2024

SISTEMAS OPERATIVOS

PRÁCTICA 1. LLAMADAS AL SISTEMA OPERATIVO

PABLO GRADOLPH OLIVA – NIA: 100458456

GRADO EN INGENIERÍA INFORMÁTICA | UNIVERSIDAD CARLOS III DE MADRID

Índice:

1. MYWC

El programa *'mywc'* es una herramienta hecha para imitar la funcionalidad del comando *'wc'* de Unix/Linux, proporcionando una forma de contar líneas, palabras y bytes de un archivo. Está programado con un enfoque destinado a profundizar en el entendimiento de las llamadas al sistema, y dejando de lado el uso típico de librerías en C para el manejo de ficheros.

1.1. FUNCIONAMIENTO

Al ejecutarse, 'mywc' lleva a cabo una serie de pasos que aseguran el correcto funcionamiento:

1. **Validación de Argumentos:** El programa verifica que se haya proporcionado un argumento adicional al nombre del programa, es decir, el nombre del archivo que se quiere procesar. Si el número de argumentos es insuficiente, se informa al usuario con un mensaje de error *'Too few arguments'* y termina su ejecución retornando -1. En caso de pasar más argumentos, el programa solo tendrá en cuenta el primero que se pase después del nombre del programa y trabajará con este.
2. **Apertura del Archivo:** Utilizando la llamada al sistema *'open'* con el modo *'O_RDONLY'*, *'mywc'* intenta abrir el archivo especificado en el valor del segundo argumento *'argv[1]'* en modo de solo lectura (Con este modo protegemos la integridad del archivo evitando modificaciones accidentales). Si la apertura del archivo falla, el programa notifica al usuario mostrando *'Error opening file: nombre_del_archivo'* y retornando -1.
3. **Procesamiento del archivo:** Una vez abierto el archivo, el programa procede a leerlo byte a byte utilizando la llamada al sistema *'read'*. Cada byte leído se analiza para determinar si constituye una nueva línea, una nueva palabra o si solamente se añade 1 al conteo total de bytes. Este procesamiento se realiza de la siguiente manera:
    - **Conteo de bytes:** Se incrementa el contador de bytes *'num_bytes'* por cada byte leído del archivo.
    - **Conteo de líneas:** S se encuentra un salto de línea *'\n'* se incrementa un contador de líneas *'num_lines'*.
    - **Conteo de palabras:** El conteo de palabras se hace identificando transiciones de caracteres considerados espacios por la función *'isspace'*, lo que incluye entre otras cosas, tabulaciones *'\t'* a caracteres que no son considerados espacios. Para ello, se utiliza una variable de estado, *'in_word'* para rastrear si la lectura del byte actual se encuentra en una palabra o no.
        i. Si el carácter es un espacio y previamente estábamos en una palabra *('in_word == 1'),* incrementamos el contador de palabras y marcamos que ya no estamos en una palabra *('in_word = 0').*
        ii. Si el carácter no es un espacio y no estábamos en una palabra *('in_word == 0'),* marcamos que ahora estamos en una palabra *('in_word = 1').* Esto no incrementa el contador de palabras, pero prepara el estado para que un futuro espacio pueda marcar el fin de esta palabra.

4. **Cierre y Reporte:** Por último, se cierra el archivo y se imprime el resultado como:

<líneas><espacio><palabras><espacio><bytes><espacio><nombre_del_fichero>

1.2. BATERÍA DE PRUEBAS

Para comprobar los resultados obtenidos por mi programa, tras compilar sin *warnings* he ejecutado los siguientes comandos:

1. ./mywc miarchivo.txt > salida_mywc.txt
2. wc miarchivo.txt > salida_wc.txt
3. diff salida_mywc.txt salida_wc.txt

Para distintos ejemplos de archivos:

- Archivo vacío.
- Archivo de una sola palabra.
- Archivo con una sola línea.
- Archivo estándar: Incluyendo varias líneas y palabras, para validar el conteo adecuado en un escenario típico.
- Archivos con gran cantidad de líneas vacías.
- Archivos con caracteres no ASCII para asegurar la precisión en el conteo de bytes.
- Archivos con concatenaciones de espacios y/o *'\t'* para asegurar la precisión en el conteo de palabras.

También se han hecho pruebas para los manejos de errores:

- Archivo inexistente.
- Argumentos inválidos.
- Permisos insuficientes.
- Verificación de mensajes de error correctos y claros.

Solo en algún caso he encontrado diferencias entre el comando de Unix '*wc'* y mi programa. Y se deben a que la salida de '*wc*' a veces encuentra espacios al principio, dos espacios y/o *'\t'* entre los resultados. Sin embargo, *'mywc'* imprime los resultados tal y como se explica en el apartado anterior. Y, en ningún caso, se encuentran diferencias en los números obtenidos de bytes, líneas o palabras.

2. MYLS

El programa *'myls'* es una implementación personalizada del comando *'ls'* de Unix/Linux, al igual que el ejercicio anterior, desarrollado con un enfoque de llamadas al sistema, pero, en este caso, para el manejo de directorios. El programa busca listar todas las entradas de un directorio específico con una salida más comparable a la ejecución del comando *'ls -f -1'* de Unix/Linux.

2.1. FUNCIONAMIENTO

El flujo de *'myl'* se explica de la siguiente manera:

1. **Determinación del Directorio:**
    a. Si se proporciona un argumento, *'myls'* intenta abrir el directorio específico del argumento pasado.
    b. Si no se proporciona ningún argumento, *'myls'* utiliza el directorio actual, obtenido mediante la llamada al sistema *'getcwd'*.

2. **Apertura del Directorio:** Se realiza mediante la llamada al sistema *'opendir'*. En caso de fallar (por ejemplo, si el directorio no existe o no se tienen los permisos adecuados), se notifica al usuario y el programa termina retornando -1.
3. **Lectura y listado del Directorio:** Se utiliza *'readdir'* en un bucle para leer cada entrada del directorio. Cada nombre de archivo o directorio leído se imprime en la salida estándar, mostrando así el contenido del directorio.
4. **Cierre del Directorio:** Una vez listadas todas las entradas, se cierra el directorio con la llamada *'closedir'*.

2.2. BATERÍA DE PRUEBAS

Para comprobar los resultados obtenidos por mi programa, tras compilar sin *warnings* he ejecutado los siguientes comandos:

1. ./myls nombre_directorio > salida_myls.txt
2. ls -f -1 nombre_directorio > salida_ls.txt
3. diff salida_myls.txt salida_ls.txt

Esto se ha realizado para diferentes casos:

- Directorio vacío.
- Directorio con un archivo.
- Directorio con otro directorio dentro.
- Directorio con varios tipos de entradas.
- Directorio con nombres de archivos con caracteres especiales.

Y, para el control de errores:

- Directorio inexistente.
- Permisos insuficientes.
    - Directorio sin permisos de lectura -> Muestra un mensaje de error correcto.
    - Directorio con permiso de lectura, pero sin permiso de ejecución -> Lista correctamente.
- Comprobación de que la salida es legible y coherente.

Siempre se comprueba una salida idéntica al comando *'ls -f -1'* del que ya hemos hablado.

3. MYISHERE

El programa *'myishere'* está diseñado para verificar la presencia de un arhcivo específico dentro de un directorio dado. Al igual que en los programas anteriores, con un diseño centrado en las llamadas al sistema.

3.1 FUNCIONAMIENTO

'myishere' sigue esta secuencia de pasos para cumplir con su propósito:

1. **Análisis de Argumentos:** Primero verifica que se hayan proporcionado exactamente dos argumentos: el nombre del directorio y el nombre del archivo a buscar. Si no se cumplen estos criterios, se informa al usuario y se termina la ejecución retornando -1. En caso de pasar más argumentos, tendrá en cuenta solo los dos primeros.
2. **Apertura del Directorio:** Utiliza *'opendir'* para intentar abrir el directorio proporcionado. Si la apertura falla, se notifica al usuario y se termina el programa retornando -1.
3. **Búsqueda del Archivo:** A través de un bucle que utiliza *'readdir'*, lee cada entrada del directorio abierto, comparando el nombre de cada entrada con el nombre del arhcivo proporcionado. La búsqueda continúa hasta que se encuentra una coincidencia o hasta que se hayan leído todas las entradas.
4. **Reporte de resultados:**
    a. Si se encuentra el archivo, se imprime un mensaje indicando su presencia en el directorio.
    b. Si no se encuentra, se imprime un mensaje indicando que el archivo no está presente.
5. **Cierre del directorio** independientemente del resultado de la búsqueda.

3.2. BATERÍA DE PRUEBAS

Se han realizado las siguientes pruebas para comprobar el correcto funcionamiento de *'myishere'*:

- Archivo presente.
- Archivo ausente.
- Directorio vacío.
- Nombres de archivos especiales.
- Directorio inexistente.
- Permisos insuficientes.
- Verificación de que los mensajes de salida son claros proporcionando información útil para el usuario.

En todos los casos se obtiene con éxito lo que se espera del programa.

4. CONCLUSIÓN Y ENTREGA

El desarrollo y la implementación de los programas *'mywc', 'myls'* y *'myishere'* han sido una oportunidad para explorar y comprender las llamadas al sistema y el manejo de archivos y directorios en sistemas operativos tipo Unix/Linux.

A través de estos programas, se han abordado conceptos fundamentales como la lectura de archivos, la navegación y manipulación de directorios y el manejo de errores y permisos a nivel de sistema. La implementación de estos programas ha permitido no solo replicar la funcionalidad de comandos Unix/Linux ampliamente utilizados sino también personalizar y optimizar estas operaciones para casos de uso específicos.

Todos los programas se han sometido a un riguroso proceso de pruebas antes de darlos por terminados, tal y como se ha comentado en cada uno de los apartados. Pero, además, tras realizar el entregable tal y como se especifica en el enunciado de la práctica, se ha sometido dicho entregable al script de Python proporcionado obteniendo una calificación de 10 en el mismo y superando todas las pruebas que en él se encuentran. Obteniendo siempre el resultado esperado por el programa. Se muestra una captura de pantalla que corrobora el paso de todas y cada una de las pruebas:

Esto es un indicador adicional de la correcta funcionalidad de los programas desarrollados.