# Sémaphore local

Anthony Araye et Camille Schnell

21 février 2018

## Sommaire

# 1 Introduction

*Cinq philosophes se trouvent autour d'une table avec en face d'eux un plat de spaghetti et à gauche de chaque plat se trouve un couvert (une fourchette ou un couteau). Un philosophe ne possède que trois états :*

— *penser pendant un temps indéterminé*

— *être affamé pendant un temps déterminé et fini*

— *manger pendant un temps déterminé et fini.*

*Cependant, quand un philosophe a faim, il se met en état "affamé" et va attendre que les couverts autour de son assiette soient libres pour pouvoir manger. Et dans le cas ou l'un des deux couverts n'est pas libre, le philosophe se met en état de famine pendant un temps déterminé en attendant de revérifier.*

Cette situation représente en réalité le problème du "dîner des philosophes" énoncé par Dijkstra.

Le but de ce sujet est d'implémenter un système de sémaphore afin de gérer des sections critiques et ainsi répondre au problème ci-dessus.

# 2 Manuel d’utilisation

## 2.1 Tutoriel d’utilisation

## 2.2 initialize

**Nom**

sem_initialize - Initialise un sémaphore

**Synopsis**

**#include <sem.h>**

**int sem_initialize(sem **sem*, int *value*)**

**Description**

**sem_initialize**() alloue et initialise *sem* avec *value* comme valeur initiale. *value* doit être supérieur ou égale à 0.

**Valeur renvoyée**

**sem_initialize**() renvoie 0 dans le cas où le sémaphore a bien été créé, -1 si une erreur a été levé.

**Erreurs**

EFAULT : *value* est strictement inférieur à 0.

## 2.3 acquire

**Nom**

sem_acquire - Décrémente la valeur du sémaphore

**Synopsis**

**#include <sem.h>**

**int sem_acquire(sem **sem*)**

**Description**

**sem_acquire**() décrémente la valeur du sémaphore *sem*. Dans le cas où cette valeur est inférieur ou égale à 0, alors il bloque le processus courant et le met dans la file d'attente associée au sémaphore.

**Valeur renvoyée**

**sem_acquire**() renvoie 0 dans le cas où le processus s'est bien effectué, -1 si une erreur a été levé.

**Erreurs**

?

## 2.4 release

**Nom**

sem_release - Incrémente la valeur du sémaphore

**Synopsis**

**#include <sem.h>**
**int sem_release(sem **sem*)**

**Description**

**sem_release**() incrémente la valeur du sémaphore *sem*. Dans le cas où la file associée à *sem* est non vide, alors il réveille le processus en tête de file et saute la tête de la file.

**Valeur renvoyée**

**sem_release**() renvoie 0 dans le cas où le processus s'est bien effectué, -1 si une erreur a été levé.

**Erreurs**

?

## 2.5 destroy

**Nom**

sem_destroy - Détruit le sémaphore

**Synopsis**

**#include <sem.h>**

**int sem_destroy(sem **sem*)**

**Description**

**sem_destroy**() détruit le sémaphore *sem* et désalloue la file d'attente.

**Valeur renvoyée**

**sem_destroy**() renvoie 0 dans le cas où le sémaphore a bien été détruit, -1 si une erreur a été levé.

**Erreurs**

?

# 3 Implémentation

# 4 Recette