DOCUMENTACIÓN PRÁCTICA SHELL.C

La práctica consiste en una función principal (***main***) en la cual tenemos el bucle *while* que imprime el *prompt* que hemos definido como constante, pide el comando al usuario, mediante un *for* comprueba si se deberá ejecutar en segundo plano poniendo la bandera *segplano* a 1, y una vez comprobado si hay o no segundo plano, llamamos a la función ***CadIguales***.

CadIguales

CadIguales es una función que comprueba si la cadena "FIN" es igual que la cadena que introduce el usuario, en caso afirmativo sale del programa mediante un exit(0), en caso contrario devuelve 0 a fin en la función ***main***.

Después de haber hecho esta comprobación, desde la función ***main*** llamamos a la función ***Comando*** pasándole como parámetros la cadena que ha introducido el usuario y la bandera *segplano*.

Comando

Una vez ha entrado en ***Comando*** declaro las variables *redirec* (almacenará el nombre del fichero donde el usuario quiera volcar la información)*, argumentoInd, argumendoInd2* de tipo cadena de cadenas de caracteres, donde se almacenará un argumento caso de que no haya tuberías o 2 argumentos si el usuario introduce tuberías, y las variables *argumento* y *argumento2* de tipo puntero a cadena de cadena de caracteres, que nos servirá para que el *execvp* ejecute los comandos, ya que es el tipo de datos que debe recibir. Inicializamos las variables enteras *k* e *i* a 0 para comenzar a recorrer la cadena y la variable *flag_tuberias* también a 0 (esta bandera determinará si ejecutamos un proceso único o dos procesos concatenados mediante tuberías).

1. Entra en el primer *while*, del cual le indicamos que no salga mientras que el carácter que lea sea diferente del terminador de cadena "\0", del símbolo de las tuberías "|" o de los redireccionamientos ">" o "<". Una vez dentro, mediante un *for* recorro la cadena hasta que encuentre un espacio, un terminador de cadena, una tubería o un ">" "<", y así separo los diferentes argumentos que introduce el usuario. Al salir del bucle hago que la cadena que hemos creado acabe con un terminador de cadena y hago que argumento apunte a la cadena creada, si hemos salido del bucle *for* anterior, porque encontró un espacio, incrementamos la i en una unidad y si hemos salido porque ha encontrado el símbolo de redireccionamiento de entrada "<" separará esa cadena guardándola en la variable *entrada*, es decir, si el usuario introduce: *wc < fichero.txt*, en la variable *entrada* se almacenará *fichero.txt.*

2. Al salir del bucle hago que el puntero a cadena de cadenas de caracteres acabe en *NULL*, ya que sino daría error y ahora entramos en un *if*. Si el carácter cadena[i] es un ">" guardará en una nueva cadena los caracteres hasta que encuentre un terminador de cadena, una tubería o un espacio. Si el carácter por el que salió del *for* anterior era una tubería "|" entra en el *if* y hace el mismo proceso que en el primer *while* de la función pero con *argumentoInd2* y *argumento2*, obteniendo así el segundo argumento.

3. Al finalizar lo anterior, pueden darse dos casos, uno de ellos es que la variable *ejecutar* tenga valor 0, con lo que entrará en el *if* y comprobará otros 2 casos posibles, uno es que se ejecute un único proceso (sale del primer o segundo *while* porque encontró fin de cadena, con lo que el valor de *flag_tuberias* seguirá siendo 0), caso en el cual llamará a la función ***CrearProceso*** pasándole como parámetro argumento y plano (variable que nos dice si hay que ejecutar un proceso en segundo plano o no), o de lo contrario *flag_tuberias* tendrá valor 1 y ejecutará ***tuberías*** pasándole como parámetro *argumento y argumento2*. Y el otro caso es que *ejecutar* tenga valor 1, en tal caso imprimirá un mensaje de *Error de sintaxis*, ya que el usuario habrá introducido mal los comandos. Por ejemplo:

RedirecEntrada

Esta función nos sirve para redirigir la entrada de datos estándar a un fichero, es decir, que si un usuario quiere ejecutar un comando pasándole como entrada no lo que él escriba por teclado, sino lo que hay dentro de un fichero, podrá hacerlo. Por ejemplo: *wc < fichero*

RedirecSalida

Esta función nos sirve para redirigir la salida de datos estándar a un fichero, es decir, que si un usuario ejecuta un comando, en lugar de mostrarlo en pantalla, vuelque la información en un fichero que el usuario haya elegido. Por ejemplo, si el usuario escribe *ls –la < ficherols* , guardará la información que nos dé el comando escrito en el fichero *ficherols*.

Tuberías

Si el usuario introdujo dos comandos unidos mediante la tubería entrará en la función ***tuberías.*** Declaro las variables *fd* de tipo array de 2 elementos de tipo entero, *estado* de tipo entero e *hijo* de tipo pid_t.

Inicializamos la variable *hijo* con un *fork*, es decir, creo el proceso hijo que ejecutará los comandos, ya que si ejecutase los comandos mediante execvp de forma directa, el programa dejaría de funcionar y acabaría con la ejecución de las tuberías. Compruebo si el hijo esta bien creado mediante un *if,* caso de no crearlo bien da un mensaje de error, y si ha sido bien creado comienza creando la tubería mediante la

función *pipe* la cual recibe el contenido de la posición 0 del array a enteros creado anteriormente (&fd[0]), mediante el *if* y el *else* siguientes se ejecutan de forma concatenada los dos comandos que introdujo el usuario, funcionando así la tubería.

CrearProceso

Si el usuario tan sólo introdujo un comando, entrará en esta función que recibe como parámetros *argumento* (un puntero a un array de cadena de caracteres) y *plano* (un entero que nos dirá si hay que ejecutar el comando en segundo plano o no).

Declaramos como variables locales *estado* de tipo entero e *hijo* de tipo pid_t. Creamos el hijo por la misma razón que en las tuberías, si no se crease, ejecutaríamos el comando pero finalizaría el programa, así, una vez creado el hijo ejecutará el comando deseado mediante *execvp* y si hemos elegido ejecutarlo en segundo plano, no entrará al *else* que indica que *hijo* sea *wait(&estado),* provocando así que quedé en segundo plano.