Universidad de Carabobo
Facultad Experimental de Ciencias y Tecnología
Departamento de Computación
Sistemas Operativos
Prep. Carlos I. Buchart I.

Nombre:_____

Sincronizacion de Procesos en Java y C
(Aproximaciones de Dekker y solucion de Peterson)

Si bien el lenguaje Java no esta diseñado para la construccion de sistemas operativos, destaca por la variedad de servicios que ofrece al desarrollador. Entre dichos servicios destacan la implementacion de hilos y el modificador de variables **synchronized**. La implementacion de hilos se realiza mediante la herencia de la clase **Thread**. Esta nueva clase debe implementar el metodo **run** que sera el encargado de ejecutar el codigo del hilo. Un hilo entra al estado **runnable** (preparado), al momento de la llamada al metodo **start** (heredado de Thread), llamada que efectua el padre del hilo. El metodo **yield**, por ultimo, permite seleccionar otro hilo para su ejecucion. Con respecto a C, este no soporta un control nativo de hilos, por lo que se simulara su uso mediante bifurcacion de procesos y zonas de memoria compartida (ver taller 2).

El codigo analizado muestra las aproximaciones de Dekker y la solucion de Peterson al problema de la seccion critica para dos competidores. La idea es experimentar con los valores de tiempo de “secciones criticas y no criticas”, inclusive eliminando la aleatoriedad del mismo, a fin de tratar de obtener, para las aproximaciones de Dekker, estados indeseables de interbloqueo.

Codigo completo de la practica

C:

```
// Solucion de Peterson al problema de la sección crítica
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/sem.h>

int mem_id_flags,mem_id_turn;
int *p_flags,*p_turn;
int id;

int main() {
        int hijo;

        // Crea el área de memoria compartida para las banderas y el turno
        if((mem_id_flags=shmget(IPC_PRIVATE,sizeof(int)*2,0774|IPC_CREAT))==-1) return -1;
        if(!(p_flags=(int *)shmat(mem_id_flags,(char *)0,0))) return -1;
        if((mem_id_turn=shmget(IPC_PRIVATE,sizeof(int),0774|IPC_CREAT))==-1) return -1;
        if(!(p_turn=(int *)shmat(mem_id_flags,(char *)0,0))) return -1;

        *p_turn=0;
        p_flags[0]=p_flags[1]=0;

        if(hijo=fork())
                id=0;
        else
                id=1;

        while(1) {
                int other;

                printf("Proceso %d desea entrar a la SC\n",id);
                other=1-id;
                p_flags[id]=1;
                *p_turn=other;
                while(p_flags[other]==1 && *p_turn==other); // Se usa espera activa

                printf("Proceso %d entró a la SC\n",id);
                sleep(1);
```

```
                printf("Proceso %d salió de la SC\n",id);
                p_flags[other]=0;
                sleep(1);
        }

        if(hijo) kill(hijo,9);

        shmdt((char *)p_flags);
        shmctl(mem_id_flags,IPC_RMID,(struct shmid_ds *)NULL);
        shmdt((char *)p_turn);
        shmctl(mem_id_turn,IPC_RMID,(struct shmid_ds *)NULL);

        return 0;
}
```

Java:

```
/** -Hilo
 * @author Carlos I. Buchart I. - CIBI3D
 */
public class Hilo extends Thread {
        private String name;
        private int m_iId;
        private ModeloExclusion m_oExc;

        public Hilo(String sName, int iId, ModeloExclusion oExc) {
                name = sName;
                m_iId = iId;
                m_oExc = oExc;
        }

        public void run() {
                while (true) {
                        System.out.println(name + "desea entrar a la SC");
                        m_oExc.entrarSC(m_iId);
                        System.out.println(name + "entro a la SC");
                        ModeloExclusion.SC();
                        m_oExc.salirSC(m_iId);
                        System.out.println(name + "salio de la SC");
                        ModeloExclusion.noSC();
                }
        }
}

/** -ModeloExclusion
 * @author Carlos I. Buchart I. - CIBI3D
 */
public abstract class ModeloExclusion {
        public static final int TURN_0 = 0;
        public static final int TURN_1 = 1;
        public static final int TIME = 2000;

        public static void SC() {
                try {
                        Thread.sleep((int) (Math.random() * TIME + 1000));
                } catch (InterruptedException e) {}
        }

        public static void noSC() {
                try {
                        Thread.sleep((int) (Math.random() * TIME));
                } catch (InterruptedException e) {}
        }

        public abstract void entrarSC(int iId);

        public abstract void salirSC(int iId);
}

/** -Dekker_1
 * @author Carlos I. Buchart I. - CIBI3D
 * No satisface condicion de progreso ya que requiere alternancia estricta.
```

```
 */
public class Dekker_1 extends ModeloExclusion {
	private volatile int m_iTurn;

	public Dekker_1() {
		m_iTurn = TURN_0;
	}

	public void entrarSC(int iId) {
		while (m_iTurn != iId)
			Thread.yield();
	}

	public void salirSC(int iId) {
		m_iTurn = 1 - iId;
	}

}

/** -Dekker_2
 * @author Carlos I. Buchart I. - CIBI3D
 * No satisface condicion de progreso ya que ambos hilos podrian fijar la bandera respectiva en
 * verdadero
 */
public class Dekker_2 extends ModeloExclusion {
	private volatile boolean[] m_bFlag = new boolean[2];

	public Dekker_2() {
		m_bFlag[0] = false;
		m_bFlag[1] = false;
	}

	public void entrarSC(int iId) {
		int other = 1 - iId;

		m_bFlag[iId] = true;

		while (m_bFlag[other] == true)
			Thread.yield();
	}

	public void salirSC(int iId) {
		m_bFlag[iId] = false;
	}

}

/** -Peterson
 * @author Carlos I. Buchart I. - CIBI3D
 * Solucion final
 */
public class Peterson extends ModeloExclusion {
	private volatile int m_iTurn;
	private volatile boolean[] m_bFlag = new boolean[2];

	public Peterson() {
		m_iTurn = TURN_0;
		m_bFlag[0] = false;
		m_bFlag[1] = false;
	}

	public void entrarSC(int iId) {
		int other = 1 - iId;

		m_bFlag[iId] = true;
		m_iTurn = other;

		while (m_bFlag[other] == true && m_iTurn == other)
			Thread.yield();
	}

	public void salirSC(int iId) {
		m_bFlag[iId] = false;
```

```
	}
}

/** -Test
 * @author Carlos I. Buchart I. - CIBI3D
 */
public class Test {
	public static void main(String[] args) {
		ModeloExclusion alg = new Dekker_1(); // alternar entre los algoritmos

		Hilo hilo1 = new Hilo("Hilo 1", 0, alg);
		Hilo hilo2 = new Hilo("Hilo 2", 1, alg);

		hilo1.start();
		hilo2.start();
	}
}
```

Bibliografía

Aprenda Java2 como si estuviera en primero – Universidad de Navarra. Pag. 118-126
Sistemas Operativos Sexta Edicion – A. Silberschatz, P. Galvin, G. Gagne. Pag. 173-181.
Sistemas Operativos – A. Tanenbaum. Pag. 58-

CB/cb