

Message Queue

**Lock()** pour éviter que toutes les maisons envoient leur données en même temps et posent un problème au niveau du calcul du prix

**Lock()** pour empêcher que le market prenne une valeur qui est en train d'être modifiée  
structure de données = tableau:

Update terminal



## Exécution

permet de créer et d'établir la communication entre les différents processus

## Home

- numero\_maison
- prod\_energie
- conso\_energie
- politique\_energie (always give away / always sell on the market / sell if no takers)
- $f_{2,t}$  sell/buy

Signaux :  
 $u_{1,t}$  SIGTERM  
 $u_{2,t}$  SIGILL

Message Queue

## Market

parent of Politics & Economics

- $P_t$  (energyPrice\_t)
- $P_{t-1}$  (energyPrice\_t-1)
- $f_{i,t}$  (temp U sell/buy)
- $u_{j,t} \in \{0,1\}$  (diploTensions U wars U penurieCarburant U criseDevice)
- Stock\_energie

multithreading ThreadPool  
(transactions avec Home dans un thread différent) : limite le nombre de transactions simultanées avec Home

Shared memory

## Weather

- $f_{1,t}$  temp

Signaux :  
 $u_{3,t}$  SIGUSR1  
 $u_{4,t}$  SIGUSR2

## Politics

child of Market

- $u_{1,t} \in \{0,1\}$  diploTensions
- $u_{2,t} \in \{0,1\}$  wars

## Economics

child of Market

- $u_{3,t} \in \{0,1\}$  penurieCarburant
- $u_{4,t} \in \{0,1\}$  criseDevice

## DESIGN

- relationships between processes (parent-child or unrelated) : **OK**
- messages exchanged between processes along with their types : Queues **OK**
- data structures stored in shared memory **OK** and how they are accessed **OK**
- synchronization primitives to protect access to shared resources or to count resources : **Lock()** entre **Home et Market** et entre **Weather et Market**
- signals exchanged between processes, if any, and their types **OK**
- tubes (pipes) involved in pairwise process communication, if any, and their types : **none**
- Python-like pseudo-code of main algorithms for each process -> **voir le code**

# Implementation plan:

1. Coder chaque processus séparément, les tester individuellement pour voir s'ils fonctionnent, les debugger sinon
2. Tester deux à deux les paires de processus qui communiquent entre-eux pour voir s'ils communiquent, les debugger sinon
3. Si tout fonctionne, essayer de faire communiquer tous les processus ensemble

Scénarios catastrophes :

-> tout mettre sur GitHub pour récupérer les archives en cas de besoin

-> plus d'énergie dans le market : faire une méthode qui rajoute de l'énergie dans le stock

Automatiser le démarrage et la fermeture de la simulation en remettant à 0 les données :

-> hardcoder les valeurs de base de chaque paramètre

-> faire un "finally" qui ferme les processus