Ejercicios Prácticas SO

Grado Ingeniería Informática

E.T.S. Ing. Informática y de Telecomunicación (ETSIIT)

Granada

Alejandro Alcalde
elbauldelprogramador.com

Ejercicios sesiones Prácticas

Alejandro Alcalde

1.1. Sesión 6

Exercise 1. Implementa un programa que admita tres argumentos. El primer argumento será una orden de Linux; el segundo, uno de los siguientes caracteres "<" o ">", y el tercero el nombre de un archivo (que puede existir o no). El programa ejecutará la orden que se especifica como argumento primero e implementará la redirección especificada por el segundo argumento hacia el archivo indicado en el tercer argumento. Por ejemplo, si deseamos redireccionar la salida estándar de sort a un archivo temporal, ejecutaríamos (el carácter de redirección > lo ponemos entrecomillado para que no lo interprete el shell y se coja como argumento del programa):

Ejercicio 1.c

```c
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
#include <string.h>

int main(int argc, char *argv[]) {

    if (argc < 4){
        printf("Uso: %s <programa> <<|>> <fichero>.\n", argv[0]);
        exit(-1);
    }

    char redireccion = argv[2][0];
    int fd;

    switch(redireccion){
        case '<':
            /* Proporcionamos los datos al programa desde el fichero indicado */
            close(STDIN_FILENO);
```


[1] ETSIIT, Granada.

E-mail address: algui91@gmail.com.

January 11, 2014

```
            if ((fd = open(argv[3], O_RDONLY)) == -1){
                perror("open");
                exit(-1);
            }
            if (fcntl(fd, F_DUPFD, STDIN_FILENO) == -1 ){
                perror ("fcntl falló");
                exit(-1);
            }
            if( (execlp(argv[1],argv[1], NULL)<0)) {
                    perror("Error en el execl\n");
                    exit(-1);
            }

            break;
        case '>':
            /* Escribiremos la salida del programa al fichero indicado */

            close(STDOUT_FILENO);
            if ((fd = open(argv[3], O_WRONLY | O_TRUNC | O_CREAT)) == -1){
                perror("open");
                exit(-1);
            }
            if (fcntl(fd, F_DUPFD, STDOUT_FILENO) == -1 ){
                perror ("fcntl falló");
                exit(-1);
            }
            if( (execlp(argv[1],argv[1], NULL)<0)) {
                    perror("Error en el execl\n");
                    exit(-1);
            }
            break;
    }

    exit(0);
}
```

Exercise 2. Reescribir el programa que implemente un encauzamiento de dos órdenes pero utilizando fcntl. Este programa admitirá tres argumentos. El primer argumento y el tercero serán dos órdenes de Linux. El segundo argumento será el carácter "|". El programa deberá ahora hacer la redirección de la salida de la orden indicada por el primer argumento hacia el cauce, y redireccionar la entrada estándar de la segunda orden desde el cauce.

Ejercicio 2.c

```
#include<sys/types.h>
#include<fcntl.h>
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
#include<errno.h>

int main(int argc, char *argv[])
{
    int fd[2];
    pid_t PID;
```

```
12
13      pipe(fd); // Llamada al sistema para crear un pipe
14
15      if (argc != 4){
16          printf("Uso: %s <programa1> | <programa2>\n", argv[0]);
17          exit(-1);
18      }
19
20      char *programa_1 = argv[1];
21      char *programa_2 = argv[3];
22
23      if ( (PID= fork())<0) {
24          perror("\Error en fork");
25          exit(-1);
26      }
27      if (PID == 0) { // ls
28          //Cerrar el descriptor de lectura de cauce en el proceso hijo
29          close(fd[0]);
30
31          //Duplicar el descriptor de escritura en cauce en el descriptor
32          //correspondiente a la salida estandar (stdout), cerrado previamente en
33          //la misma operacion
34          if (fcntl(fd[1], F_DUPFD, STDOUT_FILENO) == -1 ){
35              perror ("fcntl fallo");
36              exit(-1);
37          }
38          execlp(programa_1,programa_1,NULL);
39      }
40      else { // sort. Proceso padre porque PID != 0.
41          //Cerrar el descriptor de escritura en cauce situado en el proceso padre
42          close(fd[1]);
43
44          //Duplicar el descriptor de lectura de cauce en el descriptor
45          //correspondiente a la entrada estandar (stdin), cerrado previamente en
46          //la misma operacion
47          if (fcntl(fd[0], F_DUPFD, STDIN_FILENO) == -1 ){
48              perror ("fcntl fallo");
49              exit(-1);
50          }
51          execlp(programa_2,programa_2,NULL);
52      }
53
54      return(0);
55  }
```