Control Laboratorio 2 2012QT

0) **(0,5 puntos)** Control de errores y utilización de la función Usage en todos los códigos.

1) **(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 solo los ficheros fuentes. Los programas deben generarse si, y solo si, ha habido cambios en los ficheros fuentes.

2) (1.5 puntos) Haz un programa (que llamaremos **lee_num**) que sea capaz de leer un número de la entrada std. El número estará en ascii seguido de un carácter '\n'. El programa leerá el numero (asumiremos que habrá solo 1), lo convertirá a formato int (representación interna de la maquina) y lo escribirá por una pipe con nombre. Escribe también el número que ha leído por la salida de error std para comprobar que funciona correctamente.
Asumiremos que la pipe con nombre existe previamente.

Ejemplo:

```
$echo "5" | ./lee_num
He leido el numero 5
```

3) (1.5 puntos) Haz un programa (que llamaremos **cuantos_running**) que constará de un bucle infinito que irá leyendo de una pipe con nombre, por la cual le llegaran números en representación interna de la maquina. Cada vez que lea un número, lo comparará con el último número leído. Si el número es diferente, se generará un nuevo fichero cuyo contenido sea la siguiente frase: "Antes habían X programas running y ahora hay Y\n", donde X e Y corresponden al número antiguo y nuevo respectivamente.
El nombre del fichero será diferente para cada comparación, para ello concatenaremos al texto "RUN." un número que irá aumentando de forma secuencial (RUN.1, RUN.2, etc). El programa terminará cuando el número recibido sea -1.
El programa debe crear la pipe con nombre al inicio. Al iniciar el programa asumiremos que el valor del último número es 0. Ya que los nombres de ficheros se reúsan de una ejecución a la siguiente, al crear los ficheros se debe truncar su contenido.

NOTA: Para que el programa no termine cuando los procesos que escriban los números cierren la pipe, ábrela en modo lectura y escritura

4) **(3 puntos)** Haz un programa que detecte cuantos procesos están en estado R (RUN). El programa debe hacer lo mismo que haría la shell al hacer la siguiente secuencia:

```
$ps -A r  | wc -l
```

Llama al programa **controla_run**. Este programa, se utiliza juntamente con los anteriores, es decir, tendríamos que poder hacer:

```
$cuantos_running &
$controla_run | lee_num
$fib 100& (podemos coger este de la S6 a modo de ejemplo)
$controla_run | lee_num
```

```
$echo "-1" | lee_num
```

Y tendríamos que poder comprobar que se ha detectado el cambio en el número de procesos en estado R con los diferentes ficheros

5) **(2 puntos)** Haz un programa que extraiga la información de un fichero (de nombre fijo FAT.dat) que sabemos que contiene un vector encadenado. Dado un número de entrada, esa entrada contiene un número que apunta a otra entrada, formando una cadena hasta que una entrada contenga el valor -1 que indica que se ha terminado. Los números están almacenados en formato int. Llamaremos a programa **FAT** y el formato será:

```
$FAT primera_entrada
```

Donde primera entrada es el número de la primera entrada de la cadena. Los valores de las entradas encadenadas se han de mostrar por la salida std en formato ascii

Haz las siguientes pruebas y deberías obtener los siguientes resultados

```
$ FAT 0
3 4 5
$ FAT 1
6
$ FAT 2
7 8 9 10 11
```

6) **(1 punto)** El programa SumaVector_pth.c (similar al de la S10), calcula la suma de los elementos de un vector. En este caso, hemos transformado el código para que ya incluya la definición de tareas y la creación de threads. Sin embargo, hemos visto que hay un problema durante la ejecución que parece ser debido a una race condition. Analiza el código para ver si hay algún problema, y soluciónalo utilizando las llamadas a la librería de pthreads. Recuerda que a veces el problema existe pero no se detecta fácilmente.

Qué se tiene que hacer

- El makefile
- Los códigos de los programas en C
- La función Usage() para cada programa
- El fichero respuestas.txt con las respuestas a las preguntas

Qué se valora

- Que sigas las especificaciones del enunciado
- Que el uso de las llamadas a sistema sea el correcto
- Que se comprueben los errores de **todas** las llamadas al sistema
- Código claro y correctamente indentado
- Que el makefile tenga bien definidas las dependencias y los objetivos
- La función Usage() que muestre en pantalla cómo debe invocarse correctamente al programa en caso que los argumentos recibidos no sean adecuados.
- El fichero respuestas.txt

Qué hay que entregar

- Un único fichero tar con el código de *.c, respuestas.txt y el Makefile

- tar zcvf control-L2.tar.gz *.c respuestas.txt Makefile