Final Laboratorio SO1920q2

Indicaciones para hacer el Examen de Laboratorio

En este examen habrá dos entregas. Una a los 45 minutos del examen (Entrega-MITAD), en la que tendrás que entregar todo lo que hayas hecho hasta ese instante, y otra al final del examen (Entrega-FINAL), que se corresponde a la entrega final del examen.

NOTA: Se considera entrega completa sólo si se han hecho las dos entregas.

Makefile (0.5 puntos)

Crea un **Makefile** que permita generar todos los programas del enunciado a la vez y cada uno de ellos por separado. Añade una regla (clean) para borrar todos los binarios y/o ficheros objeto, y dejar sólo los ficheros fuente. Los programas deben generarse si, y solo si, ha habido cambios en los ficheros fuente.

Control de errores (0.5 puntos)

Para todos los programas que se piden a continuación deben comprobarse los errores de TODAS las llamadas a sistema (excepto el write por pantalla), controlar los argumentos de entrada y definir la función Usage().

- Para todos los ejercicios: Cuando haya un error en los parámetros el programa terminará con un exit(1), cuando sea un error de llamada a sistema acabará con un exit(2)

Controla [3 puntos]

Haz un programa que llamaremos controla.c. El objetivo de este programa es ejecutar un segundo programa durante un máximo de N segundos. Nuestro programa hará lo siguiente:

- Recibirá 2 parámetros: un tiempo límite en segundos y el nombre de un comando
- Deberá evitar tratar cualquier signal que no sea SIGALRM, SIGUSR1 o SIGCHLD
- El programa creará un proceso nuevo que mutará al programa recibido como segundo parámetro. Se pide que el nuevo proceso utilice la máscara de signals bloqueados anterior a nuestras modificaciones.
- El proceso padre controlará el tiempo transcurrido usando la alarma y lo haremos en intervalos de 1 segundos. El programa debe utilizar una espera bloqueante.
- Cada vez que el proceso padre reciba un SIGUSR1, escribiremos un mensaje en la salida std indicando el tiempo que llevamos.
- El proceso padre detectará que hemos llegado al límite de tiempo o ha finalizado el proceso hijo, lo que suceda primero. Si al llegar al límite el hijo aún está ejecutándose, le enviaremos el signal SIGKILL.
- Se pide explícitamente utilizar una única función para gestionar todos los signals.
- En cualquier caso, el padre esperará a que finalice el proceso hijo.

Final Laboratorio SO1920q2

Suma_pares [6 puntos]

El programa suma_pares tiene como objetivo sumar los números que estén en las posiciones pares de un fichero de datos. Para ello, dividiremos el problema es 3 códigos:

- Programa suma.c [2 puntos]: Recibe un parámetro que corresponde con el nombre de un fichero (que puede existir o no). El programa leerá todos los números (formato int) que le lleguen por su entrada std, los sumará, y escribirá el resultado en el fichero cuyo nombre habrá recibido por parámetro. Si el fichero no existe se deberá crear y si existe asegurar que su contenido es solo el del programa. En caso de crearse el fichero, los permisos deberán ser de lectura y escritura para el usuario y de lectura para el grupo. El programa escribirá una frase "La suma es XX", donde XX es la suma de los números.

- Programa solo_pares.c [2 puntos]: Este programa actúa como un filtro y no recibe parámetros. Este programa leerá todos los números que le lleguen de su entrada std (formato int) y escribirá por su salida std (también en formato int) solamente los números que ha leído en posiciones pares, es decir, los números leídos en la posición 0,2,4,6 etc. (NO los números que sean pares, sino los que están en posición par). Se pide explícitamente utilizar un vector de 128 enteros para leer los datos que reservaremos con memoria dinámica utilizando las funciones de la librería de C. El programa deberá liberar la memoria antes de finalizar su ejecución.

    - Tened en cuenta que la entrada std puede tener más de 128 números.

    - Debéis hacer el programa teniendo en cuenta que la entrada std podría estar asociada con un fichero de datos, una pipe, la consola, etc, y que debe funcionar en cualquier caso.

- Programa suma_pares.c [2 puntos]: El programa suma_pares realizará la suma de los números en posiciones pares de un fichero utilizando los dos anteriores. Recibirá dos parámetros: el nombre del fichero con los números de entrada y el nombre del fichero donde queremos el resultado con la suma. El programa creará una pipe sin nombre y dos procesos que mutarán a solo_pares y suma respectivamente. El programa suma_pares deberá realizar las redirecciones necesarias así como pasar los parámetros necesarios a los dos procesos nuevos para que el comportamiento sea el indicado. Se pide explícitamente que el proceso padre espere a la finalización de sus hijos.

Para poder probarlo, os damos dos ficheros de test, uno con 100 números y otro con 1000 números, en los dos casos son 1's, por lo que la suma en un caso es 50 y en el otro es 500. También os damos los binarios suma y solo_pares para que podáis hacer el resto en caso que no os funcione.

Qué hay que hacer

- El Makefile
- Los códigos de los programas en C
- La función Usage() para cada programa (donde corresponda)

Qué se valora

- Que sigas las especificaciones del enunciado
- Que el uso de las llamadas al sistema sea el correcto
- Que se comprueben los errores de **todas** las llamadas al sistema
- Que el código sea claro y correctamente indentado

- Que el Makefile tenga bien definidas las dependencias y objetivos
- Que la función Usage() muestre por pantalla como debe invocarse correctamente el programa en el caso que los argumentos recibidos no sean los adecuados
- El fichero respuestas.txt

Qué hay que entregar

Un único fichero tar.gz con el código de todos los programas, el Makefile, y las respuestas en respuestas.txt:

```
tar zcvf final_lab.tar.gz Makefile *.c
```