Control Laboratorio SO2021q1

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().

Ejercicio 1 (4.5 puntos)

[4 puntos] Escribe un programa (actualizar_fecha.c) que reciba una secuencia de nombres de fichero como argumento. El programa debe crear tantos procesos hijo como nombres de fichero haya recibido, de forma **concurrente**. Cada proceso hijo debe mutar al programa "touch" pasandole como argumento uno de los nombres de fichero recibidos (Nota: El programa "touch" actualiza la fecha de un fichero a la fecha actual y además crea el fichero en el caso que no exista). El proceso padre debe esperar a cada uno de los hijos a medida que vayan acabando, sin un orden preestablecido. Después de la muerte de cada hijo, el proceso padre mostrará el mensaje "Fecha F actualizada por P", donde F es el nombre del fichero que ha tratado ese hijo y P es el identificador de proceso (PID) del hijo. Podemos considerar que como máximo nos pasaran 5 nombres de fichero en la llamada.

[0.5 puntos] ¿En qué orden se van a escribir los mensajes? ¿Podemos garantizar que será siempre el mismo? Justifica tu respuesta en un fichero de texto llamado "respuestas.txt".

Ejercicio 2 (4.5 puntos)

Escribe un programa (signals.c) que cree N procesos hijo de forma concurrente, donde N es el primer y único argumento del programa (N es un entero, $0 < N \leq 10$). Cada hijo debe esperar 3 segundos, mostrar el mensaje "X: Timeout" (donde X es el PID del proceso) y acabar. Si antes de que pasen los 3 segundos el proceso hijo recibe un SIGUSR1, el hijo debe acabar retornando, en la llamada exit, el número de segundos que restaban para que expirara su temporizador (ver man alarm).

El proceso padre después de crear los N procesos hijos enviará un SIGUSR1 a todos sus hijos que tengan un PID par. Antes de acabar, el proceso padre esperará a todos los hijos y, para aquellos hijos con PID par, debe mostrar por pantalla el mensaje "Hijo X. Segundos restantes Y", donde X es el PID del hijo (par) e Y son los segundos restantes que ha devuelto a través de exit. Todas las esperas deben ser bloqueantes y el programa debe controlar el caso que los signals lleguen antes de la espera.

Qué hay que hacer

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

Control Laboratorio SO2021q1

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 clab.tar.gz Makefile respuestas.txt *.c
```