MEMORIA PRACTICA 2 (PARKING)

En esta segunda práctica que se nos pedía realizar, teníamos que realizar un parking con entradas y salidas gestionadas automáticamente mediante procesos ligeros (threads).

La realización la esquematizamos en 3 partes: main, función coche y función camiones.

Como introducción hemos incluido las librerías necesarias, incluida la pthread que era nueva para esta parte y 3 constantes (faltando la realización del apartado que pedía intrucirlas mediante argumentos, no me ha dado tiempo ni a pensarlo por el trabajo) que son plazas del parking, coches y camiones disponibles para entrar a aparcar.

Tenemos una variable que es un array de enteros, para que en casa posición se ponga el número/camión que este aparcado en su plaza. Un mutex para que bloquee el recurso compartido del parking cuando este aparcando algún vehículo ya que no permitimos la entrada de varios vehículos a la vez para no provocar conflictos, cuando algún vehiculo este saliendo también lo bloqueamos pero lo explicaremos mas adelante. Tenemos también una variable ocupación para que nos diga cuantas plazas ocupadas hay en el parking y un variable de condición para el mutex que nos será muy útil luego como ya veremos.

MAIN

Esta parte es muy sencilla, con una variable i para la inicialización de las diferentes variables que necesitemos, y con dos arrays de enteros unos para coches y otro para camiones para darle número a cada uno de ellos e inicializamos el mutex con la variable que queremos y la condición para el mutex. Inicializamos los diferentes arrays declarados. El de plazas a 0 (con esto queremos decir que están libres y nos están ocupadas por nadie. Los arrays para vehículos los nombramos de 1 a número de coches para el array de coches y de 101 a 100+nº camiones para los camiones tomando copia de como venía en el enunciado de la práctica. Con pthread\_t creamos las variables thread para cada vehículo y con los 2 ultimos for los creamos pasándole en cada caso los valores correspondientes (pid de cada thread, la función, y el número del vehículo), no es necesario esperar por cada thread con join ya que nunca saldrá de la función.

FCOCHE y FCAMION

Las explicare a la par ya que son similares aunque necesita realizarse en 2 funciones distintas. Lo primero una variable tipo entero con el número del vehículo la cual obtenemos mediante un casting explicado en clase el cual pasamos de un puntero a void a un puntero a entero (creo que era asi), una variable entera i y un variable entera para las plazas libres para mayor comodidad a la hora de programar, aunque no es necesaria. A continuación entramos en el bucle while (1), bucle que se repite indefinidamente. Dentro del bucle comenzamos inicializando la variable i a 0 (es necesario hacerlo aquí y no antes para que cada vez que empieza la tengamos a 0) y ponemos el vehículo a dormir un tiempo aleatorio. Bloqueamos en mutex y con el while miramos si el parking está lleno para los coches y para los camiones si está lleno o solo tiene un plaza libre (ya que el camión necesita 2 plazas libres para aparcar) y ponemos que espere por la condición para que cuando este libre vuelva a activarse. Comprobamos la primera plaza libre y para los camiones ponemos una condición para los camiones de que no sea la ultima plaza y la siguiente este libre que durara para todo el proceso, si no se cumple desbloquea el mutex y vuelve a empezar, si se cumple entramos en el if. Asignamos a esa plaza el numero del vehiculo (y a la siguiente en el caso de los camiones). La ocupación la incrementamos (en 1 para los coches y en 2 para los camiones). Calculamos las plazas libres y mostramos por pantalla la ocupación de parking. Desbloqueamos el mutex y ponemos a dormir al vehículo. Ahora entramos en la segunda parte de la función que es cuando se quiere ir el vehículo. Bloqueamos el mutex. Volvemos a poner la variable i a 0 y buscamos en la posición donde esta el vehículo. Para el camión ponemos a 0 el array parking en la posición i y en la i+1 (ya que ocupa 2 posiciones) y para el coche la posición i. Restamos la ocupación (1 para los coches y 2 para los camiones) y calculamos las plazas libres. Por ultimo, mostramos la ocupación del parking, mandamos el signal de la condición para decir que hay plazas libres y cerramos el if en el caso de los camiones. Desbloqueamos el mutex y termina el while y vuelve a repetirse.

CONCLUSIONES

En la realización de esta práctia a diferencia de la anterior nos ha resultado mas sencilla y hemos sido capaces de realizarle prácticamente en una tarde. Cabe destacar que nos peleamos bastante con el problema de la cena de los filósofos y eso ha provocado que nos resultara mas sencillo entenderlo. Como problemas a la hora de realizar la practica nos hemos encontrado la falta de tiempo para realizar los apartados que nos quedan ya que la minishell nos ha quitado mucho tiempo y que trabajábamos los 2 durante las navidades ya que terminamos la otra practica casi al borde de las vacaciones de navidad. Otro problema minimo es los meticuloso que hay que ser sobre todo con los camiones para no cometer ningún violación del segmento con la condición if creada sobre todo, pero la pusimos para todo el bucle y lo solucionamos.