UNIVERSIDAD DE MÁLAGA
Dpto. Lenguajes y Ciencias de la Computación

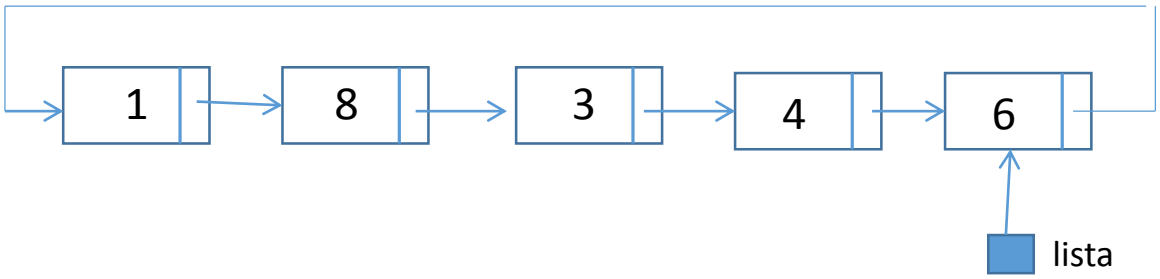
Programación de Sistemas y Concurrencia

Final Septiembre. 2015/2016
Ing. Computadores
Ing. Informática
Ing. Software

APELLIDOS _____ NOMBRE _____
DNI _____ ORDENADOR _____ GRUPO _____

Lenguaje C

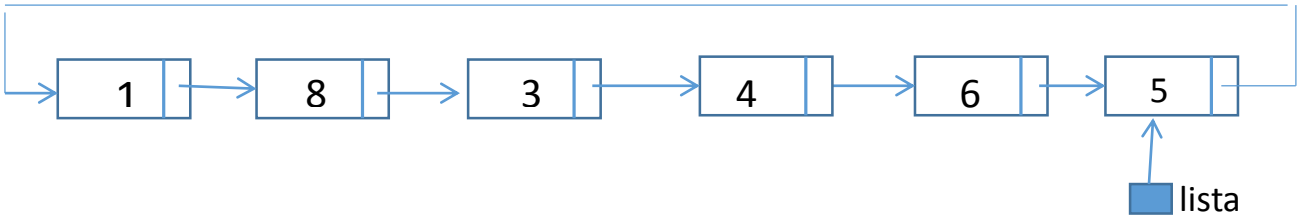
Se desea implementar una lista enlazada circular para representar la lista de procesos que se están ejecutando siguiendo la técnica de Round-Robin. La técnica Round-Robin es una forma de planificar (ejecutar) los procesos preparados para ejecución asignando un quantum de tiempo a cada uno, que es el intervalo de tiempo durante el cual se ejecuta el proceso. Por ejemplo, en la figura de abajo existen 5 procesos que se están ejecutando, el 1,8,3,4,6 y el planificador ejecutaría los procesos durante ese intervalo de tiempo siguiendo el orden 1,8,3,4,6, 1,8,3,4,6…. repitiendo el ciclo hasta que vayan terminando los procesos. Observa que, como es habitual en las listas enlazadas circulares, el puntero externo (`lista`) apunta al último nodo de la lista, con lo que se tiene fácil acceso a los dos extremos de la estructura.



Definir la estructura de datos e implementar las siguientes operaciones:

`void Crear (LProc *lista)`
Crea una lista de procesos vacía.

`void AnadirProceso (LProc *lista, int idproc)`
Añade el proceso con identificador `idproc` a la lista de procesos disponibles para ejecución. Este proceso se añade como el último nodo que se ejecutará en modo Round Robin. Dada la figura anterior, si queremos añadir el proceso 5, la lista quedaría como aparece en la figura:



`void EjecutarProcesos(LProc lista)`
Realiza un recorrido sobre la lista simulando la ejecución del quantum de tiempo de la lista de procesos. La ejecución de cada proceso se simula escribiendo su identificador en la pantalla.

`void EliminarProceso(int idproc, LProc *lista)`
Elimina el proceso `idproc` que ya ha terminado su ejecución. Suponemos que en la lista existe un proceso con ese identificador.

`void EscribirFichero ( char *nomf, LProc *lista)`
Guarda la información en un fichero binario dejando la lista vacía y liberando memoria. El fichero tendrá el formato siguiente:
<numero de procesos> <idproc><idproc>….

Concurrencia en memoria compartida

Supón que N>2 jugadores quieren hacer hacer un sorteo para elegir un ganador. Para ello, cada jugador tira una moneda, y si a alguno de ellos le ha salido un resultado distinto al de todos los demás, entonces es el ganador. En caso contrario, si ningún jugador obtiene un resultado distinto al del resto, el juego se repite de nuevo.

1.- **Semáforos**. Implementa este sistema utilizando únicamente **semáforos binarios**.
1.- **Métodos sincronizados/Locks**. Implementa este sistema utilizando monitores.

Para simular el juego suponemos que hay una Mesa en la que los jugadores ponen sus tiradas. Para todas las soluciones, utiliza el esqueleto que se encuentra en el campus virtual (es el mismo para todos los ejercicios), en el que existe una clase **Mesa** que proporciona los siguientes métodos:

```
public class Mesa {

	/**
	 * N es el número de jugadores que intervienen
	 */
	public Mesa(int N){

	}

	/**
	 *
	 * @param id del jugador que llama al método
	 * @param cara : true si la moneda es cara, false en otro caso
	 * @return un valor entre 0 y N. Si devuelve N es que ningún jugador
	* ha ganado y hay que repetir la partida. Si  devuelve un número
	* menor que N, es el id del jugador ganador.
	 *
	 * Un jugador llama al método nuevaJugada cuando quiere poner
	 * el resultado de su tirada (cara o cruz) sobre la mesa.
	 * El jugador deja su resultado y, si no es el último, espera a que el
	 * resto de los jugadores pongan sus jugadas sobre la mesa.
	 * El último jugador comprueba si hay o no un ganador, y despierta
	 * al resto de jugadores
	 *
	 */
	public int nuevaJugada(int id,boolean cara) {
	}
}
```