# Day21 A

Young W. Lim

2017-12-06 Wed

# Outline

1. Based on

2. File Processing
   - Files and Streams

# Based on

"C How to Program",
Paul Deitel and Harvey Deitel

# Files, Streams, and Programs

- views each file as a sequential stream of bytes
- when a file is opened, a stream is associated with it
- stream provide communication channels between files and programs

# Standard Streams

- three files and their associated streams
  are opened automatically
  when program execution begins

- standard input (stdin)
  - to read data from the keyboard
- standard output (stdout)
  - to print data on a screen
- standard error (stderr)
  - to print error messages

# FILE structure

- a FILE structure
  - contains information used to process the file
  - includes a file descriptor
- ```
  typedef struct {
  ...
  } FILE;
  ```
- opening a file returns a pointer to this FILE structure
- in C library's (GNU libc) source code.
  ```
  typedef struct _IO_FILE __FILE;
  ```
  ```

# _IO_FILE structure definition (1)

```
struct _IO_FILE {
  int _flags;       /* High-order word is _IO_MAGIC; rest is flags. */
#define _IO_file_flags _flags

  /* The following pointers correspond to the C++ streambuf protocol. */
  /* Note:  Tk uses the _IO_read_ptr and _IO_read_end fields directly. */
  char* _IO_read_ptr;   /* Current read pointer */
  char* _IO_read_end;   /* End of get area. */
  char* _IO_read_base;  /* Start of putback+get area. */
  char* _IO_write_base; /* Start of put area. */
  char* _IO_write_ptr;  /* Current put pointer. */
  char* _IO_write_end;  /* End of put area. */
  char* _IO_buf_base;   /* Start of reserve area. */
  char* _IO_buf_end;    /* End of reserve area. */
  /* The following fields are used to support backing up and undo. */
  char *_IO_save_base; /* Pointer to start of non-current get area. */
  char *_IO_backup_base;  /* Pointer to first valid character of backup area */
  char *_IO_save_end; /* Pointer to end of non-current get area. */

  struct _IO_marker *_markers;

  struct _IO_FILE *_chain;
```

# _IO_FILE structure definition (2)

```
  int _fileno;
#if 0
  int _blksize;
#else
  int _flags2;
#endif
  _IO_off_t _old_offset; /* This used to be _offset but it's too small.  */

#define __HAVE_COLUMN /* temporary */
  /* 1+column number of pbase(); 0 is unknown. */
  unsigned short _cur_column;
  signed char _vtable_offset;
  char _shortbuf[1];

  /*  char* _save_gptr;  char* _save_egptr; */

  _IO_lock_t *_lock;
#ifdef _IO_USE_OLD_IO_FILE
};
```

$3$

$3$

# accessing file descriptor from a FILE pointer

- member _fileno : file descriptor
- fileno() returns a file descriptor

```
#include <stdio.h>

int main(void) {
  FILE *fp;

  fp = fopen("t.c", "r");           3
                                    3
  printf("%d\n", fileno(fp));

  printf("%d\n", fp->_fileno);

}
```

# file descriptor, file, inode tables

- a file descriptor table
  - file descriptor
    - a handle used to access a file or an i/o resource
    - fd=0 (stdin), fd=1 (stdout), fd=2 (stderr)
    - an index into an array (the open file table)
- a file table
  - contains mode information (read, write, append . . . )
  - contains index into inode table (Linux)
- an inode table
  - describes the actual underlying files

from http://www.linfo.org/inode.html

# inode

- inode
  - stores all the information about a file or a directory
    - metadata: size, ownership, access permissions, timestamps, a reference counter file type
  - except its name and and its actual data
  - each file has inode number
    - unique throughout the filesystem
  - file name -> inode number -> inode table entry -> inode

from http://www.linfo.org/inode.html

# inode pointer structure

- Unix File System (UFS) and Linux ext3 cases
  - use inode pointer structure
  - to list the addresses of a file's data blocks
  - an inode has its metadata and 15 pointers
    - 12 pointers to direct blocks (*)
    - 1 singly indirect pointer (**)
    - 1 doubly indirect pointer (***)
    - 1 triply indirect pointer (****)

from https://en.wikipedia.org/wiki/Inode_pointer_structure

# Opening a file

- a FILE structure
  - contains information used to process the file
  - includes a file descriptor
- a file descriptor
  - an index into an array : the open file table
  - each array element contains a file control block
- a file control block (FCB)
  - OS uses to administer a particular file
- opening a file returns a pointer to FILE structure