Rapport ENSTA 2021 – ROB305 – Howard Wolowitz

# Mesure de temps et échantillonnage en temps

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor. Cras elementum ultrices diam. Maecenas ligula massa, varius a, semper congue, euismod non, mi.

## Gestion simplifiée du temps Posix

Nous avons implémenté un ensemble de fonctions facilitant l’utilisation de la structure timespec, ainsi que surchargé des opérateurs pour pouvoir les utiliser. Nous les avons ensuite testés.

## Timers avec callback

Nous avons implémenté un *callback* qui incrémente un compteur, comme on peut le voir ci-dessous.

void handler(int /\*sig\*/, siginfo\_t\* si, void\* /\*unused\*/)

{

int\* p\_counter = (int\*) si-> si\_value.sival\_ptr;

\*p\_counter += 1; // ou p\_counter[0]

std::cout << \*p\_counter << std::endl;

}

## Fonction simple consommant du CPU

Pour un compteur de 2000, les valeurs finales obtenues sont :

2000

time : 0,000075313s

## Mesure du temps d’exécution d’une fonction

En calibrant la fonction *incr()* de manière linéaire (ax+b), nous avons trouvé comme paramètres

a : 1.81389e+07

b : 18720

En faisant des vérifications entre le temps estimé et le temps réel, nous avons obtenu comme résultats :

time : 55,117513781s

expected time : 55,129140625s

diff time : -1,988373156s

Donc l’estimation est plutôt satisfaisante.

## Amélioration des mesures

Proin porttitor, orci nec nonummy molestie, enim est eleifend mi, non fermentum diam nisl sit amet erat. Duis semper. Duis arcu massa, scelerisque vitae, consequat in, pretium a, enim. Pellentesque congue. Ut in risus volutpat libero pharetra tempor.

Cras vestibulum bibendum augue. Praesent egestas leo in pede. Praesent blandit odio eu enim. Pellentesque sed dui ut augue blandit sodales. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aliquam nibh. Mauris ac mauris sed pede pellentesque fermentum. Maecenas adipiscing ante non diam sodales hendrerit.

# Familiarisation avec l’API multitâches pthread

## Exécution sur plusieurs tâches sans mutex

## Mesure de temps d’exécution

## Exécution sur plusieurs tâches avec mutex

# Classes pour la gestion du temps

## Classe Chrono

## Classe Timer

## Calibration en temps d’une boucle

# Classes de base pour la programmation multitâche

## Classe Thread

## Classes Mutex et Mutex::Lock

## Classe Semaphore

## Classe Fifo multitâches

# Inversion de priorité