|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Producir | Consumir | Posición prod | Posición consum | Sem element | Sem huecos | Proceso  bloqueado |
| 4 | 0 | 4 | 0 | 4 | 4 | Ninguno |
| 3 | 3 | 7 | 3 | 4 | 4 | Ninguno |
| 3 | 2 | 2 | 5 | 5 | 3 | Ninguno |
| 2 | 0 | 4 | 5 | 7 | 1 | Ninguno |
| 2 | 0 | 5 | 5 | 8 | -1 | Productor |
| 0 | 4 | 6 | 1 | 5 | 3 | Ninguno |
| 0 | 4 | 6 | 5 | 1 | 7 | ninguno |

**1) El planteamiento de este problema clásico de sincronización (El problema de**[**Lectores escritores**](https://plataforma.utp.edu.co/mod/url/view.php?id=1587)**).**

El problema del lector y escritor se basa en procesos lectores y escritores que acceden a un recurso compartido, haciendo que para poder acceder a este sea de exclusión mutua, haciendo que solo pueda entrar un proceso a la vez, ya sea un lector o un escritor.

**2) En que consiste dar prioridad al lector.**

Si hay un proceso de lectura en la zona crítica y llega otro proceso este entrara después de este sin importar que ya haya uno de escritura esperando. La única forma que uno de lectura espere es que este uno de escritura ya adentro.

**3) En que consiste dar prioridad al escritor.**

Si hay un escritor dentro de la zona crítica y hay un proceso de lectura y otro de escritura esperando para entrar se le dará prioridad al de escritura. La única forma en que entre uno de lectura es que no haya ninguno de escritura esperando ni dentro de la zona critica.

**4) Observe la siguiente solución y responda:**

**lector()**

**{**

**wait(sem\_lectores);**

**n\_lectores=n\_lectores+1;**

**if (n\_lectores == 1)**

**wait (sem\_recurso);**

**signal (sem\_lectores);**

**<Consultar recurso compartido >**

**wait (sem\_lectores);**

**n\_lectores=n\_lectores-1;**

**if (n\_lectores == 0)**

**signal (sem\_recurso);**

**signal (sem\_lectores);**

**}**

**escritor ()**

**{   wait (sem\_recurso);**

**/\* Se puede modificar el recurso \*/**

**signal (sem\_recurso);**

**}**

**a) Para que es usado el semáforo sem\_recurso y cuál debe ser su valor inicial?**

El valor de la variable sem\_recurso debe iniciar el máximo del buffer, y este es el semáforo es el que controla la entrada a la zona critica diciendo si hay o no recursos disponibles.

**b) Para que se usa la variable n\_lectores?**

La variable n\_lectores es la que se usa para controlar cuantos lectores hay dentro de la zona critica.

**c) Para qué es usado el semáforo sem\_lectores?**

La variable sem\_lectores es el semáforo que permite la entrada de lectores a la zona critica.

**d) Qué hace el primer proceso lector?**

Si el es primer lector verifica si la zona critica está vacía, y si lo está bloquea esta misma para que no puedan acceder más procesos a este.

**e) Qué hace el resto de los procesos lectores?**

Los otros procesos lectores pueden entrar a la zona de exclusión mutua mientras este reservada para lectura, afectando cada uno las variables de n\_lectores y sem\_lectores

**f) Que hace el último proceso lector?**

El último proceso al salir lo que hace desbloquear la zona critica con la variable sem\_lectores para que puedan entrar los procesos de escritura.

**g) A quien concede prioridad esta solución?**

Esta solución ofrece prioridad a los procesos de lectura.

**SOBRE COMUNICACIÓN ENTRE PROCESOS.**

**Who:** Este comando dice que usuarios están loggeados en este momento en la máquina.

**Sort:** Este comando sirve para ordenar dependiendo de la forma que se le diga.

**Pr:** Este comando sirve para imprimir en una o varias líneas.

**Lp:** Envía o cancela solicitudes a una impresora.

**Wc:** Es un comando que sirve para contar dependiendo del parámetro que se le ponga.

El comando who | sort | pr | lp: Primero who permite saber quién está conectado, sort ordena esta información en un fichero, pr prepara este fichero para imprimir y finalmente lp le envía la solicitud de imprimirlo a una impresora.

El comando who | wc -l : Con who consulta quienes están conectados a nuestro equipo, y luego a través de wc-l son contados en líneas, es decir, dice cuántos hay.

El pipeline es el símbolo “|”, que sirve para conectar procesos, teniendo el resultado del primero y pasándoselo al segundo y así consecutivamente hasta acabar