# PRÁCTICAS – TEMA 1

## Ejercicios

1. Prueba a ejecutar el comando ps en tu sistema:
   1. ¿Qué procesos se están ejecutando?
   2. ¿Qué información se obtiene con el comando ps?
   3. ¿Y con ps-f?
   4. ¿Y con ps-AF?
2. ¿Qué información se obtiene con el comando tasklist? ¿Cómo se pueden ver los procesos en Windows desde una interfaz?
3. Haz un programa que cree dos procesos
   1. El proceso padre ha de mostrar el contenido de la carpeta /run
   2. El proceso hijo ha de mostrar el contenido del escritorio.

El output de estos procesos se ha de escribir en un archivo llamado salidaEjercicioTres.txt

Haz una versión en la que el orden en el que se escriba el output sea aleatorio y otra en la que se pueda ver antes el contenido del escritorio que el de la carpeta /run.

1. Realizar una aplicación que cree un proceso hijo, donde se definirá una variable entera de valor 10. El padre incrementará dicho valor en 2 y el hijo restará 2. Antes de ejecutarla, ¿Qué valores crees que obtendrás? Desarrolla el código e indica el resultado obtenido.
2. Realizar una aplicación que cree un proceso hijo y uno nieto, donde se definirá una variable entera de valor 10. Todos los procesos restan 2 de esa variable, realizándose la resta siempre DESPUÉS de la llamada a fork.

¿Qué valores crees que obtendrás? Mostrar los resultados de las operaciones por pantalla.

Ahora modifica ligeramente el ejercicio haciendo que la resta sea siempre ANTES de la llamada a fork. ¿Qué valores crees que obtendrás? Mostrar los resultados de las operaciones por pantalla.

1. Ejecuta el ejemplo 6, explica lo que está pasando y mata al proceso hijo.
2. Ejecuta la versión 1 del ejemplo 7.
   1. Abre un terminal a parte y crea un proceso huérfano.
   2. Crea un proceso zombie y muéstralo.
3. Ejecuta la versión 2 del ejemplo 7 y explica por qué no se generan procesos huérfanos.
4. Utilizando el esquema que se muestra a continuación:

Imagen en blanco y negro de un reloj en la pared

Descripción generada automáticamente con confianza baja

Es importante que no quede ningún proceso huérfano.

Podéis comprobar que el árbol se crea y se destruye correctamente si obtenéis una salida similar a la siguiente (utilizar de manera orientativa):

Texto

Descripción generada automáticamente

1. Supón un programa que represente un proceso que crea a dos hijos y tiene la siguiente salida:
   * 1 envía mensaje a 2
     + 2 recibe mensaje de 1
     + 2 envía mensaje a 1
   * 1 recibe mensaje de 2
   * 1 envía mensaje a 3
     + 3 recibe mensaje de 1
     + 3 envía menaje a 1
   * 1 recibe mensaje de 3

Los mensajes se enviarán usando la técnica aprendida.

Cierra los descriptores cuando no sean necesarios.

1. Se utilizando el ejercicio 9, se ha de realizar lo siguiente:

* Cada proceso además de ser identificado por un número tendrá un valor.
* Los procesos hijos han de pasarle a su padre su valor, el padre ha de recoger estos valores y sumarlos al suyo.
* Este proceso se ha de repetir hasta que se alcance el proceso de arriba, que tiene que contener la suma del valor de todos los procesos.
* Los procesos solo pueden leer cuando han creado a todos sus hijos.
* Los descriptores solo han de abrirse cuando se necesiten y se han de cerrar tan pronto como no sean necesarios en cada proceso. Esta gestión de recursos será parte de la evaluación.
* Inicialmente los procesos solo pueden conocer su valor.

Imagen en blanco y negro

Descripción generada automáticamente con confianza media

La salida puede gestionarse de la siguiente manera (utilizar de forma orientativa).

Texto

Descripción generada automáticamente

Elementos de ayuda:

* Hacer cast de int a cadena de caracteres: *sprintf (variableTexto, “%d”, variableEntera);*
* Hacer cast de cadena de caracteres a i*nt: variableEntera = atoi(variableTexto);*
* Cuando, por ejemplo, habéis leído un mensaje de 2 caracteres y queréis leer uno de uno:
  + Proc 1 manda 14
  + Proc 2 lee 14
  + Proc 1 manda 2
  + Proc 2 lee 24

Para que no ocirra esto hay que limpiar el buffer (volver a ponerlo como cadena vacía) (*memset(buffer, 0, sizeof(buffer));*)

Este ejercicio es largo, y si no se hace de manera ordenada puede resultar complicado, para que resulte más sencillo se recomienda:

* Primero realizar el ejercicio solo con los procesos 1,2,3 y 4.
* Se permite crear un pipe para cada comunicación entre procesos, esto facilitará su uso, no obstante, es obligatorio abrirlos y cerrarlos de manera eficaz y correcta.
* Para cada proceso:
  + Asignar el valor.
  + Indicar el proceso y su valor.
  + Crear los pipes y procesos necesarios. (si procede)
  + Leer los valores enviados por los hijos (si procede)
  + Escribir los valores calculados (si procede)
  + Morir indicando su valor.

1. Lanza el ejemplo 10 de lectura ¿Qué sucede?

Abre otra terminal y lanza el ejemplo 10 de escritura ¿Qué sucede ahora?

¿Qué pasa si vuelves a lanzar el ejemplo de escritura?

¿y si sales del ejemplo de lectura y lo vuelves a lanzar?

¿y si lanzas el ejemplo de escritura sin que esté abierto el de lectura?

1. Vuelve a hacer el ejercicio 11 pero con FIFOS, la información se ha de enviar directamente al proceso 1 sin que esta pase por el resto de los procesos intermedios.
2. Crea un programa en el que haya un proceso padre y otro hijo.

El padre cuando reciba la señal SIGUSR1 imprimirá: Tengo tierras

El padre cuando reciba la señal SIGUSR2 imprimirá: Quiero ver obras

El hijo cuando reciba la señal SIGUSR1 imprimirá: Agua bendita

El hijo cuando reciba la señal SIGUSR2 imprimirá: Desinstala el lol

El hijo ha de enviar ambas señales al padre y viceversa, de tal manera que se muestren los mensajes.

El padre esperará a que el hijo le mande la señal sigterm para morir.

1. Crea un programa que simule un juego de ping pong con las siguientes características.

* Un padre creará dos procesos hijos.
* Estos dos procesos han de comunicarse mediante señales.
* Un proceso imprimirá PING y el otro PONG.
* Ha de imprimirse PING PONG 10 veces.
* El padre ha de permanecer vivo mientras dure esta comunicación.
* Cuando se imprima PING PONG 10 veces los procesos hijos morirán, pero no sin antes enviar al padre la señal SIGTERM para que muera.

Para que este ejercicio funcione bien es necesario imprimir los valores PING y PONG en un fichero ya que la salida por consola no funciona correctamente al usar señales.