
Lập trình Hệ thống

Unix Programming

Part 2: C Programming

Nguyễn Quốc Tuấn

Network and Communication System Department
Faculty of Electronics and Communications
UNIVERSITY OF ENGINEERING AND TECHNOLOGY

C Programming

❑ FILE & THƯ MỤC

■ Thư mục

- + Cấu trúc hình cây
- + Hệ thống
- + Phân cấp
- + Chứa các file

■ File

- + file hệ thống
- + file người dung
- + Các

C Programming

❑ THƯ MỤC

▪ Thư viện vào ra POSIX

- Thư mục có thể được đọc bởi bất kỳ ai có quyền truy cập. Nhưng chỉ kernel mới có thể ghi vào một thư mục hệ thống để bảo vệ hệ thống tệp tin
- Định dạng thực tế của một thư mục phụ thuộc vào việc triển khai Hệ thống UNIX và thiết kế của hệ thống tệp

```
#include <dirent.h>
```

```
DIR *opendir(const char *pathname);
```

```
DIR *fdopendir(int fd);
```

Both return: pointer if OK, NULL on error

```
struct dirent *readdir(DIR *dp);
```

Returns: pointer if OK, NULL at end of directory or error

```
void rewinddir(DIR *dp);
```

```
int closedir(DIR *dp);
```

Returns: 0 if OK, -1 on error

```
long telldir(DIR *dp);
```

Returns: current location in directory associated with dp

```
void seekdir(DIR *dp, long loc);
```

C Programming

❑ THƯ MỤC

▪ Thư viện vào ra POSIX

- Mô tả lệnh ls(1) trong shell

```
#include "apue.h"
#include <dirent.h>
Int main(int argc, char *argv[])
{
    DIR *dp;
    struct dirent *dirp;
    if (argc != 2)
        err_quit("usage: ls directory_name");
    if ((dp = opendir(argv[1])) == NULL)
        err_sys("can't open %s", argv[1]);
    while ((dirp = readdir(dp)) != NULL)
        printf("%s\n", dirp->d_name);
    closedir(dp);
    exit(0);
}
```

```
struct dirent
{
    ino_t d_ino;    /* i-node number */
    char d_name[]; /* null-terminated filename */
}
```

C Programming

❑ QUẢN LÝ THƯ MỤC

▪ Tạo/xóa/thay đổi thư mục

- Trong shell có thể nhìn thấy quyền của thư mục/file thông qua lệnh **ls -l**
Ví dụ quyền truy cập của lệnh passwd có thể thấy

```
- - -x- -s- -x 1 bin auth 57825 Apr 13 12:21 /bin/passwd
```

- Các hàm thay đổi quyền truy cập

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

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

```
int mkdir(const char *path, mode_t mode);    //Tạo thư mục
```

```
#include <unistd.h>
```

```
int rmdir(const char *path);                //Xóa thư mục
```

```
int chdir(const char *path);                //Chuyển thư mục
```

C Programming

❑ QUẢN LÝ THƯ MỤC

■ Tạo/xóa/đổi thư mục

- Lệnh **getcwd(S)** trả lại chuỗi kí tự (max 64 byte) biểu diễn thư mục hiện hành (pathname)

```
#include <unistd.h>
```

```
char *getcwd(char *buf, size_t size);
```

```
#include <unistd.h>
#include <stdlib.h>

main()
{ char *cwd ;
  cwd = getcwd(0, 64)
  if (cwd != NULL)
    printf("current working directory %s\n", cwd)
  else
    pexit ( "getcwd" ) ;
  if (chdir("/tmp") == -1)
    pexit ( "chdir" ) ;
  else
    (   cwd = getcwd(0, 64) ;
      if (cwd != NULL)
        printf("new current working directory");
      else
        pexit ("getcwd") ;
    )
}
```

C Programming

❑ QUẢN LÝ THƯ MỤC

■ Thay đổi sở hữu thư mục/file

- Trong shell, sở hữu của thư mục/file có thể thấy thông qua lệnh **ls -l** và có thể được thay đổi

```
#include <unistd.h>
```

```
int chown(const char *pathname, uid_t owner, gid_t group);
```

```
int lchown(const char *path, uid_t owner, gid_t group)
```

```
int fchown(int fildes, uid_t owner, gid_t group);
```

return: 0 if OK, -1 on error

- Lệnh **chown(s)** thay đổi UID và GID của file
- **fchown(S)** thay đổi quyền sở hữu của tệp đang mở được tham chiếu bởi đối số fd. Vì nó hoạt động trên một tệp đã được mở, nó không thể được sử dụng để thay đổi quyền sở hữu của một liên kết tượng trưng
- Lệnh **lchown()** hoạt động tương tự nhưng cho các file liên kết

C Programming

❑ QUẢN LÝ THƯ MỤC

■ Thay đổi quyền thư mục/file

- Trong shell, quyền của thư mục/file thông qua lệnh **ls -l** và có thể được thay đổi

#include <sys/stat.h>

*int **chmod**(const char *pathname, mode_t mode);*

*int **fchmod**(int fd, mode_t mode);*

*int **fchmodat**(int fd, const char *pathname, mode_t mode, int flag);*

return: 0 if OK, -1 on error

- Tương tự

+ Lệnh **chmod(s)** thay đổi quyền file xác định

+ **fchmod(S)** thay đổi quyền của tệp đang mở được tham chiếu bởi đối số fd. Vì nó hoạt động trên một tệp đã được mở,

- Lệnh **lchmodat()** hoạt động tương tự nhưng khi flag **AT_SYMLINK_NOTFOLLOW** được đặt - nó không thể được sử dụng để thay đổi quyền của một liên kết tượng trưng

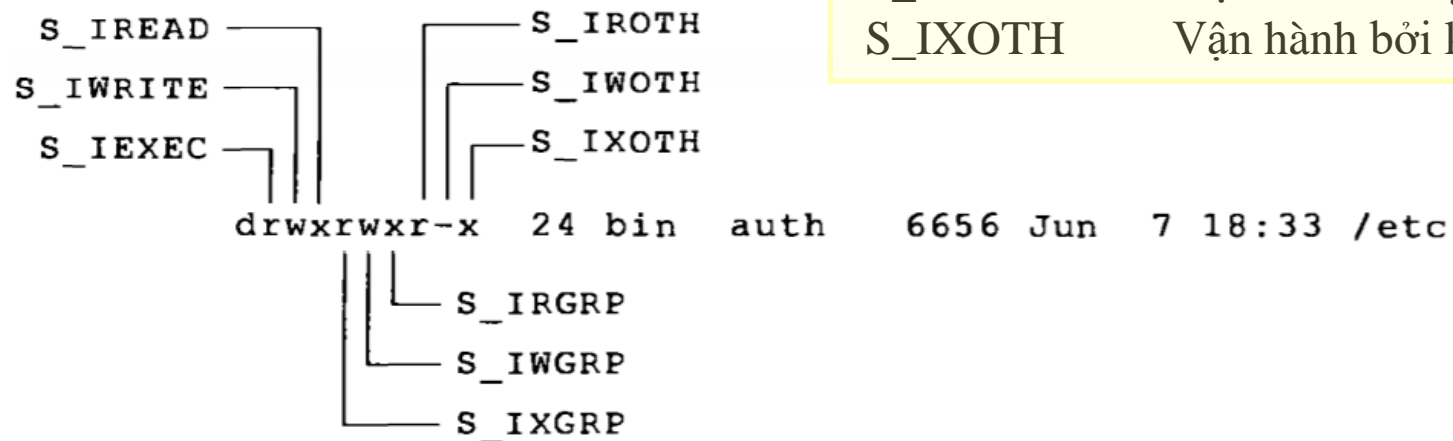
C Programming

❑ QUẢN LÝ THƯ MỤC

▪ Quyền của thư mục/file

- mode được xác định

- Ví dụ thư mục /etc



File mode passed to mkdir (S)

Mode	Description
S_IEXEC	Vận hành bởi owner.
S_IREAD	Đọc bởi owner.
S_IRGRP	Đọc bởi nhóm
S_IROTH	Đọc bởi khác
S_IWRITE	Viết bởi owner
S_IWGRP	Viết bởi group
S_IWOTH	Viết bởi khác
S_IXGRP	Vận hành bởi group
S_IXOTH	Vận hành bởi khác

C Programming

❑ QUẢN LÝ THƯ MỤC

▪ Ví dụ

```
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>

main(int argc, char *argv[])
{   int fd; rc;
    struct stat statbuf;
    printf (" Fiddling with inode\n");
    fd = open (" junk.out", O_RDONLY);
    assert (fd >= 0);
    printf (" changing file mode\n");
    rc = fchmod (fd, 0600);
    assert (rc == 0);
```

- Hàm xác nhận **assert()** đòi tự chuẩn đoán

```
if (getuid () == 0)
{   printf (" changing file owner \n ");
    rc = fchown (fd, 99, 99);
    assert (rc == 0); }
else
{   printf (" not changing file owner\n");
}
fstat (fd, &statbuf);
printf (" file mode = 0% (octal)\n", statbuf.st_mode);
printf("Owner uid = %d \n", statbuf.st_uid);
printf(" Owner gid = %d \n", statbuf.st_gid);
close(fd);
```

C Programming

❑ QUẢN LÝ THƯ MỤC

- Hàm xác nhận **assert()** cho phép các thông tin chẩn đoán phát hiện được ghi tới tệp lỗi chuẩn (Standard Error File)

#include <assert.h>

void assert(int expression)

Ví dụ:

```
#include <assert.h>
#include <stdio.h>
int main()
{ int a;   char str[50];
  printf("Input the integer number : \n");
  scanf("%d", &a);
  assert(a >= 10);
  printf("The value of a is: %d\n", a);
  printf("Input a string: ");
  scanf("%s", &str);
  assert(str != NULL);
  printf("String : %s\n", str);
  return(0);
}
```

C Programming

❑ QUẢN LÝ THƯ MỤC

Mode bitflag được sử dụng bởi `chmod()` và `lchmodat()`

Trường	bit	Mô tả
<i>S_ISGID</i>	<i>020#0</i>	<i>Đặt ID nhóm vận hành nếu là # 7, 5, 3, hay 1.</i>
<i>S_ISUID</i>		<i>Đặt ID user vận hành nếu là #6, 4, 2. or 0.</i>
<i>S_ISVTX</i>	<i>01000</i>	<i>Cất code sau khi vận hành.</i>
<i>S_IRWXU</i>	<i>00700</i>	<i>Read, write, execute bởi owner.</i>
<i>S_IRUSR</i>	<i>00400</i>	<i>Read bởi owner.</i>
<i>S_IWUSR</i>	<i>00200</i>	<i>Write bởi owner.</i>
<i>S_IXUSR</i>	<i>00100</i>	<i>Execute (tìm nếu là directory) bởi owner.</i>
<i>S_IRWXG</i>	<i>00070</i>	<i>Read, write, execute bởi group.</i>
<i>S_IRGRP</i>	<i>00040</i>	<i>Read bởi group.</i>
<i>S_IWGRP</i>	<i>00020</i>	<i>Write bởi group.</i>
<i>S_IXGRP</i>	<i>00010</i>	<i>Execute bởi group.</i>
<i>S_IRWXO</i>	<i>00007</i>	<i>Read, write, execute (search) bởi others.</i>
<i>S_IROTH</i>	<i>00004</i>	<i>Read bởi others.</i>
<i>S_IWOTH</i>	<i>00002</i>	<i>Write bởi others.</i>
<i>S_IXOTH</i>	<i>00001</i>	<i>Execute bởi others.</i>

C Programming

❑ QUẢN LÝ THƯ MỤC/FILE

▪ Quyền của thư mục/file

- Ví dụ : \$ cat cms.c

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

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
main()
```

```
(
```

```
mkdir("my dir", S_IREADIS_IWRITEIS_IEXEC);
```

```
system("ls -ld my_dir" |
```

```
chmod("my dir", S_IREADIS_IWRITEIS_IEXEC | S_IRGRP | S_IROTH);
```

```
system("ls -ld my_dir");
```

```
chown("my dir", 0; 3);
```

```
system("ls -ld my_dir" |
```

```
}
```

./cms

drwxr----- 2 qtuan qtuan 512 Oct 2 13:12 my_dir

drwxr--r-- 2 qtuan qtuan 512 Oct 2 13:12 my_dir

drwxr--r-- 2 root sys 512 Oct 2 13:12 my_dir

- *Lệnh system() cho phép chạy lệnh shell trong chương trình C*

❑ QUẢN LÝ THƯ MỤC/FILE

- Các hàm xác định trạng thái file: File thông thường, file thư mục, file block đặc biệt (cung cấp buffer trạng thái I/O), file kí tự đặc biệt (cung cấp non-buffer trạng thái I/O), file socket, file symlink..

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

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

```
int fstat(int fd, struct stat *buf );
```

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

```
int fstatat(int fd, const char *restrict pathname, struct stat *restrict buf, int flag);
```

return: 0 if OK, -1 on error

+ Lệnh *stat(s)* xác định trạng thái file

+ *fstat(S)* xác định trạng thái của tệp đang mở được tham chiếu bởi đối số fd. Vì nó hoạt động trên một tệp đã được mở,

+ Lệnh *lstat(S)* hoạt động tương tự trên file symlink và

+ *fsatat(S)* khi flag `AT_SYMLINK_NOTFOLLOW` được đặt - nó không thể được sử dụng để thay đổi quyền của một liên kết tượng trưng

C Programming

❑ QUẢN LÝ THƯ MỤC/FILE

▪ Cấu trúc stat

- Các tham số file được chỉ ra trong cấu trúc *stat*(.)

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

```
struct stat {  
    dev_t    st_dev;           /* thiết bị */  
    ino_t     st_ino;          /* inode */  
    mode_t    st_mode;        /* mode bảo vệ */  
    nlink_t    st_nlink;      /* số liên kết cứng */  
    uid_t     st_uid;         /* UID của chủ sở hữu */  
    gid_t     st_gid;         /* ID nhóm chủ sở hữu */  
    dev_t     st_rdev;        /* loại thiết bị (nếu inode  
    off_t     st_size;        /* kích thước bằng byte */  
    blksize_t st_blksize;     /* kích thước khối file */  
    blkcnt_t  st_blocks;      /* số khối được phân bổ */  
    time_t    st_atime;       /* lần truy cập cuối cùng */  
    time_t    st_mtime;       /* thời gian sửa đổi cuối */  
    time_t    st_ctime;       /* thời gian của trạng thái  
                               cuối cùng */  
};
```

C Programming

❑ QUẢN LÝ THƯ MỤC/FILE

▪ Ví dụ:

```
#include "apue.h"
Int main(int argc, char *argv[])
{ int i;
  struct stat buf;
  char *ptr;
  for (i = 1; i < argc; i++) {
    printf("%s: ", argv[i]);
    if (lstat(argv[i], &buf) < 0) {
      err_ret("lstat error");
      continue; }
    if (S_ISREG(buf.st_mode))
      ptr = "regular";
    else if (S_ISDIR(buf.st_mode))
```

```
      else if (S_ISCHR(buf.st_mode))
        ptr = "character special";
      else if (S_ISBLK(buf.st_mode))
        ptr = "block special";
      else if (S_ISFIFO(buf.st_mode))
        ptr = "fifo";
      else if (S_ISLNK(buf.st_mode))
        ptr = "symbolic link";
      else if (S_ISSOCK(buf.st_mode))
        ptr = "socket";
      else
        ptr = "*** unknown mode ***";
    printf("%s\n", ptr); }
  exit(0);
```

```
$ ./a.out /etc/passwd /dev/sr0 /dev/usb
/etc/passwd: regular
/etc: directory
/dev/sr0: block special
/dev/usb: symbolic link
```


❑ QUẢN LÝ FILE

▪ Thư viện vào ra chuẩn C

- Khi đó các lời gọi thao tác với file

<code>FILE *fopen</code>	<code>(char *path, char *mode);</code>	<code>//Mở file</code>
<code>FILE *fdopen</code>	<code>(int fildes, char *mode);</code>	<code>//Mở file dựa vào file descriptor</code>
<code>FILE *freopen</code>	<code>(char *path, char *mode, FILE*stream);</code>	<code>//Mở lại file</code>
<code>size_t fread</code>	<code>(void *ptr, size_t size, size_t nmemb,FILE*stream);</code>	<code>//Đọc nội dung từ file</code>
<code>size_t fwrite</code>	<code>(void *ptr, size_t size, size_t nmemb,FILE*stream);</code>	<code>//Ghi nội dung ra file</code>
<code>void clearerr</code>	<code>(FILE *stream);</code>	<code>//Xóa các lỗi liên quan đến file</code>
<code>int feof</code>	<code>(FILE *stream);</code>	<code>//Kiểm tra xem đã ở cuối file chưa</code>
<code>int ferror</code>	<code>(FILE *stream);</code>	<code>//Kiểm tra xem thao tác có bị lỗi hay không</code>
<code>int fileno</code>	<code>(FILE *stream);</code>	<code>//Lấy giá trị file descriptor từ một con trỏ file</code>
<code>int fseek</code>	<code>(FILE *stream, long offset, int whence);</code>	<code>//Di chuyển vị trí thao tác file</code>
<code>long ftell</code>	<code>(FILE *stream);</code>	<code>//Trả lại vị trí thao tác trong file</code>
<code>void rewind</code>	<code>(FILE *stream);</code>	<code>//Di chuyển vị trí thao tác về đầu file</code>
<code>int fgetpos</code>	<code>(FILE *stream, fpos_t *pos);</code>	<code>//Lấy vị trí thao tác</code>
<code>int fsetpos</code>	<code>(FILE *stream, fpos_t *pos);</code>	<code>//Đặt vị trí thao tác</code>

C Programming

❑ QUẢN LÝ FILE

▪ Thư viện vào ra chuẩn C

- Ví dụ

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

int main(int argc, char*argv[])
{
    FILE *fp;
    char buff[256];
    int sz;
    if((fp=fopen("vd.txt","wt"))==NULL)
    {
        perror("Failed to create file");
        return 1;
    }
    fputs("This is a test line",fp);
    fclose(fb);
}
```

*Các hàm thư viện vào ra chuẩn C
dung trong Unix, tuy thuận tiện
dễ dung nhưng có tốc độ hoạt
động không cao, thiếu linh hoạt*

C Programming

❏ FILE

▪ Thư viện vào ra POSIX

- Các hàm I/O UNIX (open, read, write, close và ioctl) sử dụng bộ mô tả tệp được định nghĩa trong <unistd.h> gọi STDIN_FILENO, STDOUT_FILENO, STDERR_FILENO

<i>System call</i>	<i>Description</i>
<i>open(S)</i>	<i>Mở file để đọc hay viết</i>
<i>creat(S)</i>	<i>Tạo file mới hoặc viết đè lên file đã tồn tại</i>
<i>close(S)</i>	<i>Đóng file khi dung thủ tục open()</i>
<i>dup(S)</i>	<i>Sao chép mô tả file đang mở</i>
<i>dup2(S)</i>	<i>Sao chép mô tả file đang mở xác định đích</i>
<i>lseek(S)</i>	<i>Đặt con trỏ tới byte xê dịch trong file</i>
<i>read(S)</i>	<i>Đọc một số byte xác định từ file</i>
<i>readv(S)</i>	<i>Đọc nội dung file vào trong đa buffer</i>
<i>write(S)</i>	<i>Viết một số byte xác định ào trong file</i>
<i>writev(S)</i>	<i>Viết vào file khi dung một số buffer</i>
<i>pipe(S)</i>	<i>Mở kênh truyền thông hai vhiềuuf</i>
<i>fcntl(S)</i>	<i>Cho phép điều khiển khi mở file</i>

C Programming

❑ QUẢN LÝ FILE

▪ Lệnh open(S)

Khuôn dạng `#include <fcntl.h>`

`int open(const char *path, int oflag, ... /* mode_t mode */);`

`int openat(int fd, const char *path, int oflag, ... /* mode_t mode */);`

return: file descriptor if OK, -1 on error

oflag	Description
<code>O_RDONLY</code>	Mở file chỉ đọc
<code>O_WRONLY</code>	Mở file chỉ viết
<code>O_RDWR</code>	Mở file cả đọc & viết
<code>O_APPEND</code>	Đặt con trỏ file tới cuối file
<code>O_CREAT</code>	Tạo file với permission tương ứng UID/GID
<code>O_EXCL</code>	Mở file bị lỗi nếu đã tồn tại
<code>O_NOCTTY</code>	Ngăn không cho file thành TTY kiểm soát nếu tên đường dẫn tham chiếu đến thiết bị đầu cuối.
<code>O_SYNC</code>	Tất cả các lần ghi tiếp theo sẽ buộc dữ liệu và siêu dữ liệu file được ghi vào đĩa trước khi quá trình ghi trở lại.
<code>O_TRUNC</code>	Nếu file tồn tại và thông thường thì kích thước file là zero
<code>O_NONBLOCK</code>	Tiến trình đọc ghi không bị block ..

C Programming

❑ QUẢN LÝ FILE

▪ **Lệnh creat(S)**

- create(S) sử dụng để tạo một tệp thông thường mới hoặc sửa đổi tệp hiện có để tiếp nhận các thuộc tính được chỉ định bởi các đối số
- Khuôn dạng; `#include <fcntl.h>`
`int creat(const char *path, mode_t mode)`
Returns: file descriptor opened for write-only if OK, -1 on error

▪ **Lệnh close(S)**

- Close(S) phá vỡ liên kết giữa bộ mô tả file và file mở liên quan của nó. Sau đó, bộ mô tả tệp có thể được kernel cấp phát trong các lần open(S) mở tiếp theo, pipe(S) fcntl (S) hoặc Dup (S)).
- Khuôn dạng; `#include <fcntl.h>`
`int close(int fildes)`
Returns: 0 if OK, -1 on error
- Khi một tiến trình chấm dứt, tất cả các file được tiến trình mở tự động được đóng lại. Lệnh close đặt errno để chỉ ra lý do thất bại nếu có.

C Programming


❑ QUẢN LÝ FILE

▪ Lệnh lseek(S)

- Bộ mô tả tệp sử dụng để tham chiếu đến con trỏ tệp nơi bắt đầu đọc hoặc ghi tiếp theo. Khi tệp được mở lần đầu, con trỏ tệp đặt = 0 hoặc bằng kích thước của tệp nếu O_APPEND sử dụng mở

- Khuôn dạng: `#include <unistd.h>`

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



<code>SEEK CUR</code>	<code>(0)</code>
<code>SEEK END</code>	<code>(1)</code>
<code>SEEK SET</code>	<code>(2)</code>

Returns: new file offset if OK, -1 on error

▪ Lệnh dup(S) & dup2(S)

- Lệnh dup(S) được sử dụng để sao chép bộ mô tả file đã tồn tại

Khuôn dạng: `#include <unistd.h>`

`int dup(int fd);`

`int dup2(int fd, int fd2);`

return: new file descriptor if OK, -1 on error

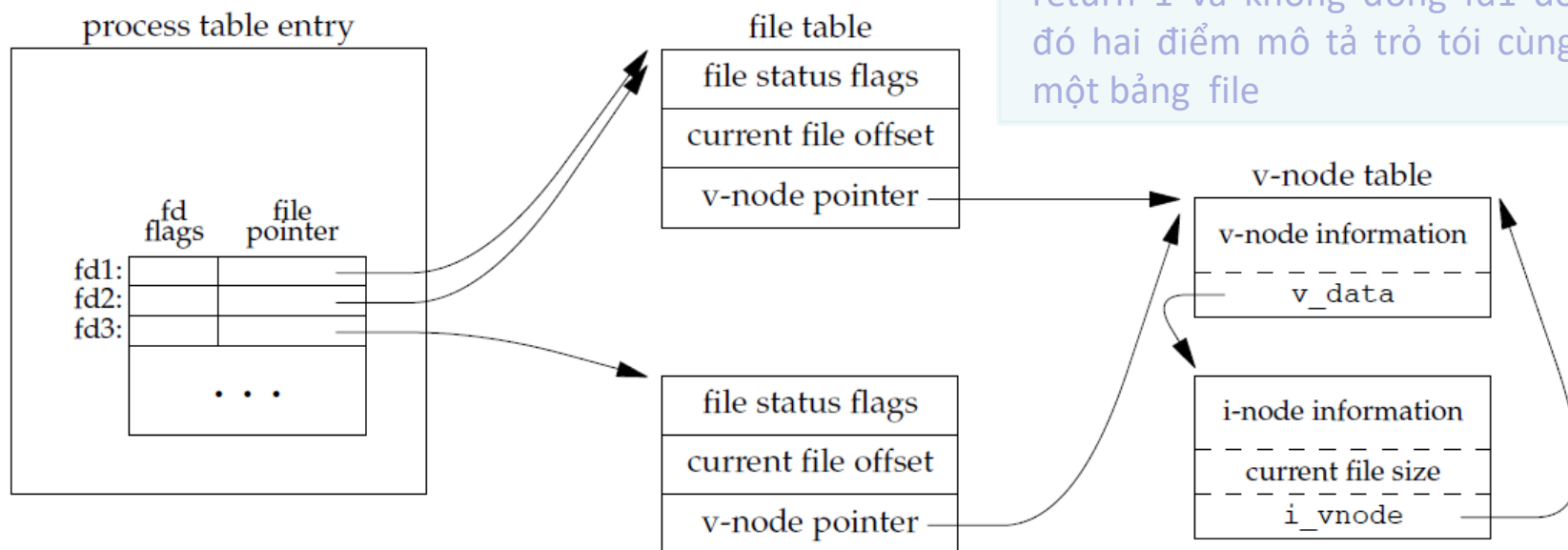
Lệnh dup2(S) cung cấp chức năng tương tự như dup(S) nhưng cho người gọi sự linh hoạt hơn với sự lựa chọn của bộ mô tả tệp mới được trả về

C Programming

❑ QUẢN LÝ FILE

- Một tiến trình có hai tệp khác nhau đang mở: một file mở trên stdin (bộ mô tả tệp 0) và tệp còn lại mở trên stdout (bộ mô tả tệp 1).
- Mỗi tệp (hoặc thiết bị) đang mở có cấu trúc v-node chứa thông tin về loại tệp và con trỏ đến các hàm hoạt động trên tệp.

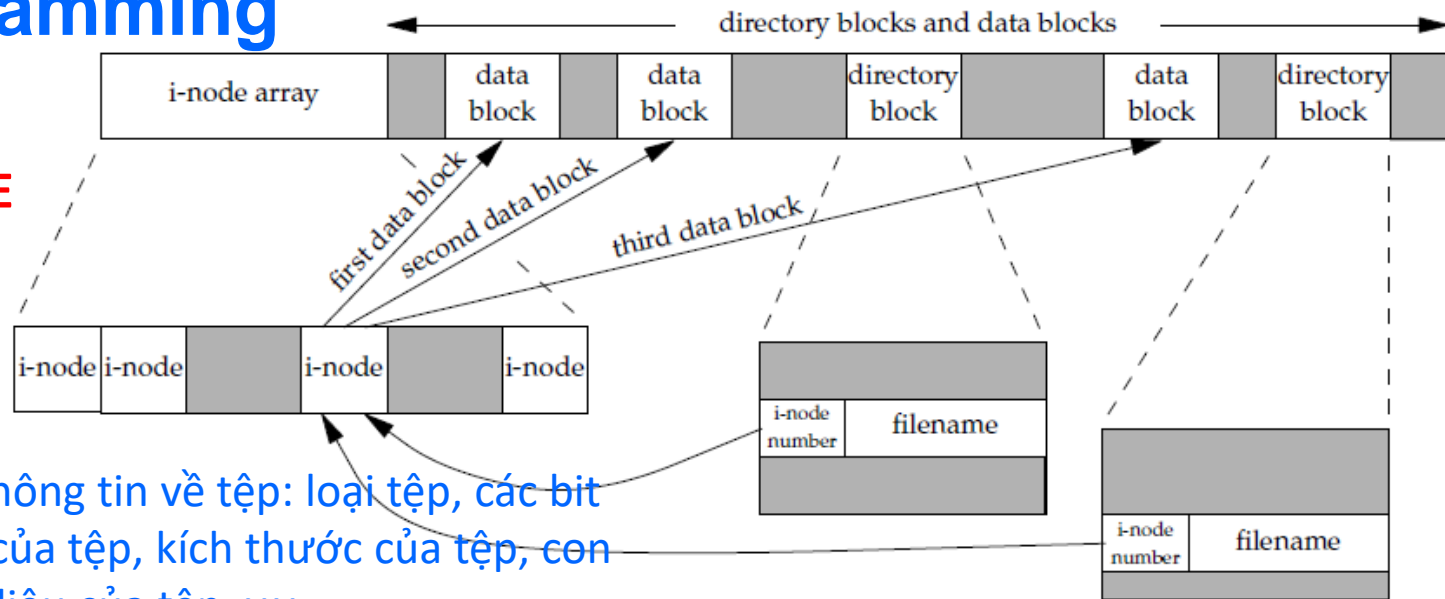
Nếu `fd = 1` thì lệnh `dup2(fd,1)` return 1 và không đóng `fd1` do đó hai điểm mô tả trỏ tới cùng một bảng file



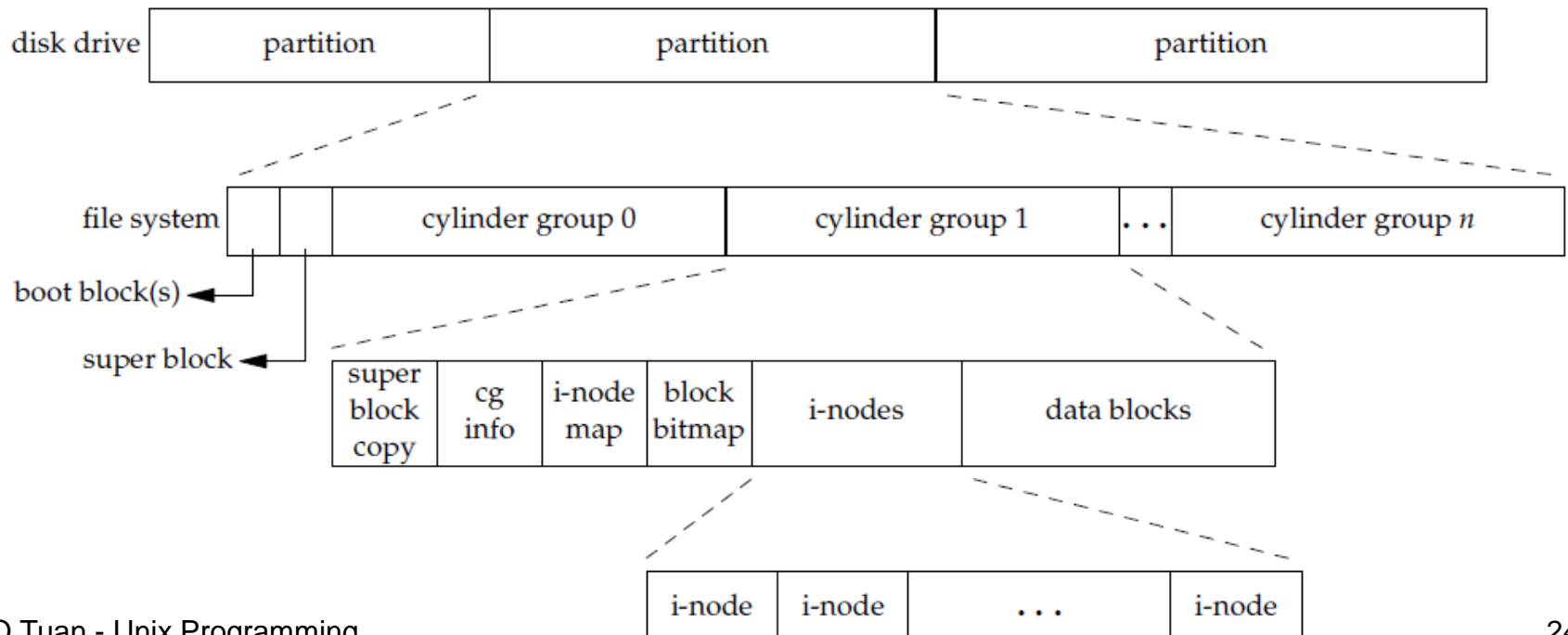
- Linux file dùng *v-node* cũng chứa i-node cho tệp. Thông tin này được đọc từ đĩa khi tệp được mở, do đó tất cả thông tin thích hợp về tệp đều có sẵn. Ví dụ: i-node chứa chủ sở, kích thước của tệp, con trỏ đến nơi đặt các khối dữ liệu thực tế tệp trên đĩa, v.v.

C Programming

❑ QUẢN LÝ FILE



i-node chứa tất cả thông tin về tệp: loại tệp, các bit cấp quyền truy cập của tệp, kích thước của tệp, còn trỏ đến các khối dữ liệu của tệp, v.v.



C Programming

- Ví dụ: # redirect.c

```
#include <fcntl.h>
```

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

```
#include <unistd.h>
```

```
#define CREATE_FLAGS (O_WRONLY | O_CREAT | O_APPEND)
```

```
#define CREATE_MODE (S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH)
```

```
int main(void) {
```

```
    int fd;
```

```
    fd = open("my.file", CREATE_FLAGS, CREATE_MODE);
```

```
    if (fd == -1) {
```

```
        perror("Failed to open my.file"); return 1; }
```

```
    if (dup2(fd, STDOUT_FILENO) == -1) {
```

```
        perror("Failed to redirect standard output"); return 1; }
```

```
    if (r_close(fd) == -1) {
```

```
        perror("Failed to close the file"); return 1; }
```

```
    if (write(STDOUT_FILENO, "OK", 2) == -1) {
```

```
        perror("Failed in writing to file"); return 1; }
```

```
    return 0;
```

```
}
```

after open

file descriptor table

[0]	standard input
[1]	standard output
[2]	standard error
[3]	write to my.file

after dup2

file descriptor table

[0]	standard input
[1]	write to my.file
[2]	standard error
[3]	write to my.file

after close

file descriptor table

[0]	standard input
[1]	write to my.file
[2]	standard error

❑ QUẢN LÝ FILE

C Programming

❑ QUẢN LÝ FILE

▪ Điều khiển file

#include <fcntl.h>

*int **fcntl**(int fd, int cmd, ... /* struct flock *flockptr */);*

Returns: depends on cmd if OK, -1 on error

```
struct flock {  
    short l_type; /* F_RDLCK, F_WRLCK, or F_UNLCK */  
    short l_whence; /* SEEK_SET, SEEK_CUR, or SEEK_END */  
    off_t l_start; /* offset in bytes, relative to l_whence */  
    off_t l_len; /* length, in bytes; 0 means lock to EOF */  
    pid_t l_pid; /* returned with F_GETLK */  
};
```

- Các *cmd* của *fildes*:

- + F_DUPFD: Duplicate a file descriptor
- + F_GETFD: Get file descriptor flag ...
- + F_SETFD: Set file descriptor flag
- + F_S/GETFL: Set/Get file status flag and access mode
- + F_GETLK: Get first lock that blocks descriptions specified by arg
- + F_SETLK: Set or clear segment lock specified by arg
- + F_SETLKW: Giống F_SETLK nhưng làm tiến trình ngủ ...

- Hàm **fcntl** có thể thay đổi đặc tính của file đã mở.
- Đối số **fildes** của **fcntl** chỉ định bộ mô tả và đối số **cmd** chỉ định hoạt động. Hàm **fcntl** có thể nhận các tham số bổ sung tùy thuộc vào giá trị của **cmd**.

C Programming

❑ QUẢN LÝ FILE

▪ Ví dụ fcntl

setblock.c

```
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>
```

```
int setblock(int fd)
{
    int fdflags;
    if ((fdflags = fcntl(fd, F_GETFL, 0)) == -1)
        return -1;
    fdflags &= ~O_NONBLOCK;
    if (fcntl(fd, F_SETFL, fdflags) == -1)
        return -1;
    return 0;
}
```

File status flag	Description
O_RDONLY	open for reading only
O_WRONLY	open for writing only
O_RDWR	open for reading and writing
O_EXEC	open for execute only
O_SEARCH	open directory for searching only
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)

Ví dụ fcntl thay đổi mode I/O được liên kết với mô tả file *fd* thành blocking bằng cách xóa cờ tập O_NONBLOCK. Để xóa cờ, sử dụng bitwise AND với complement của cờ O_NONBLOCK.

Nếu thành công, setblock trả về 0. Nếu không thành công, setblock trả về -1 và đặt errno.

C Programming

❑ QUẢN LÝ FILE

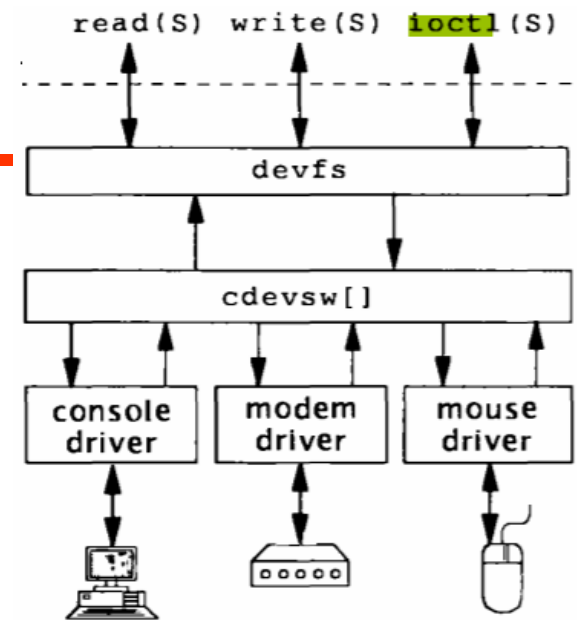
▪ Hàm ioctl file thiết bị

```
#include <unistd.h> /* System V */  
#include <sys/ioctl.h> /* BSD and Linux */  
int ioctl(int fd, int request, ...);
```

Returns: -1 on error, something else if OK

- Hàm ioctl cung cấp phương tiện thu thập thông tin trạng thái thiết bị hoặc thiết lập các tùy chọn điều khiển thiết bị. ioctl có cú pháp thay đổi : hai tham số đầu là một bộ mô tả tệp đang mở và số nguyên xác định yêu cầu khác nhau có thể có thông số bổ sung khác nhau.
- Mỗi trình điều khiển thiết bị có thể xác định bộ lệnh ioctl của riêng nó. Tuy nhiên, hệ thống cung cấp các lệnh ioctl chung cho các lớp thiết bị khác nhau. Ví dụ danh mục cho các lệnh ioctl chung này được hỗ trợ trong FreeBSD được tóm tắt trong

Category	Constant names	Header	Number of ioctls
disk labels	DIOxxx	<sys/disklabel.h>	4
file I/O	FIOxxx	<sys/filio.h>	14
mag tape I/O	MTIOxxx	<sys/mtio.h>	11
socket I/O	SIOxxx	<sys/sockio.h>	73
terminal I/O	TIOxxx	<sys/ttycom.h>	43



C Programming

❑ QUẢN LÝ FILE

▪ Hàm ioctl

```
#include "apue.h"
#include <termios.h>
#ifdef TIOCGWINSZ
    #include <sys/ioctl.h>
#endif
static void pr_winsize(int fd)
{ struct winsize size;
  if (ioctl(fd, TIOCGWINSZ, (char *) &size) < 0)
    err_sys("TIOCGWINSZ error");
  printf("%d rows, %d columns\n", size.ws_row, size.ws_col); }
static void sig_winch(int signo)
{ printf("SIGWINCH received\n");
  pr_winsize(STDIN_FILENO); }
Int main(void)
{ if (isatty(STDIN_FILENO) == 0)
    exit(1);
  if (signal(SIGWINCH, sig_winch) == SIG_ERR)
    err_sys("signal error");
  pr_winsize(STDIN_FILENO);          /* print initial size */
  for ( ; ; )                       /* and sleep forever */
    pause(); }
```

Chương trình in kích thước cửa sổ màn hình hiện tại và chuyển sang chế độ ngủ. Mỗi khi kích thước cửa sổ thay đổi, SIGWINCH sẽ bị bắt và kích thước mới được in. Chương trình chỉ chấm dứt bằng một báo hiệu (signo).

```
struct winsize
{
  unsigned short ws_row; /* rows */
  unsigned short ws_col; /* columns */
  unsigned short ws_xpixel; /* Hsize */
  unsigned short ws_ypixel; /* Vsize */
};
```

C Programming

❑ QUẢN LÝ FILE

▪ Đường ống

- Cơ chế giao tiếp liên tiến trình UNIX là đường ống, được biểu diễn bằng một tệp đặc biệt. Hàm pipe tạo ra một bộ đệm giao tiếp mà dữ liệu được ghi vào `fildes[1]` có thể được đọc từ `fildes[0]` trên cơ sở nhập trước xuất trước.

- Khuôn dạng

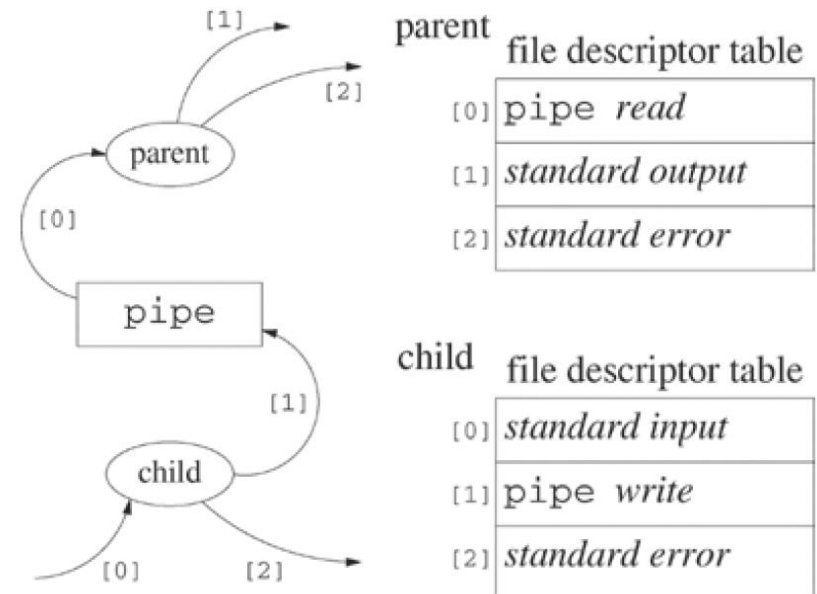
```
#include <unistd.h>
int pipe(int fildes[2]);
```

- Nếu thành công, đường ống trả về 0. Nếu không thành công, đường ống trả về -1 và đặt `errno`.

- Một số lỗi

Sai nguyên nhân

EMFILE Nhiều hơn `MAX_OPEN-2` bộ mô tả tệp đã sử dụng bởi tiến trình
ENFILE số lượng tệp mở đồng thời trong hệ thống sẽ vượt quá giới hạn



C Programming

❑ QUẢN LÝ FILE

▪ Đường ống

- Ví dụ:

```
#include <unistd.h>
#include <string.h>
#define BUFSIZ 256

char *message = "data to be sent through the pipe";
char buf[BUFSIZ];

main()
{  int fd[2], nread;
   if (pipe(fd) == -1) pexit("pipe creation:");
   if (write(fd[1], message, strlen(message)) < 0)
       pexit("pipe write:");
   nread = read(fd[0], buf, BUFSIZ);
   printf("nread=%d, data read= [%s]\n", nread, buf);
}
```

Chú ý: Cơ chế đường ống cho phép dữ liệu ghi vào buffer đường ống

C Programming

❑ QUẢN LÝ FILE

▪ File link

- Thư mục UNIX có hai loại liên kết – Một liên kết gọi liên kết cứng - mục nhập thư mục. Một liên kết tượng trưng gọi là liên kết mềm, là một file sử dụng để sửa đổi tên đường dẫn khi nó gặp phải trong quá trình phân giải tên đường dẫn.
- Khi một tệp bị xóa, số lượng liên kết sẽ giảm. Nếu số liên kết bằng zero file sẽ bị xóa. Loại liên kết này được là liên kết cứng (trái ngược với liên kết biểu tượng).
- Các file liên kết cứng có cùng số inode. Ví dụ /home/log and /usr/log có hai đường dẫn khác nhau nhưng có thể truy cập cùng 1 file vật lý do đó có liên kết file bằng 2.

Ví dụ: *\$ touch file1*

```
$ ls -li file1
```

```
98 -rw-r--r-- 1 qtuan qtuan 0 Oct 2 10:12 file1
```

```
$ ln file1 file2
```

```
98 -rw-r--r-- 2 qtuan qtuan 0 Oct 2 10:12 file1
```

```
98 -rw-r--r-- 2 qtuan qtuan 0 Oct 2 10:12 file2
```

```
$ rm file1
```

```
$ ls -li file2
```

```
98 -rw-r--r-- 1 qtuan qtuan 0 Oct 2 10:12 file2
```


C Programming

❑ QUẢN LÝ FILE

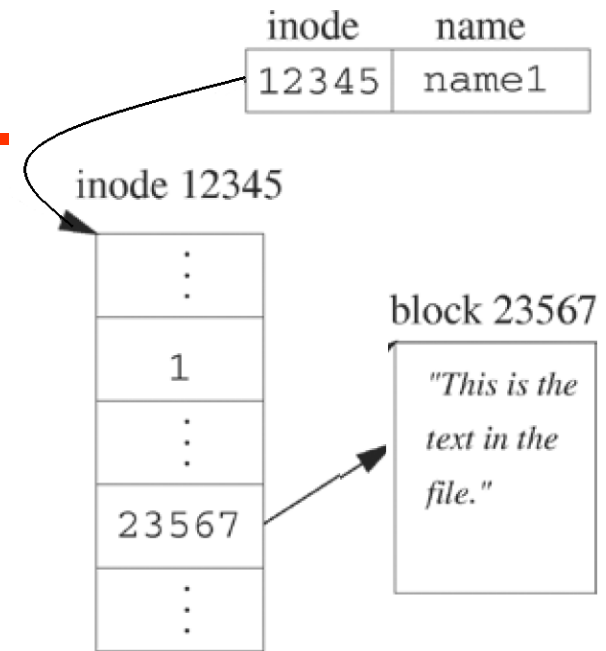
▪ Hard link

- Liên kết cứng được tạo

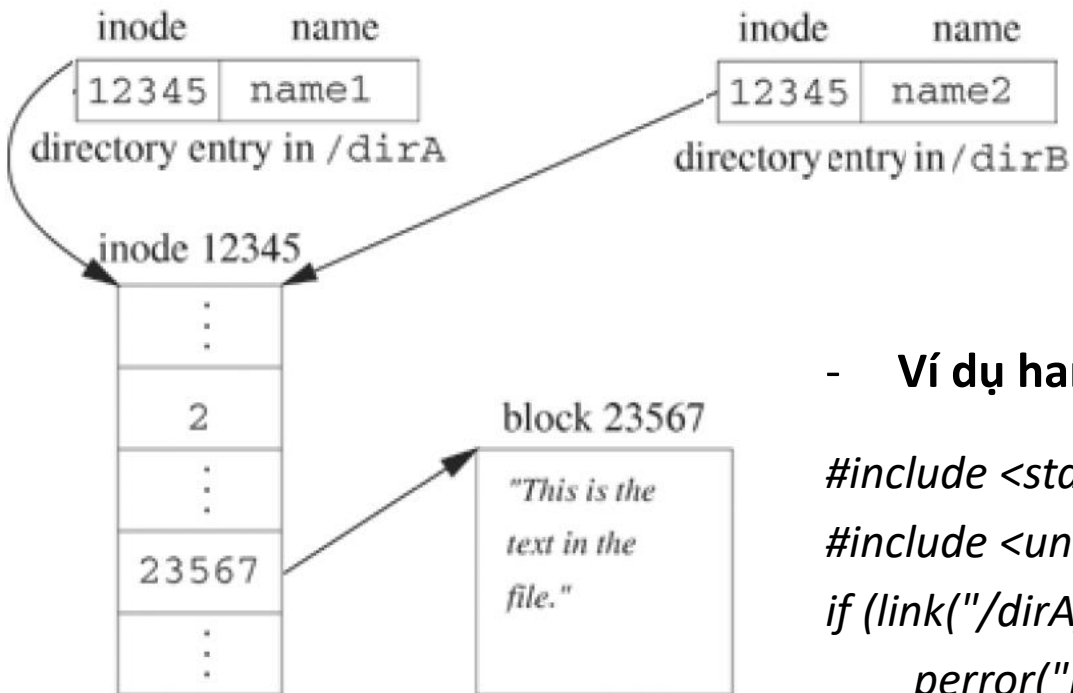
```
#include <unistd.h>
```

```
int link (char *path, char *newpath) ;
```

- Liên kết mới sẽ phân bổ mục nhập thư mục mới và tăng số liên kết của inode tương ứng, không sử dụng thêm dung lượng đĩa khác.
- Khi xóa file hoặc gọi hàm hủy liên kết khỏi một chương trình, HDH sẽ xóa mục nhập thư mục tương ứng và giảm số liên kết trong inode. Nó không giải phóng inode và các khối dữ liệu tương ứng trừ khi số lượng liên kết giảm xuống 0.



❑ QUẢN LÝ FILE



- Ví dụ hard link

```
#include <stdio.h>
#include <unistd.h>
if (link("/dirA/name1", "/dirB/name2") == -1)
    perror("Failed to make a new link in /dirB");
```

- Lệnh **ln** tạo một liên kết (mục nhập thư mục) tham chiếu đến cùng một inode như `dirA / name1`. Không cần thêm dung lượng đĩa, ngoại trừ có thể nếu mục nhập thư mục mới làm tăng số lượng khối dữ liệu cần thiết để lưu giữ thông tin thư mục. Inode bây giờ có hai liên kết.
- Hàm **unlink** loại bỏ mục nhập thư mục được chỉ định bởi **ln**. Nếu số liên kết của tệp là 0 và không có tiến trình nào mở tệp, **unlink** sẽ giải phóng không gian bị chiếm bởi tệp tin.

```
#include <unistd.h>
int unlink(const char *path);
```

C Programming

Liên kết tượng trưng

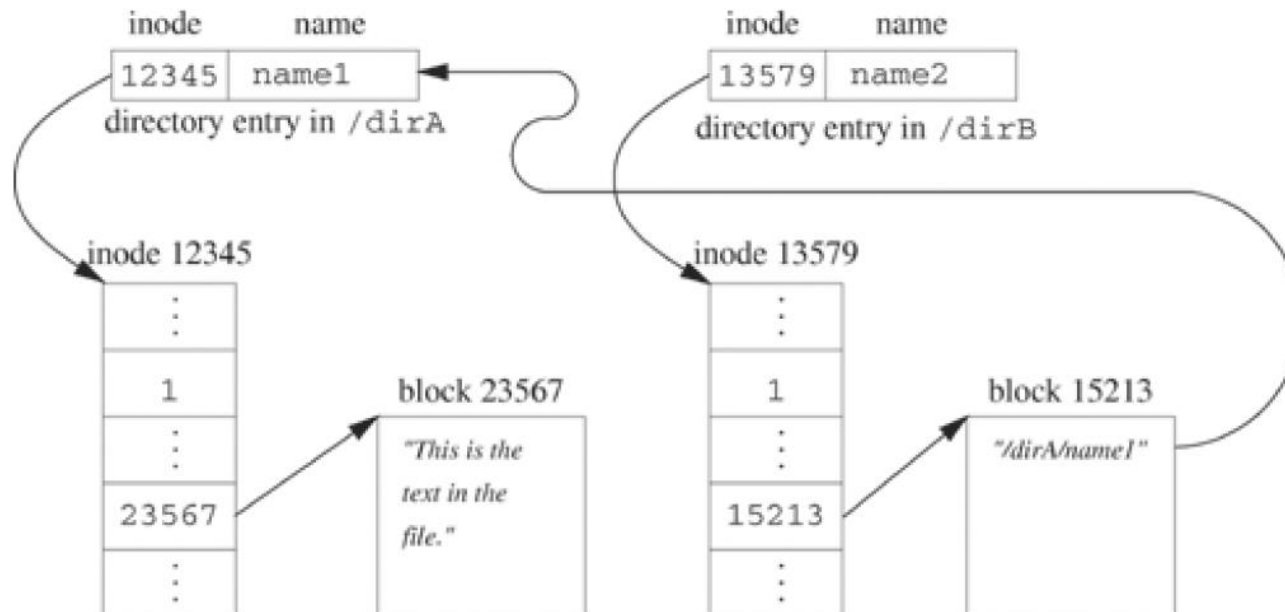
❑ QUẢN LÝ FILE

▪ Soft link

- **Symlink** là file chứa tên của *file* hoặc thư mục khác. tham chiếu đến tên symlink khiến hệ điều hành xác định vị trí inode tương ứng với liên kết đó. Hệ điều hành giả định rằng các khối dữ liệu của inode tương ứng chứa một tên đường dẫn khác.

- Shell

`$ ln -s /dirA/name1 /dirB/name2`



C Programming

- Liên kết tượng trưng

❑ QUẢN LÝ FILE

▪ Soft link

- Đối path1 chỉ định vị trí của file liên kết tượng trưng được chỉ định bởi đối path2 sẽ trở đến. Xét

```
path1 = /home/qtuan/bin/myshell
path2 = /home/SV-18001915/shlink
```

- Nếu: `$ ls -l /home/SV-18001915/shlink`
lrwxrwxrwx 1 SV-18001915 group 23 Jun 10 19:37 /home/SV-18001915/shlink -> /home/qtuan/bin/myshell
- `$ cat flk.c`

```
#define BUFSIZ 256
main()
{ char buf[BUFSIZ];
  int nread;
  nread = readlink("/home/SV-18001915/shlink", buf, BUFSIZ);
  if (nread < 0) pexit("readlink");
  buf[nread] = '\0';
  printf("nread %d\n", nread);
  printf("symlink = [%s]\n", buf);
```

```
$ a.out
nread 23
symlink = [/home/qtuan/bin/myshell]
```

}

C Programming

❑ QUẢN LÝ FILE

▪ Siêu dữ liệu file

Xét file mstat.c

```
#include <sys/types.h>
#include <stat.h>
#include <sys/sysmacros.h>
main(int argc, char* argv[])
{
    struct stat s;
    if (argc < 2)
    {
        printf("Usage: %s filename\n", argv[0]);
        exit (-1);
    }
    if (stat (argv[1], &s) != 0)
    {
        perror("mstat");
        exit (-1);
    }

    printf("st_dev = (%d, %d) \n", major (s.st_dev),
        minor(s.st_dev));
    printf("st_ino = %d\n", s.st_ino);
    printf("st_mode = %0\n", s.st_mode);
    printf("st_nlink = %d\n", s.st_nlink);
    printf("st_uid = %d\n", s.st_uid);
    printf("st_gid = %d\n", s.st_gid);
    printf("st_rdev = (%d, %d) \n", major (s.st_rdev),
        minor(s.st_rdev));
    printf("st_size = %d \n", s.st_size);
    printf("st_atime=%s", ctime(&s.st_atime));
    printf("st_mtime=%s", ctime(&s.st_mtime));
    printf("st_ctime = %s", ctime(&s.st_ctime));
}
```

- Các file chứa thông tin gọi là siêu dữ liệu file mô tả các thuộc tính của file như: chủ sở hữu, quyền truy cập, loại và kích thước của file và nơi dữ liệu được lưu trữ trong hệ thống file

C Programming

❑ QUẢN LÝ FILE

- **Siêu dữ liệu file**
- Ta kiểm tra chính nó

```
$ mstat mstat
      st_dev      = (1, 42)
      St_ino       = 22087
      St_mode      = 100755
      St_nlink     = 1
      St_uid       = 13583
      St_gid       = 50
      st_rdev      = (0, 0)
      St_size      = `101148
      St_atime     = Sat Oct
      St_mtime     = Sat Oct
      St_ctime     = Sat Oct
```

Sử dụng lệnh:
\$ **ls -li mstat**

```
22087 -rwxr-xr-x 1 qtuan qtuan 101148 Oct 4 10:12 mstat
```

Thanks