Examen Final SO1718q2

Makefile (0,5 puntos)

Cread un **Makefile** que permita generar todos los programas del enunciado a la vez y cada uno de ellos por separado. Añadid una regla (clean) para borrar todos los binarios y/o ficheros objeto y dejar solo los ficheros fuentes. Los programas deben generarse si, y sólo si, ha habido cambios en los ficheros fuentes.

E/S y procesos (4,5 puntos)

A) **[2,5 puntos]** Implementad un programa (llamadlo **exam1.c**) que espera dos parámetros: el nombre del fichero de entrada y el nombre de fichero donde dejar el resultado (si este fichero no existe, debe crearse con permisos `rw-r----`; si existe, su contenido debe borrarse). El código debe crear un proceso hijo que, tras redireccionar los canales estándar a los ficheros recibidos como parámetros, ejecutará el programa `desc_ok` (un descompresor de ficheros comprimidos del que os facilitamos el ejecutable). El proceso padre finalizará al detectar la muerte del hijo.

   Nota: Los ficheros descomprimibles con `desc_ok` tienen extensión `.Q`.

B) **[1 punto]** Modificad el programa anterior (llamadlo **exam2.c**) para que sea el proceso padre quien escriba los datos en el fichero de salida. Para transmitir la información entre hijo y padre debéis utilizar una pipe sin nombre. El proceso padre debe ir escribiendo los datos en el fichero de salida a medida que `desc_ok` los escribe en la pipe.

C) **[1 punto]** Modificad el programa exam1.c (llamadlo **exam3.c**) para que muestre por el canal de error el tamaño en bytes del fichero sin comprimir, el del fichero comprimido y la relación entre sus tamaños (tamaño_comprimido/tamaño_original). Se valorará la eficiencia en la forma como obtenéis la información.

Signals (3 puntos)

[3 puntos] Escribid un programa (llamadlo **signals.c**) que espere un único parámetro en el vector de argumentos, que se interpretará como un número entero N. El programa deberá mostrar un mensaje cada segundo; al llegar a N segundos, finalizará. Además tratará los siguientes signals:

- SIGUSR1: debe actualizar un contador de signals SIGUSR1 recibidos. Si este contador llega al valor 3, el programa finalizará inmediatamente.
- SIGUSR2: debe provocar la finalización inmediata del proceso.
- El resto de signals estarán bloqueados.

Las rutinas de atención a SIGALRM, SIGUSR1, y SIGUSR2 deben escribir un mensaje indicando que entran en ejecución.

El programa **no** debe realizar ningún tipo de espera activa.

Disponéis del ejecutable signals_ok que se comporta correctamente.

Examen Final SO1718q2

Preguntas (2 puntos)

Contestad en el fichero respuestas.txt a las siguientes preguntas:

A) **[0,25 puntos]** ¿Cómo ejecutarías exam1.c para que el resultado de la descompresión aparezca por pantalla?

B) **[0,25 puntos]** ¿En exam2.c, por qué es conveniente que el proceso padre escriba los datos en el fichero de salida a medida que `desc_ok` los escribe en la pipe? (la alternativa sería que el proceso padre esperase la muerte de `desc_ok` para empezar a leer de la pipe y escribir los datos en el fichero de salida).

C) **[0,5 puntos]** ¿Cómo determinaríais el rango de direcciones de pila que tiene asignadas un proceso?

D) **[0,5 puntos]** Sean M y S dos versiones de un mismo programa; la única diferencia entre ellas es que M solicita memoria dinámica utilizando `malloc()` y S utilizado `sbrk()`. Si ejecutamos M y S con los mismos datos de entrada, ¿es posible que el tamaño del heap del proceso que ejecuta M acabe siendo mayor que el tamaño del heap del proceso que ejecuta S?

E) **[0,5 puntos]** Dado un fichero llamado file.txt, indicad las órdenes que permiten crear un softlink y un hardlink a file.txt. ¿Cuántos inodes estarán asignados a estos tres ficheros (file.txt, el hardlink y el softlink)?

Qué se tiene que hacer

- El Makefile
- El código de los programas en C con la función Usage() en cada programa
- El fichero respuestas.txt con las respuestas a las preguntas

Qué se valora

- Que sigáis las especificaciones del enunciado.
- Que el uso de las llamadas a sistema sea el correcto y 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 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.gz con exam*.c, Makefile, y las respuestas en respuestas.txt:

```
tar zcvf final.tar.gz Makefile respuestas.txt exam*.c signals.c
```