Que hay que entregar

En este examen tenéis que hacer o modificar los siguientes ficheros: Makefile, mstat_v1.c, mstat_v2.c, mstat.c, calcula.c, calcula_con_memoria.c y clab1_respuestas.txt. Entrega un único fichero .tar.gz a través del Racò. (tar zcvf clab1.tar.gz Makefile mstat_v1.c mstat_v2.c mstat.c calcula.c, calcula_con_memoria.c clab1_respuestas.txt)

Consideraciones generales (0,75 puntos)

Debes entregar un Makefile que sirva para compilar todos los códigos que generes en este examen. El Makefile debe incluir, aparte de las que tu consideres necesarias, las reglas all y clean.

Además del correcto diseño e implementación, también se valorarán:

- La robustez de vuestro código, especialmente el control de errores
- La eficiencia de vuestro código en aquellos casos que aplique.
- La usabilidad de vuestro código
- La claridad del código

Ejercicio 1: MSTAT (3 puntos)

Queremos hacer un programa que llamaremos mstat. Este programa recibirá como parámetros N nombres de ficheros (mínimo 1) y ejecutará el comando stat para cada uno de ellos (podéis hacer *man stat si quereis saber que hace*). Aquí tenéis tres ejemplos de uso: Ejem. 1: ejecución sin parámetros. Ejem.2: Ejecución con dos ficheros y Ejem. 3: ejecución cuando uno de los ficheros no existe. El texto en cursiva es la salida de stat y el texto en negrita es el que escribe el programa mstat.

Realizaremos el código por etapas. Después de los ejemplos explicamos cómo hacerlo.

```
#mstat
usage: mstat nom_fitxer1 [nom_fitxer2.. nom_fitxerN]
#mstat a.c b.c
Ejecutamos el comando stat del fichero a.c
File: `a.c'
  Size: 0               Blocks: 0          IO Block: 4096   regular empty
file
Device: 801h/2049d      Inode: 4190212     Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/  alumne)   Gid: ( 1000/  alumne)
Access: 2013-10-15 11:01:52.000000000 +0200
Modify: 2013-10-15 11:01:52.000000000 +0200
Change: 2013-10-15 11:01:52.000000000 +0200
stat ejecutado correctamente
Ejecutamos el comando stat del fichero b.c
  File: `b.c'
  Size: 0               Blocks: 0          IO Block: 4096   regular empty
file
Device: 801h/2049d      Inode: 4190214     Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/  alumne)   Gid: ( 1000/  alumne)
Access: 2013-10-15 11:01:55.000000000 +0200
Modify: 2013-10-15 11:01:55.000000000 +0200
Change: 2013-10-15 11:01:55.000000000 +0200
stat ejecutado correctamente
```

```
#mstat hola
stat: cannot stat `hola': No such file or directory
stat ha detectado un problema con el fichero
```

1.1 mstat_v1: creamos N procesos

Haz un fichero que llamaremos mstat_v1.c. En esta primera fase, el programa mstat simplemente creará N procesos (1 por cada parámetro del programa), que se ejecutarán de forma secuencial. Cada uno de los procesos hijos se encargará de procesar un parámetro del programa. En esta primera versión, simplemente escribirán un mensaje indicando que fichero les ha tocado (por ejemplo "Ejecutamos el comando stat del fichero a.c"), aunque en esta primera versión ejecutaremos el programa stat. Después de escribir el mensaje, cada proceso hijo finaliza con un exit(0);

1.2 mstat_v2: controlamos como acaban los procesos

* Copia el programa mstat_v1.c al programa mstat_v2.c y modifícalo.

Añade el control necesario para detectar como terminan los procesos hijos. Queremos saber el valor que el programador ha puesto en el exit. Si el valor detectado es 0, escribiremos el mensaje "stat ejecutado correctamente", si no es cero, escribiremos el mensaje "stat ha detectado un problema con el fichero". Prueba a cambiar el valor que ponemos en el exit para comprobar que funciona.

1.3 mstat: Mutamos a stat

* Copia el programa mstat_v2.c a mstat.c y modifícalo para que cada proceso hijo mute para ejecutar el programa stat recibiendo como parámetro el fichero que le toca a cada uno. Haz pruebas con diferentes casos para ver que todo funciona correctamente.

EJERCICIO 2: SIGNALS (2,5 puntos)

El programa calcula.c ejecuta un bucle con tantas iteraciones como indica la variable ITERACIONES_MAIN. En cada iteración, llama a la función calcula pasándole como parámetro un valor fijo (ITERACIONES_CALCULA). No hace falta que miréis la función calcula, simplemente es de relleno.

- Modifica el programa calcula.c para que las variables ITERACIONES_MAIN y ITERACIONES_CALCULA se reciban como parámetro en el programa (primer y segundo parámetro respectivamente). Añade la función usage para asegurar que se reciben. Haz varias pruebas para ver como el programa tarda más o menos en función de las variables.
- Modifica el programa para que, en caso de recibir el signal SIGUSR1 se muestre por pantalla el valor de la variable ITERACION_ACTUAL (haz los cambios que consideres necesarios).
- Modifica el programa para que reciba un tercer parámetro que será la duración máxima que queremos que tenga el programa (en segundos). Al cabo de estos segundos, se mostrara un mensaje diciendo cual es la iteración actual (valor de ITERACION_ACTUAL) y el programa termine con un exit(1). Adapta la función usage teniendo en cuenta estos cambios.

EJERCICIO 3: GESTIÓN DE MEMORIA (2,75 puntos)

Anota en el fichero clab1_respuestas.txt:

- Las regiones de memoria en las que se encuentran las variables ITERACIONES_MAIN y ITERACIONES_CALCULA .

- Para cada región de las que hayas indicado, anota el rango de direcciones de la región. Anota también como has conseguido esa información (comando, fichero, etc)

El programa calcula_con_memoria.c es una modificación del programa anterior. Ahora, la función calcula devuelve un valor. En el código que os damos, la variable ITERACIONES_MAIN tiene valor 1, por lo tanto solo necesitamos un entero para almacenar el valor de retorno. Queremos hacer una variación de este programa de forma que el número de iteraciones sea un parámetro del programa y por lo tanto el tamaño del vector dependa de ese parámetro. Modifica el fichero para incluir este cambio y utiliza las funciones de la librería de C para reservar la memoria. Incluye, antes de acabar el programa, la función necesaria para liberar la memoria reservada.

- Modifica también el programa para obtener la dirección límite del heap antes y después de hacer la reserva y escribe por pantalla el tamaño que se ha reservado.

EJERCICIO 4: Utilización del sistema (1 punto)

Anota en fichero clab1_respuestas.txt que comando deberías utilizar para mostrar todos los procesos que se están ejecutando ordenado por el tamaño de memoria que ocupan. Comprueba que es correcto utilizando el código anterior, ejecutando varias instancias a la vez con parámetros diferentes. Copia en el fichero clab1_respuestas.txt no solo el comando sino el resultado de las primeras líneas que muestra el comando.