OS, class6

Producer/Consumer

N

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | ... |  |

|  |  |
| --- | --- |
| **Producer** | **Consumer** |
| creates an item  stores it | buys an item |

**Condition** :

if I cannot have it -> wait.

Make available what I’m waiting for.

Think about different scenarios : *should they be synchronized or not?*

Producer : If I don’t have storage space, I can’t create

I can’t store

WAIT

Consumer can release free space

Producer : If I don’t have a newly created item, I can’t store it

Consumer : If I don’t have stored item, I can’t buy one

Consumer : If no filled space, WAIT

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Producer** | **Consumer** |  |
| Aquire(SfreeSpace)<-  Release(SfilledSpace)<- | creates an item  **if(no free space) WAIT**  i <- ChooseEmptySpot ()  stores item in i  **release filled space** | **if (no filled space) WAIT**  i <- ChooseItem()  buys item i  **release free space** | ->Aquire(SfilledSpace)  ->Release(SfreeSpace) |

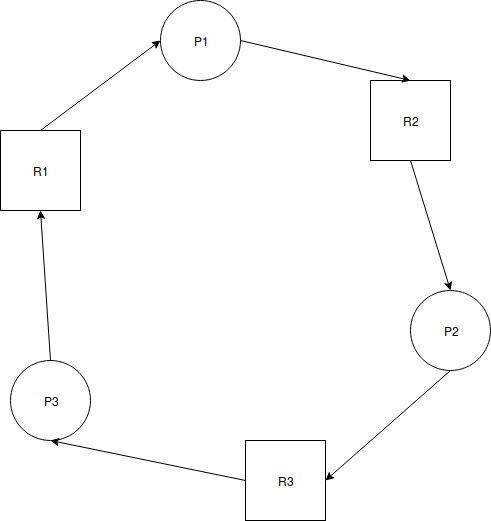
() Acquire(Sfreespace); if its = 0; it’s going to make the system wait. Either way, it’s going to give the space to the item

We’re going to create two semaphores :

* Sfreespace(N)
* Sfilledspace(0)

This is useful for the second part of the lab 5.

In the first part there is a race in creating a deadlock



|  |  |  |
| --- | --- | --- |
| T1 | T2 | T3 |
| emacs | freef | vi |
| 6am | 10am | 10pm |
| T3 -> | T2 -> | T1 |