Control Laboratorio 1 SO1920q1

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().

Procesos

A) **[2.5 puntos]** Escribe un programa (status.c) que reciba una lista de PIDs como argumento. El programa debe mostrar por pantalla el estado actual de estos procesos (p.ej. Running, Sleeping, Zombie, etc.). Para hacerlo, el programa debe crear de forma concurrente tantos hijos como PIDs haya recibido, y cada hijo debe mutar al comando "grep State /proc/PID/status", dónde PID es uno de los PIDs pasados como argumento. (Pista: para construir el string "/proc/PID/status" puedes utilizar la función sprintf).

B) **[2 puntos]** Modifica el programa anterior (llámalo status2.c) para que si alguno de los PIDs pasados como argumento no existe, el programa muestre el mensaje "Proceso PID no existe" (dónde PID es el PID del proceso que no existe) y acabe directamente sin esperar al resto de hijos. Para saber si un proceso existe, podemos recoger el valor que el programa "grep" devuelve en la llamada exit(). Si devuelve un valor diferente de 0 indica que ha habido un error en la ejecución del comando. Para este apartado podemos considerar que como máximo el programa recibirá 10 PIDs como argumento.

C) **[0.5 puntos]** ¿En el apartado A, podemos garantizar en que orden se van escribir los mensajes de cada uno de los hijos? Justifica tu respuesta en el fichero "respuestas.txt".

D) **[0.5 puntos]** Abre una terminal y ejecuta un comando que te permita descubrir el PID del proceso que ejecuta ese intérprete de comandos. Ahora ejecuta el programa "status" desde esa misma terminal pasándole como argumento su PID (obtenido con el comando anterior). Indica en el fichero "respuestas.txt" en que estado está el proceso que ejecuta el intérprete de comandos y razona por que motivos se encuentra en ese estado determinado.

Signals

E) **[3.5 puntos]** Escribe un programa (signals.c) que cree N procesos hijo, dónde N es el primero y único argumento del programa. Entre la creación de un hijo y el siguiente, el padre esperará a la recepción de un SIGUSR1 utilizando una espera bloqueante. Los procesos hijo deben mostrar por pantalla un mensaje con su PID y esperar 1 segundo con una espera activa. Al cabo de 1 segundo, el proceso debe mandar un SIGUSR1 a su padre y acabar. El padre debe controlar el caso en que el SIGUSR1 llegue antes de realizar la espera.

Control Laboratorio 1 SO1920q1

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

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 clab1.tar.gz Makefile respuestas.txt *.c
```