

PONTIFICIA UNIVERSIDAD CATÓLICA MADRE Y MAESTRA

FACULTAD DE CIENCIAS E INGENIERÍA



Sistemas Operativos II

ISC ISC365-101

L4 – Emulacion de ls.

Presentado por:

Junior Hernandez 2018-0999 10135069

Entregado a:

Juan Ramón Felipe Núñez Pérez

Fecha de realización: 30 de octubre del 2022

Fecha de entrega: 19 de septiembre del 2022

Santiago de los caballeros; República Dominicana.

Introducción

Desde que se introdujo Unix en la década de 1970, muchos sistemas operativos lo han utilizado como base. Muchos de estos sistemas operativos han fallado, mientras que otros han tenido éxito.

Linux es uno de los sistemas operativos basados en Unix más populares. Es de código abierto y se utiliza en muchas industrias de todo el mundo.

Una característica sorprendente del sistema operativo Linux es la interfaz de línea de comandos (CLI) que permite a los usuarios interactuar con su computadora desde una sola capa. El shell de Linux es un entorno REPL (leer, evaluar, imprimir, resolver) donde los usuarios pueden escribir un comando y el shell lo ejecutará y devolverá el resultado.

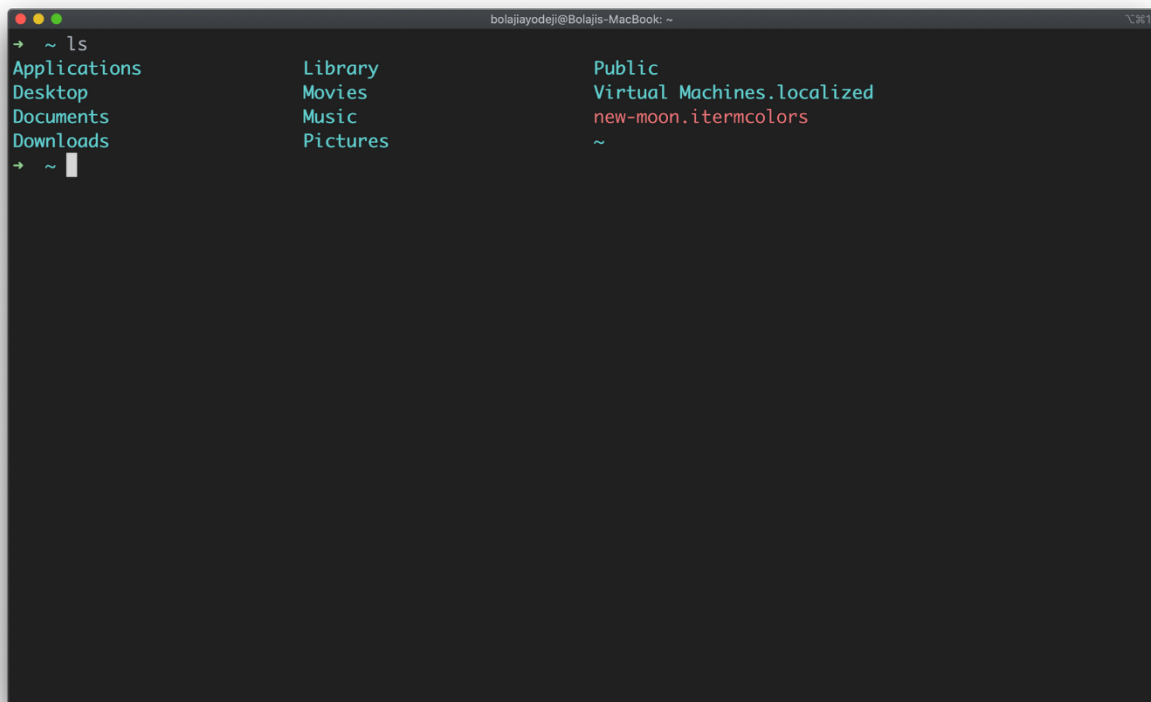
El comando `ls` es uno de los muchos comandos de Linux que un usuario puede usar para listar archivos o directorios a través de la CLI.

Marco Teórico

El comando ls se utiliza para listar archivos o directorios en Linux y otros sistemas operativos basados en Unix.

Al igual que navegas en tu explorador de archivos o finder con una interfaz gráfica de usuario, el comando ls te permite enumerar todos los archivos o directorios en el directorio actual de forma predeterminada, e interactuar con ellos a través de la línea de comandos.

Inicia tu terminal y escribe ls para ver esto en acción:

A screenshot of a macOS terminal window titled 'bolajayodeji@Bolajis-MacBook: ~'. The terminal shows the command 'ls' being executed, resulting in a three-column listing of files and directories. The first column contains 'Applications', 'Desktop', 'Documents', and 'Downloads'. The second column contains 'Library', 'Movies', 'Music', and 'Pictures'. The third column contains 'Public', 'Virtual Machines.localized', 'new-moon.itermcolors', and '~'. The prompt '~ ' is visible at the bottom left of the terminal window.

```
+ ~ ls
Applications      Library           Public
Desktop           Movies            Virtual Machines.localized
Documents         Music             new-moon.itermcolors
Downloads         Pictures          ~
+ ~
```

El comando ls también acepta algunos indicadores (también conocidos como opciones) que son información adicional que cambia la forma en que se listan los archivos o directorios en su terminal.

(Hernandez, n.d.)

Desarrollo

Emulando ls simple

Usando acceso a librerías especializadas se accederá a un directorio y de ahí a la tabla de nodos-i para obtener los atributos (stat) de los archivos en el directorio.

Como LS permite listar el contenido de un directorio, es muy útil a la hora de navegar en el sistema operativo. Con este se pueden ver los atributos que tienen los archivos y cuenta con opciones que permiten definir qué tan potente se quiere que sea la búsqueda y listado con el mismo.

Para resolver el problema se utilizan varias funciones, las cuales son:

Main: solo examina los argumentos pasados al programa para determinar la ruta y las opciones utilizadas en tiempo de ejecución.

ls: comprueba si el directorio está disponible para determinar posteriormente qué realización utilizar.

lsArgs: similar a showArch, pero dado que se usa para la opción ls -l en este caso, necesita buscar las estadísticas de cada archivo.

showArch: itera a través del conjunto de archivos, imprime el nombre de cada archivo y considera si los archivos están ocultos.

showInfo: imprime los archivos junto a los datos de stats de la función anterior.

parsePermits: Esta función se encarga de analizar y parsear los permisos del documento desde el campo st_mode de stats recibidas como parámetro.

Código:

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

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

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

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

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

```
#include <string.h>
```

```
void ls(char directorio[], char * modo) {
```

```
    printf("%s", modo);
```

```
    if (opendir(directorio) == NULL) {
```

```

    fprintf(stderr, "\nError: No se puede abrir el directorio %s\n",
        directorio);
} else {
    if (strcmp(modo, "l") == 0) {
        lsArgs(directorio, 1);
    } else if (strcmp(modo, "d") == 0) {
        printf("%s\n", directorio);
    } else if (strcmp(modo, "h") == 0) {
        showArch(directorio, 1);
    } else {
        showArch(directorio, 0);
    }
}
}
}

```

```

void showArch(char directorio[], int opt) {
    DIR * dir = opendir(directorio);
    struct dirent * dirent;
    char dirAux[20][100];
    int count = 0;
    while ((dirent = readdir(dir)) != NULL) {
        if (opt == 1) {
            printf("%llu", dirent->d_ino);
        }
        printf("%s\n", dirent->d_name);
    }
}
}

```

```

void lsArgs(char directorio[], int opt) {
    DIR * dir = opendir(directorio);
    struct dirent * dirent;
    char dirAux[20][100];
    int count = 0;
    while ((dirent = readdir(dir)) != NULL) {
        char nodo[200];
        strcpy(nodo, directorio);
        strcat(nodo, "/");
        strcat(nodo, dirent -> d_name);
        struct stat info;
        if (stat(nodo, & info) == -1) {
            perror(nodo);
        } else {
            showInfo(dirent -> d_name, & info);
        }
    }
}

void showInfo(char * archivo, struct stat * info_p) {
    char * ctime();
    char permits[11];
    parsePermits(info_p -> st_mode, permits);
    printf("%s", permits);
    printf(" %4d", (int) info_p -> st_nlink);
    printf(" %8ld", (long) info_p -> st_size);
    printf(" %.12s", 4 + ctime( & info_p -> st_mtime));
    printf(" %s\n", archivo);
}

```

```
void parsePermits(int mode, char str[]) {  
    strcpy(str, "-----");  
    if (S_ISDIR(mode))  
        str[0] = 'd';  
    if (S_ISCHR(mode))  
        str[0] = 'c';  
    if (S_ISBLK(mode))  
        str[0] = 'b';  
    if ((mode & S_IRUSR))  
        str[1] = 'r';  
    if ((mode & S_IWUSR))  
        str[2] = 'w';  
    if ((mode & S_IXUSR))  
        str[3] = 'x';  
    if ((mode & S_IRGRP))  
        str[4] = 'r';  
    if ((mode & S_IWGRP))  
        str[5] = 'w';  
    if ((mode & S_IXGRP))  
        str[6] = 'x';  
    if ((mode & S_IROTH))  
        str[7] = 'r';  
    if ((mode & S_IWOTH))  
        str[8] = 'w';  
    if ((mode & S_IXOTH))  
        str[9] = 'x';  
}
```

```

int main(int argc, char * argv[]) {
if (argc == 1) {
    ls(".", "");
} else {
    if (strcmp(argv[1], "l") == 1 || strcmp(argv[1], "d") == 1 ||
        strcmp(argv[1], "h") == 1) {
        if (argv[2] == NULL) {
            ls(".", argv[1]);
        } else {
            ls(argv[2], argv[1]);
        }
    } else {
        ls(argv[1], "");
    }
}
return 0;
}

```

Salida:

```

junior@junior-VirtualBox:~/Descargas$ ./lsdir
fork_exec.c
lsdir.c
captu.sh
execpl
lsdir
..
execpl2
fork_exec1.c
execpl3
a.out
freq.sh
fork_exec2.c
fork_exec3.c

```


Conclusión

Como se observa en el trabajo se logró hacer a través de código una emulación simple de ls para la lectura de archivo en la ubicación actual del directorio, ayudando a entender como este funciona de una manera más interna, ya que vemos como hacer una simple lectura de archivos lleva un complejo análisis para ser codificado.

References

Hernandez, R. D. (n.d.). *Freecodecamp.org*. Retrieved from Freecodecamp.org:

<https://www.freecodecamp.org/espanol/news/el-comando-linux-ls-como-listar-archivos-en-un-directorio-indicadores-de-opcion/>

<https://devdocs.io/c/>

<https://programmerclick.com/article/3464812837/>