Trabajo practico N°1 – Sistemas Operativos 1 – 2019

Aichino Ignacio - Altamirano Julieta

Este trabajo consta con programas en C para GNU/Linux. Los distintos programas cumplen determinadas funciones y se especifican a continuación:

1. **main.c:** contiene el main del programa. Ejecuta los comandos ingresados por pantalla, con la ayuda de las funciones declaradas en los otros archivos C. Además, de ingresarse como parámetro en su ejecución un archivo, este lee los comandos dentro del mismo y los ejecuta, en lugar de que el usuario los ingrese por pantalla. Consta, además del main, de la siguiente función:
    - **fileCommand:** esta función recibe como parámetro el nombre del archivo del cual se van a leer los comandos a ejecutar. Su función es leer línea por línea e ir ejecutando cada una de ellas.
2. **structures.h:** este header file contiene la declaración de las funciones de todos los archivos enumerados a continuación (3 a 6), para que puedan ser llamadas independientemente del archivo en el que fueron creadas. Además, contiene estructuras de datos que también son accedidas desde todos los archivos del programa, para su lectura y modificación. Las estructuras de datos son las siguientes:
    - **DirectoryValues:** esta estructura de datos contiene las variables que son principalmente modificadas en el archivo “change_directory.c”.
    - **InstructionValues:** esta estructura de datos contiene las variables que son principalmente modificadas en el archivo “instructions.c”.
    - **ExecuteValues:** esta estructura de datos contiene las variables que son principalmente modificadas en el archivo “execute.c”.
    - **RedirectionValues:** esta estructura de datos contiene las variables que son principalmente modificadas en el archivo “redirection.c”.
3. **change_directory.c:** contiene las funciones encargadas de la correcta ejecución del comando “cd”**.** Consta de las siguientes funciones y su finalidad:
    - **initializeDirectoryValues:** inicializa las variables de la estructura “DirectoryValues”.
    - **commandLinePrompt:** inicializa el prompt en la dirección “/home/usuario”.
    - **fixDirectionPath:** recibe como parámetros una variable para almacenar la dirección corregida y un booleano que se encarga de avisar si se trata de una sola palabra o varias (dependiendo de si la dirección contiene espacios o no). Esta función se encarga de corregir la dirección ingresada por el usuario. En caso de no contener espacios el nombre de la carpeta, se limita a quitarle el “\n” del final. En caso de contenerlos, además de lo anterior, como el código guarda el nombre como palabras separadas, se encarga de unir todas en una misma variable, separando cada palabra con su respectivo espacio, para que pueda ser ejecutada correctamente la redirección.
    - **changeDirectory:** se encarga de chequear si la instrucción “cd” fue ingresada con los parámetros correspondientes y de su correcta ejecución. Si se ingresó “cd”, vuelve a la

dirección “/home/usuario”, si se ingresó “cd ..” vuelve atrás una carpeta y con el comando “cd carpeta” se mueve a esa carpeta, solo si esta existe en el directorio actual. También se encarga de actualizar el prompt con la dirección actual.

4. **instructions.c:** se encarga del tratamiento de las instrucciones ingresadas por pantalla o mediante el batchfile. Lo hace por medio de las siguientes funciones:
    - **saveInput:** recibe como parámetro la entrada que realizó el usuario por pantalla con la instrucción a ejecutar. Se encarga de almacenarla en un buffer para su futura ejecución.
    - **inputClassification:** chequea la primera palabra de la instrucción almacenada previamente en el buffer y clasifica la entrada dependiendo ésta, para su futura ejecución.
5. **execute.c:** es el archivo encargado de la ejecución de las instrucciones.
    - **initializeExecuteValues:** inicializa las variables de la estructura “ExecuteValues”.
    - **errorMenssage:** es la función encargada de analizar el error e imprimir en pantalla el mensaje apropiado para el mismo.
    - **execute:** se encarga de ejecutar la instrucción dependiendo de la clasificación dada por “inputClassification”. Las distintas ejecuciones son realizadas por el proceso hijo, exceptuando la del “cd”, ya que, si redirigimos la dirección en el proceso hijo, una vez que este muere perdemos la redirección. Por ello, se optó por redirigir el proceso padre directamente para que el mismo conserve la dirección actual.
6. **redirection.c:** este archivo es el encargado de realizar la redirección del comando “echo”. Dependiendo de lo ingresado por el usuario, se puede optar por imprimir en pantalla el contenido de un archivo o de escribir en este lo ingresado por pantalla.
    - **initializeRedirectionValues:** inicializa las variables de la estructura “RedirectionValues”.
    - **output:** se encarga de escribir lo ingresado por pantalla después del comando “echo” y de escribirlo dentro del archivo que se menciona al final, luego del símbolo “>”. Si el archivo mencionado no existe, lo crea, y si el mismo ya tiene contenido dentro, le agrega el nuevo, no lo sobrescribe.
    - **input:** se encarga de leer el contenido del archivo mencionado después del símbolo “<”, en caso de existir el mismo, y de imprimirlo por pantalla.
7. **batchfile:** archivo que contiene un set de instrucciones a leer y ejecutar por el programa.