INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO

SISTEMAS OPERATIVOS

2CV7

PRACTICA 5: SEMÁFOROS

INTEGRANTES:

- RAMÍREZ NARVAEZ ALAN MAURICIO
- MUÑOZ BALDERAS JORDY
- PÉREZ GARDUÑO JOSÉ EMILIANO

FECHA ENTREGA:07/11/2018

Introducción

SEMÁFOROS

Un semáforo da acceso al recurso a uno de los procesos y se lo niega a los demás mientras el primero no termine. Los semáforos, junto con la memoria compartida y las colas de mensajes, son los recursos compartidos que suministra UNIX para comunicación entre procesos.

Ahora un segundo proceso lo intenta y para ello también decrementa el contador. Esta vez el contador se pone a -1 y como es negativo, el semáforo se encarga de que el proceso quede "bloqueado" y "dormido" en una cola de espera. Este segundo proceso no continuará por tanto su ejecución y no accederá al fichero.

Para utilizar los semáforos en un programa, deben seguirse los siguientes pasos:

-Obtener una clave de semáforo.Para ello se utiliza la función key_t ftok(char *, int) a la que se suministra como primer parámetro el nombre y path de un fichero cualquiera que exista y como segundo un entero cualquiera.

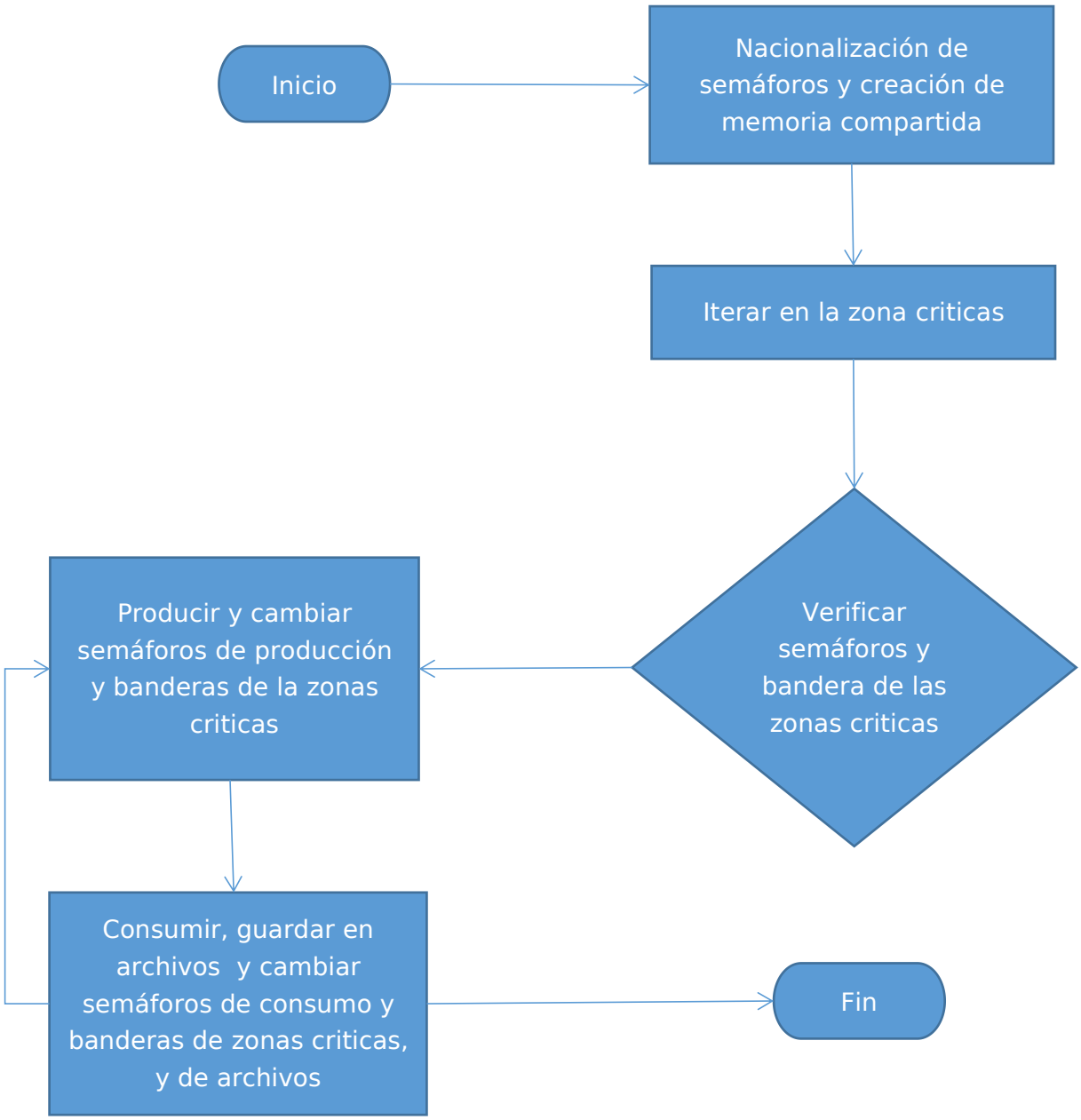
-Obtener un array de semáforos. La función int semget (key_t, int, int) nos permite obtener un array de semáforos.

-Uno de los procesos debe inicializar el semáforo. La función a utilizar es int semctl (int, int, int, int). El primer parámetro es el identificador del array de semáforos obtenido anteriormente, el segundo parámetro es el índice del semáforo que queremos inicializar dentro del array de semáforos obtenido. Si sólo hemos pedido uno, el segundo parámetro será 0.

-Ya está todo preparado, ahora sólo queda usar los semáforos. El proceso que quiera acceder a un recurso común debe primero decrementar el semáforo. Para ello utilizará la función int semop (int, strcut sembuf *, size_t).

Desarrollo

Diagrama Solución



Se crean y se se vinculan el productor (4 productores) y consumidor (3 consumidores) tanto a los semáforos como a la memoria compartida, también se crea las zonas criticas que se van a utilizar (10 zonas criticas).
Cada productor y consumidor itera en las zonas criticas y comprueban su bandera para saber si pueden producir o consumir. En cualquier caso se tiene que incrementar y decrementar los semáforos para poder utilizar las zonas criticas una vez se haya producido o consumido.
Cuando se consume también se tienen semáforos para los archivos en los cuales se van a escribir en un archivo correspondiente lo consumido. Esto debido a que son considerados zonas criticas y que no haya inconsistencia de datos o problemas entre dos o mas procesos.
Una vez se concluyan las producciones y consumos, se termina la ejecución y se puede proceder a revisar los archivos creados por el consumidor; cada uno con sus respectivos mensajes.

Conclusiones

Muñoz Balderas Jordy:
Los semáforos son de gran utilidad ya que con ellos podemos indicar a dos o mas procesos cuando pueden hacer uso de algún recurso y no haya inconsistencia de datos.
Una zona critica puede llegar a ser código, archivos, memoria compartida o cualquier otro recurso compartido. Con esta combinación podemos sincronizar los procesos.

Ramírez Narvaez Alan Mauricio:
Con esta practica llegue a la conclusión de como pueden usarse los procesos para dividir tareas, así como el uso de señales para manipular estos, como por ejemplo puede terminarse un proceso en determinado punto y que continúe otro.

Pérez Garduño José Emiliano:
Aprendimos a utilizar los semáforos, las funciones que tienen, sus aplicaciones y que tipos de semáforos hay, los problemas que pueden presentar, como la desincronización de los procesos, sus ventajas y desventajas, como definir la memoria critica y así evitar la inconsistencia de datos. Lo que mas aprendí fue que los semáforos son muy complicados de programar pero a la vez permiten mayor libertad en ciertas ejecuciones.