùng lseek() để trỏ đến các vị trí khác nhau của file, ví dụ như sau:

lseek(fd, 0, SEEK_CUR)	Lấy giá trị hiện tại của file offset
lseek(fd, 0, SEEK_SET)	Trỏ đến đầu file
lseek(fd, 0, SEEK_END)	Trỏ đến cuối file (EOF)
lseek(fd, -1, SEEK_END)	Trỏ đến byte cuối cùng ngay trước EOF
lseek(fd, -10, SEEK_CUR)	Trỏ đến 10 byte ngay trước offset hiện tại
lseek(fd, 10000, SEEK_END);	Trỏ đến 1000 byte ngay sau EOF

Chú ý rằng, gọi lseek() chỉ thay đổi giá trị file offset được lưu ở kernel của một file có mô tả fd. Nó hoàn toàn không truy xuất vào bộ nhớ vật lý của file.

Chúng ta cũng không thể dùng lseek() cho các loại file đặc biệt như pipe, FIFO, socket và terminal. Khi dùng lseek() cho các file này, sẽ trả về fail, với errno là ESPIPE.

Chúng ta sẽ xem xét một đoạn code nhỏ sau đây để hiểu rõ hơn về cách sử dụng các system call thao tác file như open(), read(), write() và lseek. Các bạn hãy chạy chương trình này trên máy của mình để xem kết quả nhé:

```
#include<stdio.h>

#include<sys/stat.h>
#include<sys/types.h>
#include<fcntl.h>
#include<string.h>

int main(void)
{
	long position = 0;
	int fd = 0;
	char buf[] = "Hello, this is thevngeeks";
	char buf_read[8];
	char buf_read1[32];
	int length = 0;

	/*Mở một file hello.txt với cờ O_RDWR, ghi string buf[] vào file đó*/
	fd = open("hello.txt", O_RDWR);
	if(fd == -1)
	{
	   return -1;
	}
	length = write(fd, buf, sizeof(buf));

	/*dùng lseek() với đối số SEEK_CUR để xem giá trị file offset hiện tại*/
	position = lseek(fd, 0, SEEK_CUR);
	printf("Current position:%ld\n", position);

	/*lseek()với đối số SEEK_SET để trỏ về đầu file*/
	position = lseek(fd, 0, SEEK_SET);

	/*Dùng read() để đọc file, lưu vào string buf_read.
	  String đọc ra sẽ là buf[]: Hello, this is thevngeeks*/

	memset(buf_read, 0x0, sizeof(buf_read));
	length = read(fd, buf_read, sizeof(buf_read));
```

```
    buf_read[length] = '\0';
    printf("buf_read:%s\n", buf_read);

    /*File offset hiện tại là bao nhiêu sau read()?*/
    position = lseek(fd, 0, SEEK_CUR);
    printf("current position:%ld\n", position);
    /*Dùng write() ghi chuỗi “HELLO” vào file từ vị trí file offset*/
    write(fd, "HELLO",5);
    /*Trở về đầu file để đọc nội dung file*/
    position = lseek(fd, 0, SEEK_SET);

    /*Nội dung file sau khi đã ghi đè chuỗi “HELLO” vào file từ vị trí offset*/
    memset(buf_read1, 0x0, sizeof(buf_read1));
    length = read(fd, buf_read1, sizeof(buf_read1));
    buf_read1[length] = '\0';
    printf("buf_read:%s\n", buf_read1);

    /*Đóng file hello.txt bằng mô tả file fd*/
    close(fd);

    return 0;

}
```

Kết luận

File offset và system call lseek() giúp hỗ trợ thao tác đọc/ghi file trở lên hiệu quả và linh hoạt hơn rất nhiều khi chúng ta có thể điều chỉnh vị trí trong file để đọc hoặc ghi nội dung mà chúng ta muốn thay vì cứ phải đọc từ đầu file. Bài học sau sẽ giúp bạn theo dõi được trạng thái input/output để làm việc được với nhiều file trong tiến trình.