

# Sistemas Operativos

[Inicio](#)   [Teoría](#) ∨   [Prácticas](#)   [Notebooks](#)   [Exámenes](#) ∨   [SO – Blog](#)   [Logout](#)   

[Inicio](#) » [PRACTICAS](#) » [C Prácticas](#) » Esqueleto Práctica Parte C – Señales 2

## Esqueleto Práctica Parte C – Señales 2

[C Prácticas](#)



### Señales versión 2

Realmente la versión 1 no es una versión que pueda quedar como final por un problema de sincronización generada por el uso de **pause()** como explico en el ejemplo de [Comunicación padre-hijos con señales](#).

Así que os propongo una ligera modificación en esta versión. Esta modificación es tan sutil que parece la misma versión y podréis reutilizar casi todo.

Ahora se trata de que los hijos no esperen una señal del padre para avanzar y ponerse a leer la combinación ganadora. Como explico en el [artículo de Pipes](#), la lectura de un Pipe cuando éste no tiene información para ser leída provoca el bloqueo del lector. Por tanto vamos a usar este hecho para sincronizar al padre con los hijos.

Cuando los hijos generen su apuesta, entonces se ponen a leer directamente la combinación ganadora del pipe. El padre no generará la combinación ganadora hasta que todos los hijos le hayan enviado la señal indicando que ya han apostado. Por tanto los hijos se esperarán en el pipe hasta que el padre envíe la combinación ganadora por el mismo.

Respecto a la versión 1, simplemente tendremos que eliminar el código relativo a la señal que el padre envía para que los hijos se pongan a leer la combinación ganadora.

Os dejo únicamente el esqueleto del fichero main.c, que es el único que cambia respecto a la versión 1.

Recordar que deberéis hacer un [makefile](#) para compilarlos.

## main.c

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <string.h>
#include <signal.h>
#include <ucontext.h>
#include <sys/types.h>
#include "funciones.h"

// DEFINES Y CONSTANTES GLOBALES

int main(int argc, char *argv[]){
//Se define la estructura para el manejador de las señales que recibe el padre
struct sigaction manejador;

//Se inicializa el contador de mensajes recibidos, que han sido enviados por los hijos al haber realizado su apuesta
manejador.sa_sigaction = ; //Nombre de la funcion manejadora
manejador.sa_flags = SA_SIGINFO; //Esto lo definimos para enviar informacion adicional junto a la señal
sigaction(SIGRTMIN, &manejador , NULL); //Establecemos el manejador de la señal a la función manejadora

//Declaración e inicialización de otras variables globales

...

//Creamos el pipe o los pipes y controlamos que se haya creado correctamente.
//Determinar el protocolo a usar tanto si empleamos un único pipe para todos los hijos o un pipe por hijo
```

```
//Creamos las estructutras hijos
HIJO hijos[numHijos]; // Declaramos el vector de estructuras HIJO

for(i=0; i<numHijos;i++){ //Inicializamos las estructuras de hijos
    hijos[i].pid=0;
    hijos[i].num=0;
    hijos[i].premio=0;
    //Añadimos el pipe correspondiente al hijo a la estructura si hemos decidido crear un pipe por hijo
}

//Creamos el bucle principal donde se llama a fork en cada iteración del padre.
//Los hijos hacen exit en su código de forma que no iteran.
for(i=0;i<numHijos;i++){
    //Código del padre
    //mantenimiento de la estructura de hijos

    //Código del hijo
    //Generamos la apuesta del hijo.

    //Comunicamos al padre que hemos apostado

    //Leemos del Pipe, aquí se bloquea el hijo si todavía no está la combinación ganadora en el

    //Calculamos los aciertos

    //Salimos comunicando aciertos
    exit(aciertos);
}
//... este código sólo lo ejecuta el padre tras todas las iteraciones...

//Esperamos a que todos los hijos apuesten

//Generamos la combinacion ganadora

//Escribimos la combinación ganadora en el/los pipes

//Esperamos a la terminación de todos los hijos en un bucle con pause()
//Obenenemos el valor de retorno de cada hijo y calculamos el importe de su premio

//Generamos el fichero de resultados del sorteo en curso y salimos
}
```

## Deja un comentario

Conectado como alu28. [Edita tu perfil.](#) ¿Salir? Los campos obligatorios están marcados con \*

Escribe aquí...

Publicar comentario »

Copyright © 2025 Sistemas Operativos  
Escuela Politécnica Superior de Elche  
Universidad Miguel Hernández  
Miguel Onofre Martínez Rach