Operating Systems

Grado en Informática. Course 2020-2021

Assigment 1: File systems. In this lab assigment we'll do two different tasks

- we'll add commands to the shell started in the previous lab assignment, so that it can view and manipulate the filesystem, listing, adding and deleting files or directories.
- we create two **separate standalone programs**. *list.c* to list files and/or directories and *delete.c* to delete files and/or directories

It is assumed that, as these two latter programs share much of their code with that of the shell, we'll use the appropiathe methods (for example, split the source code into several files that will compile separately) to get this done.

TASK 1

Add to the shell started in previous assignments the following commands

create [**-dir**] **name** Creates a file or directory in the file system. **name** is the name of the file (or directory) to be created. If **-dir** is specified a directory is to be created, otherwise an empty file will be created. If *name* is not given, the **contents of the current working directory** will be listed (nor recursively neither long listing) Example:

```
->create fich.txt
->create -dir carpeta
->create -dir /root/folder
cannot create /root/folder: permission denied
```

delete [**-rec**] **name1 name2 ...** Deletes files or directories.

- If **-rec** is given and some of the *names* stand for a directory, it will try to delete that directory and **ALL OF ITS CONTENTS**.
- If **-rec** is not given, a directory **will only be deleted if it is empty**.
- **-rec** with files has no special effect: a file will get deleted exactly

the same way with or without the *-rec* option.

- If no *name* is given, the **contents of the current working directory** will be listed (nor recursively neither long listing).
- When a file or directory cannot be removed, an appropiate message must be given to the user

list [**-long**] [**-dir**] [**-hid**] [**-rec**] **name1 name2 name3 . . .** Lists the directories and/or files with names *name1, name2 . . .*. The meaning of `-dir,-long, -dir and -rec` options is as follows

- **-long** stands for long listing: if present, items will be listed **exactly** in the format described below. Otherwise only the name and size of each file and/or directory will be listed. For directories, the directory file itself is what is going to be listed
- **-dir** This option only affects directories meaning that **IT IS THEIR CONTENTS, NOT THE DIRECTORY FILE ITSELF, WHAT IS GOING TO BE LISTED**.
- **-hid** Combined with the *-dir* option means that when listing a directory's contents, hidden files and/or directories (those whose name starts with **.**) will also get listed.
- **-rec** Combined with the *-dir* option means that when listing a directory's contents, subdirectories will be listed recursively. (if the *-hid* option is also given, hidden subdirectories will also get listed, except **.** and **..** to avoid infinite recursion).
- Subdirectories' contents will be listed **after** all the files in the directory
- if not *name* is given the current working directory will be listed
- To check what type of filesystem object a name is, one of the *stat* system calls must be used. **DO NOT USE THE FIELD `d_type` IN THE DIRECTORY ENTRY**.
- Note that the **-long** option also affects listing the contents of directories.

FORMAT FOR LONG LISTING
For a file (or directory) the format of the listing must be exactly as follows, **ONLY ONE LINE PER FILE**

```
date_last_modified inode_number owner group mode_drwx_format size (number_of_links) name
```

in the case the file is a symbolic link

```
date_last_modified inode_number owner group mode_drwx_format size (number_of_links) name -> file_the_link_points_to
```

Example:

```
-> list -long   enlace p1.c /home/antonio fjhskfhsdkf
Sep 15 13:35  15888887 antonio antonio -rw-r--r--       4 (  1)  enlace -> fich
Sep 10 11:40  15888887 antonio antonio -rw-r--r--    2713 (  1)  p1.c
Sep 20 12:13  15859713 antonio antonio  drwxrwxrwx  12288 (156)  /home/antonio/
cannot access fjhskfhsdkf: No such file or directory
```

TASK 2

- Create two standalone programs, `list.c` and `delete.c` that will do exactly the same things as the *list* and *delete* shell commands. They can be called from the system's shell (not **our** shell) with the same arguments as the list and delete commands made for **our** shell.
- These programs capable of dealing with wildcard characters (*,?, ...). **THIS MUST BE THE LAST THING TO BE DONE**

Information on the system calls and library functions needed to code these programas is available through `man`: (*open, opendir, readdir, stat, lstat, unlink, rmdir, realpath, readlink ...*).

IMPORTANT:

- These programs should compile cleanly (produce no warnings even when compiling with `gcc -Wall`)
- **NO RUNTIME ERROR WILL BE ALLOWED (segmentation, bus error ...)**. Programs with runtime errors will yield no score.
- These programs can have no memory leaks
- When one program cannot perform its task (for whatever reason, for example, lack of privileges) it should inform the user (See *errors* section at *HEPFUL INFORMATION*)
- All input and output is done through the standard input and output

HELPFUL INFORMATION

The following funtions (*ConvierteModo(), ComvierteModo2()* and *Convierte-*

Modo3()) convert the mode of one file (in a *mode_t* integer) to `"-rwxrwxrwx"` form. Note that these three functions have different ways of memory allocation. You can use any of them (it's the programmer's choice), but make sure you understand how they work and the implications of choosing one over the others

```
char LetraTF (mode_t m)
{
  switch (m&S_IFMT) { /*and bit a bit con los bits de formato,0170000 */
    case S_IFSOCK:  return 's';  /*socket */
    case S_IFLNK:   return 'l';    /*symbolic link*/
    case S_IFREG:   return '-';   /* fichero normal*/
    case S_IFBLK:   return 'b';   /*block device*/
    case S_IFDIR:   return 'd';   /*directorio */
    case S_IFCHR:   return 'c';   /*char  device*/
    case S_IFIFO:   return 'p';   /*pipe*/
    default: return '?';   /*desconocido, no deberia aparecer*/
  }
}

char * ConvierteModo (mode_t m, char *permisos)
{
  strcpy (permisos,"---------- ");

  permisos[0]=LetraTF(m);
  if (m&S_IRUSR) permisos[1]='r';  /*propietario*/
  if (m&S_IWUSR) permisos[2]='w';
  if (m&S_IXUSR) permisos[3]='x';
  if (m&S_IRGRP) permisos[4]='r';   /*grupo*/
  if (m&S_IWGRP) permisos[5]='w';
  if (m&S_IXGRP) permisos[6]='x';
  if (m&S_IROTH) permisos[7]='r';   /*resto*/
  if (m&S_IWOTH) permisos[8]='w';
  if (m&S_IXOTH) permisos[9]='x';
  if (m&S_ISUID) permisos[3]='s';  /*setuid, setgid y stickybit*/
  if (m&S_ISGID) permisos[6]='s';
  if (m&S_ISVTX) permisos[9]='t';
  return permisos;
}

char * ConvierteModo2 (mode_t m)
{
  static char permisos[12];
  strcpy (permisos,"---------- ");

  permisos[0]=LetraTF(m);
  if (m&S_IRUSR) permisos[1]='r';  /*propietario*/
  if (m&S_IWUSR) permisos[2]='w';
  if (m&S_IXUSR) permisos[3]='x';
  if (m&S_IRGRP) permisos[4]='r';   /*grupo*/
  if (m&S_IWGRP) permisos[5]='w';
  if (m&S_IXGRP) permisos[6]='x';
  if (m&S_IROTH) permisos[7]='r';   /*resto*/
  if (m&S_IWOTH) permisos[8]='w';
  if (m&S_IXOTH) permisos[9]='x';
  if (m&S_ISUID) permisos[3]='s';  /*setuid, setgid y stickybit*/
  if (m&S_ISGID) permisos[6]='s';
  if (m&S_ISVTX) permisos[9]='t';
  return (permisos);
}

char * ConvierteModo3 (mode_t m)
{
  char * permisos;
  permisos=(char *) malloc (12);
  strcpy (permisos,"---------- ");

  permisos[0]=LetraTF(m);
  if (m&S_IRUSR) permisos[1]='r';  /*propietario*/
  if (m&S_IWUSR) permisos[2]='w';
  if (m&S_IXUSR) permisos[3]='x';
  if (m&S_IRGRP) permisos[4]='r';   /*grupo*/
  if (m&S_IWGRP) permisos[5]='w';
  if (m&S_IXGRP) permisos[6]='x';
  if (m&S_IROTH) permisos[7]='r';   /*resto*/
  if (m&S_IWOTH) permisos[8]='w';
  if (m&S_IXOTH) permisos[9]='x';
  if (m&S_ISUID) permisos[3]='s';  /*setuid, setgid y stickybit*/
  if (m&S_ISGID) permisos[6]='s';
  if (m&S_ISVTX) permisos[9]='t';
  return (permisos);
}
```

Errors

When a system call cannot perform (for whatever reason) the task it was asked to do, it returns a special value (usually **-1**), and sets an external integer variable variable (**errno**) to an error code

The man page of the system call explains the reason why the system call produced such an error code.

A generic message explaning the error can be obtained with any of these methods:

- the *perror()* function prints that message to the standard error (the screen, if the standard error has not been redirected)
- the *strerror()* function returns the string with the error description if we supply it with the error code
- the external array of pointers, `extern char * sys_errlist[]`, contains the error descriptions indexed by error number so that `sys_errlist[errno]` has a description of the error associated with *errno*

WORK SUBMISSION

- Work must be done in pairs.
- The source code will be submitted to the subversion repository under a directory named **P1**
- The name of the main programs will be `shell.c`, `list.c` and `delete.c`. Programs must be able to be compiled with `gcc shell.c`, `gcc list.c` and `gcc delete.c` Optionally a `Makefile` can be supplied so that all of the programs can be compiled with just `make`
- **ONLY ONE OF THE MEMBERS OF THE GROUP** will sub-

mit the source code. The names and logins of all the members of the group should be in the source code of the main programs (at the top of the file)

- Works submited not conforming to these rules will be disregarded.

DEADLINE: 23:00, Friday November 6th, 2020

ASSESSMENT: For each pair, it will be done in its corresponding group, during the lab classes