Makefile (0,5)

Escribe un Makefile para compilar todos los programas del examen, usando reglas independientes para cada uno.

Incluye una regla llamada "clean" para eliminar todos los binarios, archivos objeto y/o ficheros temporales.

Los programas deben ser compilarse si y sólo si se han actualizado los archivos de código fuente.

Control de Errores y función Usage (0,5)

Todos los códigos fuente deben incluir el control de errores y la utilización de la función Usage().

NOTA: el fichero adjunto **lee_enteros** lee el contenido de un fichero, pasado como parámetro, de enteros en la representación interna de la máquina y escribe el contenido por pantalla (en formato leíble por los humanos). Úsalo como ayuda para el resto del examen.

mod51.c (1,25 puntos)

Escribe un programa en C llamado mod51.c que recibe un número entero como parámetro. Este programa debe mostrar por la salida estándar tantos números como los indicados en el parámetro, en la representación interna de la máquina, con valor entre 0 y 50 (módulo 51). Genera los números de forma pseudo-aleatoria utilizando:

- srand(getpid) para generar una semilla para la seria aleatoria
- rand() para generar los números.
- Después aplicadle en módulo 51. El operador módulo es %
    - valor_modulo=valor%51

No debe haber ningún separador entre cada entero que mostramos por la salida estándar

```
$> mod51 65
```

Muestra por la salida estándar 65 números (entre 0 y 50) en el formato de la representación interna de la máquina.

filtra1a10.c (3 puntos)

Crea un programa (filtra1a10.c) que lea de la entrada estándar una secuencia de números enteros en la representación interna de la máquina (como la salida del programa anterior). Este programa debe hacer.

- Crear/sobrescribir el fichero **freq1a10.int** .
- Este fichero (**freq1a10.int**) almacenará las frecuencias o número de apariciones, **valores enteros** en **representación interna de la máquina,** de los enteros que lea desde la entrada

estándar, y **cuyo valor esté entre 0 y 9**, siendo el número leído la posición en el fichero donde acumular las apariciones de ese valor leido.

- Inicializar el fichero freq.int con valor 0 desde la posición 0 hasta la posición 9.
- Cada vez que lea un número desde la entrada estándar, el programa se dirigirá a la posición en freq.int indicada por el número leído y sumará 1 al valor existente.
- El programa acaba cuando se acaben los datos en la entrada estándar.
- NO SE DEBE UTILIZAR LA MEMORIA DEL PROCESO PARA GESTIONAR ACUMULADOS PARCIALES DE LOS NUMEROS LEIDOS POR LA EE. ESTE MANTENIMIENTO DEBE GESTIONARSE DIRECTAMENTE EN EL FICHERO.

maxmin.c (3,5 puntos)

Crea el programa **maxmin.c**. Éste recibe como parámetros una lista de enteros.
Por cada entero debe generar un hijo que ejecutará **mod51** con el parámetro correspondiente.
Estos hijos deben mostrar su salida en una pipe sin nombre.
También debe generar otro hijo que ejecute el programa **filtra1a10** leyendo de la pipe y generando el correspondiente fichero **freq1a10.int**
En último lugar debe crear otro hijo que consulte el fichero freq1a10.int, una vez generado por filtra1a10 y muestre por pantalla cuál es el entero (de 1 a 10) más frecuente.
Todos los procesos hijos deben ejecutarse de forma concurrente.
El proceso padre esperará la finalización de todos los hijos.
Por ejemplo:

```
$> maxmin 5 62 8 33 49
En freq1a10.int el valor más frecuente es el 4
```

Contesta en el fichero respuestas.txt (1,25 punto)

a) (0,5 p) Indica y justifica cómo en maxmin.c sincronizas la generación del fichero freq1a10.int con la consulta de los valores más y menos frecuentes.
b) (0,25) Crea un hardlink y un softlink al fichero **maxmin.c**, llámalos **HLmaxmin.c** y **SLmaxmin.c**. Escribe los comandos para cada uno en el fichero de respuestas.
c) (0,5 p) Demuestra que **maxmin.c** y **HLmaxmin.c** son realmente hardlinks. Escribe tu respuesta y los comandos que sustenten tu argumento en el fichero de respuestas.

Qué se valora

a) Seguir las especificaciones de los ejercicios
b) Uso correcto de las llamadas al sistema
c) Control de errores en las llamadas al sistema
d) Claridad en el código e indentación adecuada
e) Que las reglas del Makefile tengan las dependencias y los objetivos requeridos correctos

f) La función Usage() muestra por pantalla, en caso de invocación incorrecta, la línea de comandos correcta para ejecutar el programa.

Qué se debe entregar.

Un único tar.gz con el código de mod51.c, filtra1a10.c, maxmin.c, Makefile, y respuestas.txt:

```
tar zcvf clab2.tar.gz mod51.c filtra1a10.c maxmin.c Makefile respuestas.txt
```