SISTEMAS OPERATIVOS

GRADO EN INGENIERÍA INFORMÁTICA

Práctica 1: Llamadas al sistema operativo

Curso 2019/2020

Jorge Rodríguez Fraile, 100405951, Grupo 81, 100405951@alumnos.uc3m.es
Carlos Rubio Olivares, 100405834, Grupo 81, 100405834@alumnos.uc3m.es

Índice

Descripción de código

mycat.c

Lo que primero que haremos será comprobar que se pasa el parámetro necesario y que no se exceda la longitud máxima (PATH_MAX), que es la dirección del fichero a copiar, y si no se proporciona avisa de la falta y devuelve -1. Si hemos recibido la dirección del fichero como parámetro lo abrimos, comprobando que no falle, e iremos leyendo el fichero en trozos de 1024 bytes, que es el tamaño del buffer, y los iremos imprimiendo por pantalla con write. Cuando read nos devuelva 0 habremos acabado de leer el fichero y cerramos el fichero antes de que termine el programa, también comprobado que no falle el cierre del fichero actual.

myls.c

Al igual que en mycat, primero comprobamos si nuestro directorio sobrepasa nuestra constante límite, en cuyo caso devuelve un -1. Como este programa puede admitir un argumento o ninguno, hacemos una bifurcación: si no tiene ningún argumento, abrimos el directorio actual con getcwd, si tiene un argumento, abrimos el directorio que se nos pasa como argumento, y, en caso contrario (más de un argumento), devolvemos -1. Una vez hecho esto, comprobamos que el directorio se ha abierto bien, y empezamos a leer sus ficheros. Si la lectura del fichero en el que nos encontramos no es NULL, imprimimos su nombre. Una vez obtengamos NULL, salimos del bucle de lectura, y cerramos el directorio, comprobando que se ha cerrado correctamente.

mysize.c

Para este programa hacemos las mismas comprobaciones con la constante ‘PATH_MAX’ como en los dos anteriores. Hecho esto, comprobamos la correcta apertura del directorio y creamos un bucle de lectura de ficheros idéntico al myls. En cada lectura comprobamos si el fichero se ha abierto correctamente, y por otro lado si el tipo de fichero (d_type) coincide con uno regular mediante la constante ‘DT_REG’, si ha dado error al abrir uno de los ficheros indicará con un mensaje por pantalla cuál ha sido y continuará con el siguiente fichero. Hecho esto, se consigue su tamaño mediante la llamada lseek (utilizando como argumentos el fichero actual, 0 como desplazamiento, y SEEK_END como posición inicial donde comenzar el desplazamiento; como el desplazamiento es 0, SEEK_END acaba siendo el tamaño de nuestro fichero) y se imprime junto con su nombre, después se cierra dicho fichero y se comprueba que se ha cerrado correctamente. Una vez terminado el bucle, se cierra el directorio y se comprueba si ha dado error, terminando así el programa.

Pruebas realizadas

mycat.c

Prueba propuesta, consiste en ejecutar el programa con el archivo proporcionado (f1.txt) y ver que funciona correctamente. Esta prueba fue pasada con éxito.

Falta de parámetros, ejecutamos el programa pero sin pasar la dirección de un fichero, y el programa deberá avisar de que falta y devolver -1. El resultado fue el esperado, se imprimió 'Not enough arguments' y devuelve fallo.

Pasar un fichero inexistente como argumento, el programa deberá dar error de apertura ya que no encontrará el fichero en el directorio. Tras realizar la prueba es nos devuelve "Error al abrir fichero", por lo que supera esta prueba.

Pasar un fichero y que no tengamos permisos de lectura, el programa tendrá que devolver un "error al leer fichero" en pantalla y un -1 como resultado. Esta prueba ha sido superada con éxito.

myls.c

La primera prueba que proponemos es un run del programa en el directorio de la carpeta p1_tests. El resultado tendrá que ser un listado de todos los elementos que se encuentran en dicha carpeta. El resultado es el esperado.

Por otro lado, intentamos leer como argumento un directorio que no exista, el resultado que obtenemos es un error al abrir el directorio, lo esperado.

También hemos probado ejecutar el programa sin argumentos, con lo que tendría que obtener el directorio actual y seguir con el programa sin ningún problema.

Direccion del directorio superior al máximo permitido (PATH_MAX), el programa tendría que devolver un -1. El resultado es correcto.

mysize.c

Como prueba inicial, probamos el mysize.c en la carpeta de tests, que nos da el resultado esperado, cada fichero .txt con su tamaño espaciado con una tabulación.

A continuación, probamos con un directorio que contenga un fichero sin los permisos necesarios, lo que nos da un error de lectura.

Como última prueba, estaremos en un fichero que supere nuestro límite de tamaño, el programa devuelve error y un -1.

Conclusiones

Durante la creación de estos programas, hemos tenido problemas a la hora de entender la utilización de la función lseek en mysize, aunque con la ayuda de diversos foros y vídeos, ha resultado fácil de solucionar. Por otro lado, hemos tenido un problema con la constante MAX_PATH, ya en el directorio en el que nos encontrábamos era demasiado largo y superaba el umbral delimitado por la constante, por lo que no imprimía nada. Aparte de esto, no hemos encontrado problemas mucho más grandes, ya que gracias a la guía y páginas web de foros de código adicionales, hemos podido desenvolvernos de manera relativamente sencilla a lo largo de la práctica.

Gracias a esta práctica hemos podido ver la estructura interna de los ficheros y cómo trabajar con ellos, además de profundizar un poco más en el lenguaje de C con nuevas funciones como lseek, o constantes como MAX_PATH, por otro lado hemos creado nuestra propia versión de comandos existentes en la terminal como ls, que nos ayuda a entender algo más el funcionamiento de este tipo de instrucciones.