Operating Systems

Grado en Informática. Course 2017-2018

Assigment 1: File systems

PART 1

Add to the shell started in previous assigments the following commands

info – gives info on the files and or directories supplied as its arguments

- its complete sysntax is

  `info name1 name2 name3 ....`

  where

  * **name1 name2 name3 ...** are the names of the files (or directories) to give info for.
  * it will produce a line for each item info is given on, in the same format as `ls -li` (resolving, if necessary, symbolic links). It is equivalent to `ls -li` for files and `ls -lid` for directories. **IT DOES NOT LIST THE CONTENTS OF A DIRECTORY**, just the directory itself.

- Example:

```
-> info  enlace p1.c /home/antonio
15888887 -rw-r--r--   1 antonio antonio     4 Sep 15 13:35 enlace -> fich
15888887 -rw-r--r--   1 antonio antonio  2713 Sep 10 11:40 p1.c
15859713 drwxrwxrwx 156 antonio antonio 12288 Sep 20 12:13 /home/antonio/
```

recursive – displays the flag recursive, which can be `ON` or `OFF`

- its complete sysntax is

  `reursive [ON | OFF]`

- **recursive ON** sets the flag to ON
- **recursive OFF** sets the flag to OFF
- this flag affects how the following command will list directories. If the flag is ON directories will be listed recursively.

list – lists the directories and/or files supplied to it as command line arguments

- its complete sysntax is

  `list [-l] name1 name2 name3 ....`

  where

  * **-l** stands for long listing: if present, directories (and files) would be listed **EXACTLY** as the command `info` does. Otherwise, just the size and name of the file (or directory) will be printed.
  * For directories **ITS CONTENTS WILL ALSO BE LISTED**. If the flag *recursive* is on , directories will be listed recursively
  * 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` of the directory entry.
  * If no names are given the current working directory will be listed.

eliminate

- deletes a file or a directory
- its complete sysntax is

  `eliminate [-f] name`
- if **-f** is given and name is a directory, it will try to delete the directory and **ALL OF ITS CONTENTS**. If **-f** is not given, a directory will only be deleted if its empty.
- if no *name* is given it will do nothing
- When a file or directory cannot be removed, an appropiate message must be sent to the user

PART 2

- Create two standalone programs `list.c` and `eliminate.c` that will do the same things as the *list* and *eliminate* shell commands. The recursive flag for the list program will be specified via a `-r` option (Example: `list -r -l /home/usuario /var/mail`).
- Make these programs capable of dealing with wildcard characters (*,?, ...)

Information on the system calls and library funtions needed to code

these programas is available through `man`: (*opendir, readdir, stat, lstat, unlink, rmdir, realpath ...*).

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
- All input and output is done through the standard input and output

HELPFUL INFORMATION

The following funtions convert the mode of one file (in a *mode_t* integer) to `"rwxrwxrwx"` form. Note that the three functions have different ways of allocating memory

```
char TipoFichero (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]=TipoFichero(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]=TipoFichero(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]=TipoFichero(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 `eliminate.c`. Programs must be able to be compiled with `gcc shell.c`, `gcc list.c` and `gcc eliminate.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 submit 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 October 20th, 2017

ASSESSMENT: Week starting October 23rd