Prácticas de Ampliación de Sistemas Operativos • Tarea entregable: exec_lines

Descripción del programa `exec_lines`

Escribe un programa llamado `exec_lines` que lea un flujo de líneas (secuencias de bytes terminadas en `\n`) por la entrada estándar en **bloques de tamaño configurable**, y ejecute cada una de dichas líneas como si fueran órdenes introducidas por teclado. Para simplificar el problema, supondremos que el único carácter de control que se recibe es `\n`, y que el único separador es el espacio en blanco (no hay tabuladores). También supondremos que **las líneas no tendrán más de un tamaño máximo determinado**, y si una línea fuera mayor, devolveríamos un error indicándolo, y terminaríamos la ejecución. Cada una de las lineas a ejecutar puede tener **una** redirección o **una** tubería, es decir, que cada línea puede ser de los siguientes cinco tipos:

- `comando opciones`
- `comando opciones < fichero_entrada`
- `comando opciones > fichero_salida`
- `comando opciones >> fichero_salida`
- `comando1 opciones1 | comando2 opciones2`

donde `comando` es cualquier ejecutable. Ejemplos serían:

- `cat mi_fichero`
- `wc -l < fichero_entrada`
- `ls -la > fichero_salida`
- `df -h -u >> fichero_salida`
- `ls -l -a | wc -l`

El programa tiene que aceptar las siguientes opciones:

```
./exec_lines [-b BUF_SIZE] [-l MAX_LINE_SIZE]
```

donde `BUF_SIZE` tomará el valor 16 por defecto, siendo el tamaño de buffer que se deberá usar al leer la entrada estándar, y MAX_LINE_SIZE es el tamaño máximo de la línea que admitimos, tendrá un valor por defecto de 32 caracteres y **si una línea supera esa longitud, `exec_lines` sacará un error indicándolo** (el texto del error se puede ver en los ejemplos de ejecución)

Explicación detallada

El programa deberá leer la entrada estándar línea a línea usando la llamada al sistema `read` con el tamaño de bloque que se le pase. Si la línea supera la longitud máxima permitida no se ejecutará, sino que se sacará un mensaje de error indicando el número de línea y su contenido, y se parará la ejecución.

Una vez leída la línea, tendréis que analizarla buscando alguno de los operadores de redirección o tubería: `<`, `>`, `>>`, `|`. **Solamente dejamos que aparezca uno de ellos**, por lo que si hay más de uno, deberéis sacar

un mensaje de error (ver los ejemplos de ejecución). En el caso de los operadores de redirección, vuestro programa tendrá que abrir el correspondiente fichero (para lectura o escritura según corresponda, añadiendo o sobreescribiendo según el separador), hacer la adecuada redirección en la tabla de descriptores y ejecutar el programa solicitado. En el caso del operador de tubería, tendréis que lanzar dos procesos: el de la parte izquierda con la redirección adecuada para escribir en la tubería, y el de la parte derecha con la redirección para leer de la tubería, ejecutando cada uno el programa adecuado.

En el caso de que la ejecución de alguna línea produjera un valor de salida distinto de 0, se detendría la ejecución de `exec_lines`. Para informar del problema, se sacará la salida estándar de error el número de línea que causó el error y el valor de exit del proceso convenientemente interpretado. Si fue un valor de terminación por un exit indicando un error, se mostrará el valor devuelto, y si la terminación se produjo por una señal que mató al proceso, se mostrará el valor de dicha señal.

Ejemplos de ejecución de `exec_lines`

```
$ ./exec_lines -h
Uso: ./exec_lines [-b BUF_SIZE] [-l MAX_LINE_SIZE]
Lee de la entrada estándar una secuencia de líneas conteniendo órdenes
para ser ejecutadas y lanza los procesos necesarios para ejecutar cada
línea, esperando a su terminación para ejecutar la siguiente.
-b BUF_SIZE         Tamaño del buffer de entrada 1<=BUF_SIZE<=8192
-l MAX_LINE_SIZE    Tamaño máximo de línea 16<=MAX_LINE_SIZE<=1024

$ echo $?
0

$ ./exec_lines -b
./exec_lines: option requires an argument -- 'b'
Uso: ./exec_lines [-b BUF_SIZE] [-l MAX_LINE_SIZE]
Lee de la entrada estándar una secuencia de líneas conteniendo órdenes
para ser ejecutadas y lanza los procesos necesarios para ejecutar cada
línea, esperando a su terminación para ejecutar la siguiente.
-b BUF_SIZE         Tamaño del buffer de entrada 1<=BUF_SIZE<=8192
-l MAX_LINE_SIZE    Tamaño máximo de línea 16<=MAX_LINE_SIZE<=1024

$ echo $?
1

$ ./exec_lines -b 0
Error: El tamaño de buffer tiene que estar entre 1 y 8192.
./exec_lines: option requires an argument -- 'b'
Uso: ./exec_lines [-b BUF_SIZE] [-l MAX_LINE_SIZE]
Lee de la entrada estándar una secuencia de líneas conteniendo órdenes
para ser ejecutadas y lanza los procesos necesarios para ejecutar cada
línea, esperando a su terminación para ejecutar la siguiente.
-b BUF_SIZE         Tamaño del buffer de entrada 1<=BUF_SIZE<=8192
-l MAX_LINE_SIZE    Tamaño máximo de línea 16<=MAX_LINE_SIZE<=1024

$ echo $?
1
```

```
$ ./genera_bytes.py -n 129 | ./exec_lines -l 128,
Error: Tamaño de línea mayor que 128.

$ echo $?
1

$ touch test ; echo ls test | ./exec_lines
test

$ echo $?
0

$ echo -e "ls
/home/usuario/directorio/muylargo/para/generar/una/linea/enorme" |
./exec_lines -l 16
Error, línea 1 demasiado larga, se salta su ejecución: "ls
/home/usuario/directorio/muylargo/para/generar/una/linea/enorme"

$

$ touch test ; echo -e "ls test\nls test" | ./exec_lines
test
test

$ touch test ; echo -e "ls test\nls notest" | ./exec_lines
test
Error al ejecutar la línea 2. Terminación normal con código 1.

$ echo -e "ls\nevince" | ./exec_lines  # matamos evince con un kill desde
otro terminal
test
Error al ejecutar la línea 2. Terminación anormal por señal 9.

$ touch test ; echo -e "ls test > salida\ncat salida" | ./exec_lines
test

$ touch test ; echo -e "ls test >> salida\ncat salida" | ./exec_lines
test
test

$ touch test ; echo -e "ls test | wc -c" | ./exec_lines
5
```

Entrega

La entrega consistirá en un único archivo `exec_lines.c`. El archivo debe compilar conforme a los archivos `tasks.json` o `Makefile` incluidos en `ASO-Semana1.tar.xz`, es decir, con `gcc -ggdb3 -Wall -Werror -Wno-unused -std=c11 exec_lines.c -o exec_lines`. La primera línea de `exec_lines.c` debe ser la directiva para compilar conforme al estándar POSIX: `#define_POSIX_C_SOURCE 200809L`. El fichero `exec_lines.c` deberá pasar como mínimo los tests que se os indicarán más adelante.

Evaluación

Para aprobar la práctica es imprescindible que los programas pasen las pruebas de ejecución que se os proporcionarán, sin que esto signifique que no se puedan detectar otros errores con pruebas adicionales que el profesor haga durante la evaluación.

Se tendrá en cuenta la correcta indentación del código, su estructuración, el manejo de los posibles errores de las llamadas al sistema o funciones de biblioteca, así como la correcta gestión de la memoria dinámica. Otro punto importante, y por eso lo volvemos a recordar, es que se penalizará severamente el no usar bufferes para las lecturas/escrituras, es decir, leer o escribir byte a byte, pudiendo llegar a suspender la práctica, así como no contemplar lecturas o escrituras parciales si fuera necesario para la garantizar la corrección del algoritmo. Adicionalmente, las ineficiencias que se detecten en el código también pueden ser objeto de penalizaciones.