File I/O - Chapter 3

Most UNIX I/O can be done with 5 system calls:

open, read, write, close, lseek

- open(2) (man open)
- #include <fcntl.h>
- int open(const char *path, int flags, mode_t mode)
- flags
  - O_RDONLY, O_WRONLY, O_RDWR
  - O_APPEND, O_CREAT, O_EXCL, O_TRUNC
- mode (needed only with O_CREAT, 0 or not there)
  - number representing permissions on creation
  - C -- 777 vs 0777 vs 0x777
  - permissions:
    - read (r--, 4)
    - write (-w-, 2)
    - execute (--x, 1) (lookup in directory)

Open (page 2)

- Who ---
  - user (0700)
  - group (0070)
  - others (0007)

- umask(2) -- remove mode bits during open

- fd = open ("/file/name", O_RDWR|O_CREAT, 0700);

- fd return value
  - fd >= 0 - open successful
  - fd < 0 - open unsuccessful, error in errno

Read and Write

ssize_t read (int fd, void *buf, size_t nbytes)
ssize_t write (int fd, const void *buf, size_t nbytes)

- fd is value returned by open
- buf is usually an array of data
- nbytes is size of buf (read) or size of data to write

- returned value
  - > 0 number of bytes read or written
  - 0   EOF
  - < 0 error

lseek

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

- fd -- number returned by open
- off_t -- Not necessarily a long or int ... could be a quad!
- "file pointer" in the file
- whence
  - SEEK_SET - offset bytes from start of file
  - SEEK_CUR - offset bytes from current location
  - SEEK_END - offset bytes from end of the file
- return value -- new location of the file pointer (or error)

- ret = lseek (fd1, (off_t)0, SEEK_CUR) ?
- ret = lseek (fd1, (off_t)1000000, SEEK_END) ?
- program onemeg.c

Close & more

int close(int fd)

- Done using the file referenced by fd.

- I/O efficiency -- 1 byte vs 8k bytes.

- Example program using system calls.
  - cat.c

File Sharing

Process Table	Open File Table	V-Node
fd entry ------>	status/pointer--->	real file entry

- 2 independent processes open a file
  - p1: open().... AND p2: open()....
  - 2 process tables
  - 2 entries in open file table
  - 1 v-node entry

- 2 processes by fork
  - p1: open(); ... fork() to get p2
  - 2 process tables
  - 1 entry in open file table
  - 1 v-node entry

- Note: fork(); ... open() is the first one

Dup, dup2

```
int dup(int oldfd)
int dup2(int oldfd, int newfd)
```

- Copies oldfd pointer to new fd location.
- Does not change open file table, just the process fd table
- Two process fd entries point to same open file table entry

- dup -- returns first unused fd in table.

- dup2 -- if newfd is open, close newfd, then dup

```
if ( (fd = open (file, O_RDONLY, 0)) < 0) error
if (dup2(fd, 0) < 0) error ...
```

Other file related system calls (not complete)

- int fcntl(int fd, int cmd, ...)
  - duplicate fds
  - get/set fd flags
  - record locks

- int ioctl(int d, unsigned long request, void *argp)
  - "catchall"
  - special hardware control ...
  - e.g. terminal baudrate