【C】Day9

Course	Advanced C
Study Date	@April 15, 2022

【Ch8】The Unix Interface

8.4 Random Access-Lseek

The system call `lseek` provides a way to move around in a file without reading or writing any data

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

sets the current position in the file whose descriptor is `fd` to `offset`, which is taken relative to the location specified by `origin`.

`origin` can be 0, 1, or 2 to specify that offset is to be measured from the beginning, from the current position, or from the end of the file respectively.

For example, to append to a file, seek to the end before writing:

```
lseek(fd, 0L, 2);
```

To get back to the beginning:

```
lseek(fd, 0L, 0);
```

With `lseek`, it is possible to treat files more or less like large arrays, at the price of slower access.

For example, the following function reads any number of bytes from any arbitrary place in a file. It returns the number read, or -1 on error.

```
#include <unistd.h>

// get: read n bytes from position pos
int get(int fd, long pos, char *buf, int n) {
  if(lseek(fd, pos, 0) >= 0)
    return read(fd, buf, n);
  return -1;
}
```

The return value from `lseek` is a long that gives the new position in the file, or -1 if an error occurs.

The standard library function `fseek` is similar to `lseek` except that the first argument is a `FILE*` and the return is non-zero if an error occurred.

8.5 Example-An Implementation of Fopen and Getc

A file pointer is a pointer to a structure that contains several pieces of information about the file:

- A pointer to a buffer
- A count of the number of characters left in the buffer
- A pointer to the next character position in the buffer
- The file descriptor
- Flags describing read/write mode, error status

The data structure that describes a file is contained in `<stdio.h>`.

The following is an excerpt from `<stdio.h>`

```
#define NULL 0
#define EOF (-1)
#define BUFSIZ 1024
#define OPEN_MAX 20 // Max #files open at once

typedef struct _iobuf {
  int cnt; // Characters left
```

```
  char *ptr; // Next character position
  char *base; // Location of buffer
  int flag; // Mode of file access
  int fd; // File descriptor
} FILE;
extern FILE _iob[OPEN_MAX];

#define stdin (&_iob[0])
#define stdout (&_iob[1])
#define stderr (&_iob[2])

enum _flags {
  _READ = 01,
  _WRITE = 02,
  _UNBUF = 04,
  _EOF = 010,
  _ERR = 020
};

int _fillbuf(FILE *);
int _flushbuf(int, FLE *);

#define feof(p) (((p)->flag & _EOF) != 0)
#define ferror(p) (((p)->flag &_ERR) != 0)
#define fileno(p) ((p)->fd)

#define getc(p) (--(p)->cnt >= 0 \? (unsigned char)*(p)->ptr++ : _fillbuf(p))
#define putc(x, p) (--(p)->cnt >= 0 \? *(p)->ptr++ = (x) : _flushbuf((x), p))

#define getchar() getc(stdin)
#define putchar(x) putc((x), stdout)
```