Unix System Programming

File I/O

Chandravva Hebbi
Department of Computer Science and Engineering
chandravvahebbi@pes.edu

Unix System Programming

File I/O

Chandravva Hebbi

Department of Computer Science and Engineering

UNIX SYSTEM PROGRAMMING

Topics to be Covered

- ❖chown()
- ❖fcntl()
- ❖Ioctl()
- ❖stat()

chown() - Unix, Linux System Call

- These system calls change the owner and group of the file specified by path or by fd.
- Only a privileged process may change the owner of a file.

```
int chown(const char *path, uid_t owner, gid_t group);
int fchown(int fd, uid_t owner, gid_t group);
int lchown(const char *path, uid_t owner, gid_t group);
```

On success, zero is returned. On error, -1 is returned, and errno is set appropriately.

fcntl Function

The fcntl function can change the properties of a file that is already open.

int fcntl(int filedes, int cmd, ... /* int arg */);

Returns: depends on cmd if OK (see following), -1 on errore sysem calls.

The fcntl function is used for five different purposes.

- Duplicate an existing descriptor (cmd = F_DUPFD)
- Get/set file descriptor flags (cmd = F_GETFD or F_SETFD)
- Get/set file status flags (cmd = F_GETFL or F_SETFL)
- Get/set asynchronous I/O ownership (cmd = F_GETOWN or F_SETOWN)
- Get/set record locks (cmd = F_GETLK, F_SETLK, or F_SETLKW)

fcntl()

F_DUPFD: Duplicate the file descriptor filedes.

F_GETFD: Return the file descriptor flags for filedes as the value of the function. Currently, only one file descriptor flag is defined: the FD_CLOEXEC flag.

F_SETFD: Set the file descriptor flags for filedes. The new flag value is set from the third argument

F_GETFL:Return the file status flags for filedes as the value of the function.

File status flags for fcntl

- O_RDONLY: open for reading only
- O_WRONLY: open for writing only
- O_RDWR: open for reading and writing
- O_APPEND: append on each write
- O_NONBLOCK: nonblocking mode
- O_SYNC: wait for writes to complete (data and attributes)
- O_DSYNC: wait for writes to complete (data only)
- O_RSYNC: synchronize reads and writes
- O_FSYNC: wait for writes to complete (FreeBSD and Mac OS X only)
- O_ASYNC: asynchronous I/O (FreeBSD and Mac OS X only)

UNIX SYSTEM PROGRAMMING

fcntl()

F_SETFL: Set the file status flags to the value of the third argument (taken as an integer).
The only flags that can be changed are O_APPEND, O_NONBLOCK, O_SYNC, O_DSYNC, O_RSYNC, O_FSYNC, and O_ASYNC.

F_GETOWN: Get the process ID or process group ID currently receiving the SIGIO and SIGURG signals.

F_SETOWN: Set the process ID or process group ID to receive the SIGIO and SIGURG signals.

ioctl Function

The ioctl function has always been the catchall for I/O operations

❖Terminal I/O was the biggest user of this function

int ioctl(int filedes, int request, ...);

❖Returns: -1 on error, something else if OK

❖The ioctl function is included in the Single UNIX Specification only as an extension for dealing with STREAMS devices

●an open file descriptor

●a request code number

●either an integer value, possibly unsigned (going to the driver) or a pointer to data (either going to the driver, coming back from the driver, or both)

UNIX SYSTEM PROGRAMMING

stat, fstat, and lstat Functions

```
#include <sys/stat.h>

int stat(const char *restrict pathname, struct  stat *restrict buf);

int fstat(int filedes, struct stat *buf);

int lstat(const char *restrict pathname, struct  stat *restrict buf);
```

All three return: 0 if OK, -1 on error

Stat structure

```
struct stat {
      mode_t    st_mode;      /* file type & mode (permissions) */
      ino_t     st_ino;       /* i-node number (serial number) */
      dev_t     st_dev;       /* device number (file system) */
      dev_t     st_rdev;      /* device number for special files */
      nlink_t   st_nlink;     /* number of links */
      uid_t     st_uid;       /* user ID of owner */
      gid_t     st_gid;       /* group ID of owner */
      off_t     st_size;      /* size in bytes, for regular files */
      time_t    st_atime;     /* time of last access */
      time_t    st_mtime;     /* time of last modification */
      time_t    st_ctime;     /* time of last file status change */
      blksize_t st_blksize;   /* best I/O block size */
      blkcnt_t  st_blocks;    /* number of disk blocks allocated */
    };
```

File Types

Regular file.

Directory file.

Block special file.

Character special file.

FIFO

Socket

Symbolic link

File type macros

Macro	Type of file
S_ISREG()	regular file
S_ISDIR()	directory file
S_ISCHR()	character special file
S_ISBLK()	block special file
S_ISFIFO()	pipe or FIFO
S_ISLNK()	symbolic link
S_ISSOCK()	socket

THANK YOU

Chandravva Hebbi
Department of Computer Science and Engineering

chandravvahebbi@pes.edu