Esqueleto Práctica Parte C – Semáforos y Memoria Compartida

C Prácticas

Versión con Semáforos y Memoria Compartida

Modificamos levemente el esqueleto de la versión con semáforos para introducir memoria compartida en el acceso al array de la combinación ganadora, ya no lo haremos por un pipe sino con memoria compartida entre el padre y los hijos.

Podéis ver un ejemplo en Sincronización Padre-Hijos con semáforos y memoria compartida. **Aquí, como ya no utilizamos el pipe donde los hijos se quedaban esperando, es necesario sincronizar con semáforos a los hijos para que esperen por la combinación ganadora.**

Respecto a la versión 3, tendremos que añadir el código de la creación y vinculación de la memoria compartida y añadir el código necesario para el semáforo que para a los hijos a la espera de que el padre genere la combinación. El padre debe levantar el semáforo para que los hijos lean la combinación ganadora de la memoria compartida.

Os dejo únicamente el esqueleto del ficheros main.c puesto que el código de funciones.c, funciones.h, sem.h y sem.c podéis obtenerlo de Semáforos o de Sincronización Padre-Hijos con semáforos.

Recordar que deberéis hacer un makefile para compilarlos.

main.c

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <signal.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include "funciones.h"
#include "sem.h"

// DEFINES Y CONSTANTES GLOBALES



int main(int argc, char *argv[]){

//Declaración e inicialización de los semáforos usados para la sincronización,
//un semáforo que actuará como barrera compartida entre padre-hijos.

//Declararmos el array de semáforos para que los hijos se cuelguen a la espera de que el padre
//genere la combinación ganadora. Los inicializamos a cero para que el wait de los hijos los bloquee


//Declaración e inicialización de otras variables globales

...


//Creamos las estrucutras 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;
}
```

```
//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.

  //Señalizamos la barrera que compartimos con el padre

  //Hacemos wait sobre el semáforo que nos corresponde del array para esperar a poder leer de memoria compartida.

  //Leemos de la memoria compartida

  //Calculamos los aciertos

  //Nos desvinculamos de la memoria compartida (opcional al terminar el S.O. lo desvinculará)

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

//Esperamos a que todos los hijos lleguen a su barrera, lo que indica que han apostado

//Generamos la combinacion ganadora

//Escribimos la combinación ganadora en la memoria compartida

//Siñalizamos (signal) en el array de semáforos para desbloquear a los hijos y que lean de la memoria.

//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
}
```

← Entrada anterior

Entrada siguiente →

Deja un comentario

Conectado como alu28. Edita tu perfil. ¿Salir? Los campos obligatorios están marcados con *

Escribe aquí...

Publicar comentario »

Universidad Miguel Hernández

Miguel Onofre Martínez Rach