SE03 – Nhập xuất file với C trên Linux

A. Khái niệm

1. glibc là gì?
    - Dự án Thư viện GNU C cung cấp các thư viện lõi cho hệ thống GNU và các hệ thống GNU/Linux, cũng như nhiều hệ thống khác sử dụng Linux kernal làm hạt nhân.
    - Các thư viện này cung cấp các API quan trọng bao gồm ISO C11, POSIX.1-2008, BSD, API dành riêng cho hệ điều hành, v.v.
    - Thư viện GNU C được thiết kế để trở thành thư viện ISO C tương thích ngược, di động và hiệu năng cao. Mục tiêu của nó là tuân theo tất cả các tiêu chuẩn liên quan bao gồm ISO C11, POSIX.1-2008 và IEEE 754-2008.

2. Streams Input/Output
    - Luồng (stream) là môi trường trung gian để gửi hoặc nhận dữ liệu giữa chương trình và các thiết bị nhập xuất.
    - Trong C, cấu trúc dữ liệu đại diện cho luồng được gọi là FILE. Vì hầu hết các hàm thư viện xử lý các đối tượng kiểu FILE *, nên đôi khi thuật ngữ con trỏ tệp cũng được dùng với nghĩa là luồng.
    - Kiểu dữ liệu FILE:
        - Được khai báo trong file header stdio.h. File là kiểu đối tượng hỗ trợ thao tác với dữ liệu được lưu trữ bên trong File. Nó xác định một luồng và chứa các thông tin cần thiết để điều khiển, bao gồm một con trỏ trỏ đến buffer, các chỉ mục và trạng thái của nó.
    - Các chương trình C chỉ nên xử lý các con trỏ tới các đối tượng này (nghĩa là các giá trị FILE*) chứ không phải chính các đối tượng đó.
    - Luồng tiêu chuẩn - Khi hàm main() của được gọi, sẽ có 3 luồng được mặc định mở và sẵn sàng để sử dụng. Chúng đại diện cho các kênh đầu vào và đầu ra "tiêu chuẩn" đã được thiết lập cho các tiến trình. Các luồng này được khai báo trong file header stdio.h:
        - FILE* stdin: Con trỏ trỏ đến luồng Standard input, cấp quyền truy cập vào bàn phím, là input bình thường của chương trình.
        - FILE* stdout: Con trỏ đến luồng Standard output, cấp quyền truy cập vào màn hình, được dùng làm output bình thường của chương trình.
        - FILE* stderr: Con trỏ đến luồng Standard error, cấp quyền truy cập vào màn hình, được sử dụng cho các thông báo lỗi và chẩn đoán của chương trình.

3. Một số hàm tương tác với FILE * (luồng trong C) được glibc hỗ trợ
    - Hàm: FILE *fopen(const char *filename, const char *mode)
        - Dùng để mở một tệp dưới chế độ được chỉ định trước.
        - Tham số truyền vào:

filename: Con trỏ đến chuỗi chứa tên tệp sẽ được mở.
mode: Con trỏ đến chuỗi chỉ định chế độ mà tệp được mở.

- Các chế độ mở tệp:

Các chế độ bắt buộc

r	Mở tệp để đọc. Tệp phải tồn tại để mở.
w	Tạo một tệp mới để ghi. Nếu "filename" đã tồn tại, nội dung tệp sẽ bị xóa và tệp được xem như một tệp mới.
a	Mở một file để ghi vào cuối file. Tạo một tệp mới nếu tệp không tồn tại.
r+	Mở một tệp để đọc và ghi, tệp phải tồn tại để mở.
w+	Tạo một tệp mới để đọc và ghi. Nếu "filename" đã tồn tại, nội dung tệp sẽ bị xóa và tệp được xem như một tệp mới.
a+	Mở một file để đọc và ghi vào cuối file. Tạo một tệp mới nếu tệp không tồn tại.

- Giá trị trả về:

Nếu thành công, trả về một con trỏ đến đối tượng FILE điều khiển luồng tệp đã mở. Trả về một con trỏ null nếu thất bại.

- Ví dụ mở tệp file.txt dưới chế độ đọc và lưu kết quả vào con trỏ fp:

```
FILE * fp = fopen("file.txt", "r");
```

- Hàm: long int ftell(FILE *stream)
  - Dùng để xác định vị trí hiện tại trên file của luồng tương ứng.
  - Tham số truyền vào:

stream: Con trỏ đến đối tượng FILE xác định luồng.

- Giá trị trả về:

Trả về vị trí hiện tại của luồng trên file nếu thành công. Trả về -1 nếu thất bại.

- Hàm: int fseek(FILE *stream, long int offset, int whence)
  - Dùng để xác định vị trí bắt đầu của luồng trong file được chỉ định bởi offset.
  - Tham số truyền vào:
    - stream: Con trỏ đến đối tượng FILE xác định luồng.
    - offset: Số lượng byte tính từ whence.
    - whence: Chỉ định vị trí bắt đầu tính offset.

Các hằng số whence có thể chỉ định

SEEK_SET	Chỉ định whence ở bắt file
SEEK_CUR	Chỉ định whence ở vị trí hiện tại của luồng tương ứng với file
SEEK_END	Chỉ định whence ở cuối file

- Ví dụ lấy độ dài nội dung của file file.txt thông qua luồng được trỏ tới bởi con trỏ fp:

```
FILE *fp = fopen("file.txt", "r");
fseekg (fp, 0, SEEK_END);
length = ftell(fp);
```

- Hàm: int fclose (FILE *stream)
  - Dùng để đóng luồng được chỉ định và ngắt kết nối đến tệp tương ứng.
  - Tham số truyền vào:

stream: con trỏ đến một đối tượng FILE chỉ định luồng sẽ được đóng.

- Giá trị trả về: Trả về 0 nếu thành công, EOF nếu thất bại.
- Ví dụ đóng luồng được trỏ tới bởi con trỏ fp:

```
fclose(fp);
```

- Hàm:

size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream)

- Dùng để ghi dữ liệu vào luồng.

- Tham số truyền vào:

ptr: con trỏ trỏ đến mảng chứa các dữ liệu sẽ ghi.
size: độ dài của mỗi phần tử trong mảng dữ liệu sẽ ghi.
nmemb: số lượng phần tử của mảng dữ liệu sẽ ghi.
stream: con trỏ đến một đối tượng FILE mà dữ liệu được ghi vào.

- Giá trị trả về:

Số lượng phần tử được ghi thành công dưới kiểu dữ liệu size_t, nếu kết quả này khác với tham số nmemb tức có lỗi xảy ra trong quá trình ghi.

- Ví dụ ghi dữ liệu được chứa trong mảng ký tự str vào tệp file.txt thông qua luồng được trỏ bởi con trỏ fp:

```
FILE* fp = fopen(“file.txt”, “w”);
char str[] = "Hello, nice to meet you.";
fwrite(str, 1, sizeof(str), fp);
fclose(fp);
```

- Hàm: size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream)
- Dùng để đọc dữ liệu từ luồng stream.
- Tham số truyền vào:

ptr: con trỏ trỏ đến mảng chứa các dữ liệu đọc từ file với kích thước tối thiểu là size*nmemb (byte).
size: độ dài của mỗi phần tử trong mảng dữ liệu sẽ ghi.
nmemb: số lượng phần tử của mảng dữ liệu sẽ ghi.
stream: con trỏ đến một đối tượng FILE mà dữ liệu được đọc.

- Giá trị trả về:

Số lượng phần tử được đọc thành công dưới kiểu dữ liệu size_t, nếu kết quả này khác với tham số nmemb tức có lỗi xảy ra trong quá trình đọc.

- Ví dụ đọc dữ liệu trong tệp file.txt thông qua luồng được trỏ tới bởi con trỏ fp vào mảng ký tự buffer:

```
FILE *fp = fopen("file.txt", "r");
char buffer[100];
fseekg (fp, 0, SEEK_END);
length = ftell(fp);
fseek(fp, 0, SEEK_SET);
fread(buffer, 1, length+1, fp);
fclose(fp);
```

- Hàm: int fprintf(FILE *stream, const char *format, …)
  - Dùng để ghi dữ liệu đã được format từ luồng stream.
  - Tham số truyền vào:

stream: con trỏ đến đối tượng FILE* chỉ định luồng sẽ được ghi dữ liệu vào.
format: chuỗi C chứa văn bản được ghi vào luồng. Nó có thể tùy chọn chứa các thẻ định dạng nhúng được thay thế bằng các giá trị được chỉ định trong các đối số bổ sung tiếp theo và được định dạng theo yêu cầu.

  - Cấu trúc thẻ format:

%[flags][width][.precision][length]specifier

  - trong đó:

specifier	
c	Ký tự
d hoặc i	Số nguyên thập phân có dấu
e	Ký hiệu khoa học (phần định trị/số mũ) sử dụng ký tự e
E	Ký hiệu khoa học (phần định trị/số mũ) sử dụng ký tự E
f	Số thập phân có dấu chấm động (số thực)
g	Tương tự %e hoặc %f
G	Tương tự %E hoặc %f
o	Số hệ bát phân có dấu
s	Chuỗi
u	Số nguyên thập phân không dấu
x	Số nguyên hệ thập lục phân không dấu
X	Số nguyên hệ thập lục phân không dấu (các chữ cái được in hoa)
p	Địa chỉ con trỏ
n	Không in
%	Ký tự
Flags	
-	Căn trái
+	Bắt buộc phải thêm dấu cộng hoặc dấu trừ (+ hoặc -) trước kết quả ngay cả đối với các số dương.
(space)	Chèn khoảng trắng vào trước giá trị nếu không đặt cờ
#	Được sử dụng với các specifier o, x hoặc X - giá trị được bắt đầu bằng 0, 0x hoặc 0X tương ứng cho các giá trị khác 0. Được sử dụng với e, E và f - buộc đầu ra được viết phải chứa dấu thập phân ngay cả khi không có chữ số nào theo sau. Được sử dụng với g hoặc G - kết quả giống như với e hoặc E nhưng các số 0 ở cuối không bị xóa.
0	Đệm bên trái số bằng số 0.
Width	
(number)	Số lượng ký tự tối thiểu được in. Nếu giá trị được in ngắn hơn số này, kết quả sẽ được đệm bằng khoảng trống. Giá trị không bị cắt bớt ngay cả khi kết quả lớn hơn.
*	Chiều rộng không được chỉ định trong chuỗi định dạng, mà là một tham số nguyên trước đối số phải được định dạng.
Length	

h	Đối số được hiểu là kiểu short int hoặc unsigned short int (áp dụng cho specifier chỉ số nguyên: i, d, o, u, x, X)
l	Đối số được hiểu là kiểu long int hoặc unsigned long int (áp dụng cho specifier chỉ số nguyên: i, d, o, u, x, X) và dưới dạng một ký tự rộng hoặc chuỗi ký tự rộng đối với các bộ xác định c và s.
L	Đối số được hiểu là kiểu long double (áp dụng cho specifier chỉ số thực: e, E, f, g, G)

- Giá trị trả về: Số lượng ký tự ghi được khi thành công. Một số âm nếu thất bại.

- Khác nhau giữa fopen() và fprintf()

fopen()	fprint()
Ghi dữ liệu dưới dạng nhị phâncharacter	Ghi dữ liệu dưới dạng chuỗi
Ghi dữ liệu dựa trên độ dài truyền vào	Ghi đến khi gặp ký tự kết thúc chuỗi (null character)
Có thể sử dụng được với mọi loại dữ liệu	Khi ghi dữ liệu dưới dạng nhị phân, hàm sẽ ghi đến khi gặp byte 0 (dạng nhị phân của NULL character).
Có thể sử dụng khi xây dựng chương trình copy tệp.	Không nên sử dụng khi xây dựng chương trình copy tệp vì sẽ làm mất nội dung của các tệp binary

4. Low-Level Input/Output
   - Các hàm này gọi trực tiếp hệ điều hành cho hoạt động cấp thấp hơn so với hoạt động được cung cấp bởi Streams Input/Output. Quá trình này không đệm hoặc định dạng dữ liệu (Việc này phải được thực hiện thủ công bởi lập trình viên), các khối dữ liệu được chuyển một cách ẩn danh bằng cách sử dụng các khả năng cơ bản của hệ điều hành (system call).
   - Low-Level Input/Output sử dụng một bộ điều khiển tệp hoặc bộ mô tả (một số nguyên không âm), để xác định duy nhất một tệp thay vì sử dụng một con trỏ tới cấu trúc FILE như trong trường Streams Input/Output.
   - Low-Level Input/Output được sử dụng khi:
     - Truy cập trực tiếp các tệp và thiết bị.
     - Đọc các tệp nhị phân theo khối lớn.
     - Thực hiện các thao tác I/O nhanh chóng và hiệu quả.
5. Một số hàm Low-Level Input/Output libc hỗ trợ
   - Hàm: int open (char *filename, int oflag [, int pmode]);
     - Hàm tạo và trả về một bộ mô tả tệp mới cho tệp được đặt tên theo filename.
     - Tham số truyền vào:
       - filename: con trỏ đến chuỗi chứa tên tệp được mở.
       - oflag: có thể là bất kỳ sự kết hợp logic nào của các hằng số sau

O_APPEND	Nối vào cuối file
O_BINARY	Chế độ nhị phân
O_CREAT	Tạo tệp mới nếu tệp tên “filename” không tồn tại
O_RDONLY	Chỉ cho phép đọc
O_RDWR	Cho phép đọc và ghi

O_TEXT	Chế độ văn bản
O_TRUNC	Cắt ngắn tập tin về độ dài bằng không
O_WRONLY	Chỉ cho phép ghi

- pmode có thể là một trong các giá trị sau

S_IWRITE	Cấp quyền ghi cho tệp ở mức độ hệ điều hành
S_IREAD	Cấp quyền đọc cho tệp ở mức độ hệ điều hành
S_IREAD|S_IWRITE	Cáp quyền đọc và ghi cho tệp ở mức độ hệ điều hành

- Giá trị trả về:

Một bộ mô tả tệp mới cho tệp được đặt tên theo filename nếu thành công hoặc -1 nếu thất bại

- Ví dụ mở tệp file.txt dưới chế độ nhị phân và chỉ cho phép đọc và lưu kết quả trả về vào biến handle:

```
int handle = open("file.txt", O_BINARY|O_READ);
```

- Hàm: int fileno(FILE *stream);
  - Dùng để trả về bộ mô tả tệp được dùng để khởi tạo luồng stream. Bộ mô tả tệp này vẫn thuộc sở hữu của stream và sẽ được giải phóng khi luồng được giải phóng.
  - Tham số truyền vào:

stream: Luồng cần lấy bộ mộ tệp.

- Giá trị trả về: Trả về bộ mô tả tệp tương ứng với luồng stream nếu thành công, ngược lại hàm trả về -1.

- Hàm: ssize_t read (int filedes, void *buffer, size_t size)
  - Đọc tối đa size byte từ tệp với bộ mô tả tệp filedes vào buffer
  - Tham số truyền vào:

filedes: bộ mô tả tệp của tệp được đọc.
buffer: con trỏ đến chuỗi lưu dữ liệu đọc được.
size: số byte tối đa đọc từ tệp.

- Giá trị trả về:

Số lượng byte đọc thành công (có thể nhỏ hơn size), 0 nếu file không còn dữ liệu để đọc. Trả về -1 nếu thất bại.

- Ví dụ đọc 100 bytes đầu tiên từ tệp có bộ mô tả tệp được lưu tại biến handle vào mảng ký tự buffer:

```
char buffer[200];
read(handle, buffer, 100);
```

- Hàm: ssize_t write (int filedes, const void *buffer, size_t size)
  - Ghi tối đa size byte từ buffer vào tệp có bộ mô tả tệp filedes.
  - Tham số truyền vào:

filedes: bộ mô tả tệp của tệp sẽ ghi dữ liệu vào.
buffer: con trỏ đến chuỗi lưu dữ liệu sẽ được ghi.
size: số byte tối đa ghi vào tệp.

- Giá trị trả về:

Số lượng byte ghi thành công.Trả về -1 nếu thất bại.

- Ví dụ ghi 10 phần tử đầu tiên lưu tại mảng ký tự str vào tệp có bộ mô tả tệp được lưu tại biến handle

```
char str[] = "Hello, nice to meet you.";
write(handle, str, 10);
```

- Hàm: off_t lseek (int filedes, off_t offset, int whence)
  - Dùng để thay đổi vị trí con trỏ trên tệp có bộ mộ tả tệp filedes
  - Tham số truyền vào:

filedes: Bộ mô tả tệp của tệp chỉ định.
offset: Số lượng byte đến offset từ whence.
whence: Chỉ định vị trí bắt đầu tính offset.

  - Các hằng số của whence tương tự với của hàm fseek trong Stream I/O.
  - Giá trị trả về:

Vị trí con trỏ trên file sau khi hàm kết thúc tính bằng byte từ đầu file.

- Hàm: int close (int filedes);
  - Dùng để đóng file được chỉ định, bộ mô tả tệp của file sẽ được giải phóng.
  - Tham số truyền vào:

filedes: bộ mô tả của tệp sẽ được đóng, giá trị lưu tại handle sẽ được giải phóng.

  - Giá trị trả về:

0 nếu thành công và -1 nếu thất bại.

- Ví dụ đóng file có bộ mô tả tệp được lưu trữ tại biến handle:

```
close(handle);
```

6. File System Interface
   - Ngoài Stream và Low-Level I/O, thư viện glibc còn hỗ trợ các hàm thực hiện các chức năng liên quan đến việc vận hành trên chính các tệp hơn là trên nội dung của tệp.

   - Kiểu dữ liệu: DIR
     - Đại diện cho luồng thư mục (directory stream). Tương tự luồng tệp tin, người dùng chỉ nên xử lý thông qua con trỏ DIR*.

   - Hàm: DIR * opendir(const char *dirname);
     - Mở và trả về luồng thư mục cho thư mục có tên dirname.
     - Tham số truyền vào:

dirname: tên thư mục được mở.

     - Giá trị trả về: con trỏ đến luồng thư mục tương ứng với thư mục vừa được mở nếu thành công, một con trỏ nell nếu thất bại.

   - Hàm: DIR * fdopendir (int fd)
     - Mở và trả về luồng thư mục cho thư mục có bộ mô tả tệp fd.
     - Tham số truyền vào:

fd: số nguyên dương đại diện cho bộ mô tả tệp của thư mục cần mở.

- Giá trị trả về: con trỏ đến luồng thư mục tương ứng với thư mục vừa được mở nếu thành công, ngược lại trả về con trỏ null.

- Hàm: int dirfd (DIR *dirstream)
  - Trả về bộ mô tả tệp của thư mục tương ứng với luồng thư mục dirstream trỏ tới.
  - Tham số truyền vào:

dirstream: con trỏ tới luồng thư mục tương ứng với thư mục cần lấy bộ mô tả tệp

  - Giá trị trả về: Một số nguyên dương đại diên cho bộ mô tả tệp của thư mục nếu thành công, ngược lại trả về -1.

- Kiểu dữ kiệu: struct dirent
  - Được dùng khi các hàm trả về thông tin về directory entries
  - Các trường dữ liệu:
    - char d_name[]: Tên tệp
    - ino_t d_fileno: Số serial của tệp
    - unsigned char d_namelen: Độ dài của tên tệp
    - unsigned char d_type: Loại tệp, lưu một trong các hằng sau:

DT_UNKNOWN: Loại tệp không xác định được bởi hệ thống tập tin.
DT_REG: tệp bình thường.
DT_DIR: thư mục.
DT_FIFO: pipe được đặt tên hoặc tệp FIFO.
DT_SOCK: local_domain socket.
DT_CHR: Character Device.
DT_BLK: Block Device.
DT_LNK: liên kết tượng trưng

- Hàm: struct dirent * readdir (DIR *dirstream)
  - Được dùng để đọc entry tiếp theo trong thư mục.
  - Tham số truyền vào:

dirstream: con trỏ tới luồng thư mục tương ứng với thư mục cần đọc.

  - Giá trị trả về:

Nếu thành công, hàm trả về một con trỏ đến dữ liệu kiểu dirent chứa thông tin về entry tiếp theo bên trong thư mục. Nếu entry tiếp theo trong thư mục không tồn tại hoặc có lỗi xảy ra, hàm trả về một con trỏ NULL.

- Hàm: int closedir (DIR *dirstream)
  - Dùng để giải phóng luồng thư mục.
  - Tham số truyền vào:

dirstream: con trỏ đến luồng thư mục cần đóng.

  - Giá trị trả về: Nếu đóng luồng thành công, hàm trả về 0, ngược lại trả về -1.

- Ví dụ chương trình đơn giản in tên của tệp trong thư mục làm việc hiện tại ra command line.

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

```
#include <dirent.h>
int main (void)
{
        DIR *dp = opendir ("./");
        struct dirent *ep;

        if (dp != NULL) {
        while (ep = readdir (dp))
                puts (ep->d_name);
        closedir (dp);
        }

return 0;
}
```

- Kiểu dữ liệu: struct stat
  - Được dùng khi các hàm trả về thông tin thuộc tính của tệp.
  - Các trường dữ liệu:

mode_t sr_mode	Chế độ của tệp, bao gồm các bit chỉ loại tệp và quyền truy cập cập của tệp.
ino_t st_ino	Số seri của tệp, phân biệt với các tệp khác trên cùng hệ thống tập tin.
dev_t st_dev	Xác định thiết bị chứa tệp.
nlink_t st_nlink	Số lượng hard link đến tệp.Nếu trường này bằng 0, tệp sẽ bị loại bỏ trong trường hợp không còn tiến tình nào giữ nó được mở.
uid_t st_uid	ID của người dùng sở hữu tệp
gid_t st_gid	Group ID của tệp
of_t st_size	Kích thước của tệp tính bằng byte. Với symbolic link, trường này chứa kích thước tệp được liên kếts tới.
time_t st_atime	Thời điểm cuối cùng tệp được truy cập.
unsigned long int st_atime_usec	Phần phân đoạn của thời điểm cuối cùng tệp được truy cập.
time_t st_mtime	Thời điểm cuối cùng nội dung tệp được chỉnh sửa.
unsigned long int st_mtime_usec	Phần phân đoạn của thời điểm cuối cùng nội dung tệp được chỉnh sửa.
time_t st_ctime	Thời điểm cuối cùng thuộc tính của tệp được chỉnh sửa.
unsigned long int st_ctime_usec	Phần phân đoạn của thời điểm cuối cùng thuộc tính của tệp được chỉnh sửa.
blkcnt_t st_blocks	Số khối trên đĩa mà tệp chiếm dụng, mỗi khối có kích thước 512 byte
unsigned int st_blksize	Kích thước tối ưu để đọc hoặc ghi tệp này tính bằng byte

- Hàm: int stat (const char *filename, struct stat *buf)
  - Dùng để lấy thông tin thuộc tính của tệp có tên filename vào buf.
  - Tham số truyền vào:
    - file name: tên tệp cần lấy thông tin thuộc tính.
    - buf: con trỏ đến kiểu dữ liệu struct stat để ghi các thông tin thuộc tính của tệp.
  - Giá trị trả về: Nếu thành công, hàm trả về 0, ngược lại trả về -1.
  - Ví dụ lấy thông tin thuộc tính tệp file.txt vào buffer:

```
struct stat *buf;
stat("file.txt", buf);
```

- Kiểu dữ liệu:

__ftw_func_t int (*) (const char *filename, const struct *stat, int type)

  - Một dạng con trỏ hàm được dùng bởi hàm ftw()
  - Tham số truyền vào

filename: con trỏ đến tên tệp ftw() đang duyệt đến.
stat: con trỏ đến kiểu dữ liệu struct stat lưu thông tin thuôc tính của tệp ftw() đang duyệt đến.
type: xác định thông tin về loại tệp đang duyệt đến. Có thể là một trong các hằng số sau.

FTW_F	tệp thường hoặc là tệp không thuộc bất cứ loại nào bên dưới(tệp đặc biệt, socket,...)
FTW_D	thư mục
FTW_NS	việc gọi hàm stat() thất bại kiến tham số stat không khả dụng
FTW_DNR	thư mục bị chặn quyền đọc
FTW_SL	symbolic link

- Hàm: int ftw (const char *filename, __ftw_func_t func , int descriptors)
  - Hàm ftw() sẽ đệ quy xuống hệ thống phân cấp thư mục bắt nguồn từ filename.
  - Tham số truyền vào:

filename: tên thư mục mà ftw() sẽ coi như thư mục gốc để bắt đầu duyệt.
func: con trỏ hàm thuộc kiểu dữ liệu __ftw_func_t được glibc hỗ trợ, func sẽ được gọi với mỗi đối tượng ftw() duyệt qua. ftw() sẽ truyền cho nó một con trỏ tới chuỗi chứa tên, một con trỏ tới cấu trúc struct stat chứa thông tin thuộc tính của đối tượng trả về bởi hàm stat() và một số nguyên xác định loại tệp của đối tượng. ftw() tiếp tục duyệt đối tượng tiếp theo khi func trả về giá trị 0.
descriptors: xác định số lượng bộ mô tả tệp có thể được dùng bởi ftw() khi đang duyệt.

  - Giá trị trả về:

Hàm trả về 0 nếu tất cả lần gọi func() đều trả về 0 và ftw() duyệt thành công.
Hàm trả về -1 nếu có một lần gọi func() không thành công.
Nếu func() trả về giá trị khác không 0, ftw() sẽ trả về giá trị này.

7. Phân quyền trong Linux

- Mỗi file hay thư mục trên Linux đều được gán bởi 3 loại chủ sở hữu là user, group, other:
  - User: theo như mặc định trên Linux thì người tạo ra file hay thư mục nào đó thì sẽ trở thành chủ sở hữu của chính nó, giống như việc một người A tạo ra một vậy B thì mặc định người A sẽ là chủ sở hữu của vật B đó.
  - Group: một nhóm có thể chứa nhiều người dùng cùng một lúc. Tất cả người dùng trong một nhóm sẽ có cùng quyền truy cập vào file hay thư mục đó.
  - Other: bất kỳ người dùng nào không thuộc vào 2 đối tượng phía trên.

- Mỗi một file hay thư mục trong Linux đều có 3 quyền đọc, ghi, thực thi
  - Đọc: Nếu là một file thì quyền này cho phép mở file đó lên và đọc. Nếu là một thư mục thì nó cho phép liệt kê danh sách file hay thư mục trong thư mục đó.
  - Ghi: Quyền ghi cho phép sửa đổi nội dung của file. Nếu là thư mục thì nó cho phép có thể thêm, xóa và đổi tên các file trong thư mục đó.
  - Thực thi: Trong Linux không thể chạy khi nó chưa được cấp quyền thực thi. Còn đối với thư mục thì không thể truy cập(cd) nếu không có quyền thực thi nó.

- Kiểm tra quyền trong Linux:
  - Sử dụng ls -l để hiển thị thông tin về quyền của các tệp trong working directory hiện tại

```
home@home-pc:~$ ls -l
total 52
drwxr-xr-x 2 home home 4096 Mar 22 14:21 Desktop
drwxr-xr-x 3 home home 4096 Apr  6 12:36 Documents
drwxr-xr-x 2 home home 4096 Mar 24 12:25 Downloads
drwxr-xr-x 2 home home 4096 Mar 22 14:21 Music
drwxrwxr-x 7 home home 4096 Apr  6 14:59 NetBeansProjects
-rw------- 1 home home 2053 Mar 27 11:58 Passwords.kdbx
drwxr-xr-x 3 home home 4096 Mar 27 08:52 Pictures
drwxrwxr-x 3 home home 4096 Mar 22 14:48 Programs
drwxr-xr-x 2 home home 4096 Mar 22 14:21 Public
drwx------ 4 home home 4096 Mar 22 16:30 snap
drwxr-xr-x 2 home home 4096 Mar 22 14:21 Templates
drwxr-xr-x 2 home home 4096 Mar 22 14:21 Videos
drwxrwxr-x 3 home home 4096 Mar 24 12:06 vmware
```

  - Thông tin về quyền áp dụng lên tệp sẽ hiển thị tại cột đầu tiên, thông tin này bao gồm các phần

offet 1		offset 2 3 4	offset 5 6 7	offset 8 9 10
Loại tệp		Quyền của user	Quyền của group	Quyền của other
Ký hiệu	Ý	Ký hiệu	Ý nghĩa	
d	Thư mục	r	Đọc	
-	File	w	Ghi	
		x	Thực thi	
		-	Không có quyền	

8. So sánh file
    - Cách 1: Xử dụng hàm băm mật mã.
        - Hàm băm mật mã là thuật toán toán học một chiều được sử dụng để ánh xạ dữ liệu có kích thước bất kỳ thành một chuỗi bit có kích thước cố định. Hai đầu vào giống nhau sẽ luôn cho kết quả khác nhau trong khi chỉ sai khác một bits đã có thể dẫn đến hai kết quả khác nhau hoàn toàn.

        - Ví dụ: sử dụng thuật toán băm an toàn sh256sum được hỗ trợ bởi linux để so sánh 2 tệp gốc và tệp được copy bởi chương trình file_io_2 (trong trường hợp sử dụng hàm fread()).

```
home@home-pc:~/NetBeansProjects/file_io_2/dist/Debug/GNU-Linux$ ls -la
total 48
drwxrwxr-x 2 home home  4096 Apr  7 21:25 .
drwxrwxr-x 3 home home  4096 Apr  7 19:29 ..
-rwxrwxr-x 1 home home 36784 Apr  7 21:19 file_io_2
-rw-rw-r-- 1 home home  1024 Apr  7 19:43 test
home@home-pc:~/NetBeansProjects/file_io_2/dist/Debug/GNU-Linux$ ./file_io_2 0 te
st testw
Read file end
home@home-pc:~/NetBeansProjects/file_io_2/dist/Debug/GNU-Linux$ ls -la
total 52
drwxrwxr-x 2 home home  4096 Apr  7 21:25 .
drwxrwxr-x 3 home home  4096 Apr  7 19:29 ..
-rwxrwxr-x 1 home home 36784 Apr  7 21:19 file_io_2
-rw-rw-r-- 1 home home  1024 Apr  7 19:43 test
-rw-rw-r-- 1 home home  1024 Apr  7 21:25 testw
home@home-pc:~/NetBeansProjects/file_io_2/dist/Debug/GNU-Linux$ sha256sum test
d1112a8dd26f4a0982bae37e0164f88941e09fa330f396f280c9f2a81b3243bc  test
home@home-pc:~/NetBeansProjects/file_io_2/dist/Debug/GNU-Linux$ sha256sum testw
d1112a8dd26f4a0982bae37e0164f88941e09fa330f396f280c9f2a81b3243bc  testw
```

    - Cách 2: Sử dụng diff và cmp được hỗ trợ bởi Linux
        - Lệnh diff trả về thông báo khác nhau nếu hai tệp vào có giá trị khác nhau trong khi cmp sẽ trả về vị trí khác nhau đầu tiên giữa hai tệp.

```
home@home-pc:~/NetBeansProjects/file_io_2/dist/Debug/GNU-Linux$ diff test file_i
o_2
Binary files test and file_io_2 differ
home@home-pc:~/NetBeansProjects/file_io_2/dist/Debug/GNU-Linux$ cmp test file_io
_2
test file_io_2 differ: byte 1, line 1
```