PRÁCTICA 5:
Llamadas al sistema relacionadas con procesos (II).

Objetivo:

Implementación de programas en C bajo UNIX, que utilicen las siguientes llamadas al sistema relacionadas con procesos y señales:

fork	***execl***	***execv***	***alarm***
getpid	***getuid***	***signal***	***kill***
pause	***sleep***	***exit***	***wait***

Enunciados:

Grupo A

Realizar un programa que funcione como un *PLANIFICADOR*, tomando como parámetro el número de procesos. El programa lanzará un hijo para cada proceso, realizando su ejecución mediante un algoritmo de turno circular con un quantum de 1 segundo. Los procesos hijo se abortan con <CTRL+C> (señal *SIGINT*).

Grupo B

Realizar un programa en el que un proceso padre cree dos hijos. El primer hijo funciona de tal forma que cuando el operador teclee <CTRL+\> (señal *SIGQUIT*), el hijo captura la señal y termina con estado 21, mostrando el padre en la salida estándar el estado de terminación del hijo. En el caso de que el operador teclee <CTRL+C> (señal *SIGINT*), el hijo no captura la señal, y ésta lo mata, mostrando el padre la señal que terminó con el hijo.

La señal <CTRL+\> debe ser ignorada por el segundo hijo y por el padre.

El padre debe capturar <CTRL+C>, guardando en un acumulador el número de veces que se ha pulsado <CTRL+C>. Cuando el número de veces sea de 3, el padre envía una señal *SIGKILL* al segundo hijo para matarlo.

Grupo C

Realizar un programa al que se le pase como primer parámetro en la línea de comandos el nombre de un fichero de eventos *cron_even* que tiene líneas de la forma: *comando hora*.

El programa debe crear un proceso hijo que trabaja en un bucle infinito, despertándose cada segundo, y ejecutando los eventos cuya hora haya cumplido en el último segundo.

La salida estándar de todos los comandos debe redirigirse a un fichero de contabilidad del sistema denominado *mensa_cron*.

El proceso padre debe retornar sin esperar al proceso hijo, dejando activada una señal para abortar al proceso hijo mediante una combinación de teclas.

Grupo D

Realizar un programa en el que un proceso padre crea dos hijos que ejecutan un bucle ocioso en el que cada 2 segundos muestran un mensaje identificativo.

- Si el hijo 1 recibe la señal *SIGQUIT* debe terminar con estado 21.
- El padre y el hijo 2 ignoran esta señal.
- Si el padre recibe la señal *SIGINT*, envía una señal *SIGKILL* al hijo 2.
- Los hijos no capturan *SIGINT*, con lo que esta señal los mata.

El padre debe informar del estado de terminación de sus hijos (si han terminado con exit(n), debe mostrar el valor de n por pantalla).

Grupo E

Realizar un programa que ejecute un bucle ocioso en el que, cada dos segundos, muestra un mensaje identificativo (p.e. "Proceso padre"). Al recibir la señal *SIGINT*, el programa crea un proceso hijo y espera por él. El resto de señales *SIGINT* serán ignoradas tanto por el padre como por el hijo. El proceso hijo permanece ejecutando un bucle ocioso en el que cada dos segundos muestra un mensaje identificativo (p.e. "Proceso hijo"). Al cabo de 10 segundos, el proceso padre enviará la señal *SIGKILL* al proceso hijo, terminando ambos.

Grupo F

Realizar un programa que admite como parámetro un numero *n*. El programa debe:

- Crear un hijo que ejecute un bucle ocioso en el que muestre, cada segundo, un mensaje identificativo (p.e. "Proceso hijo").
- El padre debe capturar la señal *SIGINT* (guardando en un acumulador el numero de veces que se ha pulsado CTRL+C).
- Cuando el numero de veces sea *n* (primer parámetro), el padre envía una señal *SIGKILL* al proceso hijo y termina (es decir, ambos procesos terminan de forma simultánea).

Grupo G

Realizar un programa que admite como parámetro el nombre de un fichero de texto existente. El programa debe realizar las siguientes funciones:

- Ejecutar un bucle ocioso en el que cada 2 segundos muestra un mensaje identificativo por pantalla.
- Al recibir la señal *SIGINT*, crea un hijo y el padre espera por él.

- El hijo escribe en pantalla la primera letra de cada línea del fichero y termina con exit(n), donde n es el número de caracteres del fichero.
- El padre muestra en pantalla el valor de n y termina.

Grupo H

Realizar un programa que ejecute un bucle ocioso en el que cada 2 segundos se muestra un mensaje identificativo (p.e. “Proceso padre”).

- Al recibir una señal *SIGINT*, crea un hijo y espera por él (el resto de señales *SIGINT* serán ignoradas tanto por el padre como por el hijo).
- El hijo muestra un mensaje identificativo por pantalla (p.e. “Proceso hijo”) y a continuación envía una señal *SIGUSR1* al padre.
- Cuando el padre recibe la señal del hijo, lo indica en pantalla y ambos procesos deben terminar en ese momento de forma simultánea.

Grupo I

Realizar un programa que acepte como parámetros el nombre de un fichero de texto existente y un número de línea (*numlinea*).

- El programa crea un hijo y espera por él.
- El hijo muestra en pantalla la línea del fichero de texto indicada por *numlinea*, y entra en un bucle ocioso.
- Si el hijo recibe la señal *SIGQUIT*, termina con exit(n), donde n es el número de caracteres de la línea.
- Si el hijo recibe la señal *SIGINT*, termina por acción de la señal.
- El padre ignora ambas señales (*SIGINT* y *SIGQUIT*), y termina informando sobre cómo terminó el hijo: por la señal o por el estado de exit (en cuyo caso se mostrará por pantalla el valor de n).

Grupo J

Realizar un programa al que se le pase como parámetro una serie de comandos sin argumentos, por ejemplo: ls, pwd, ps, etc. El programa debe lanzar tantos procesos hijo como sea necesario para ejecutar cada comando, redirigiendo su salida a un fichero que tiene el mismo nombre que el comando ejecutado y la extensión ".sal". Los procesos deben sincronizarse entre sí mediante el envío de señales para evitar la ejecución simultánea de mas de dos procesos.