Programmation parallèle

Cycle 2020-2021

ECE – Ing5 – Systèmes Embarqués

Etienne Hamelin & Alexandre Berne

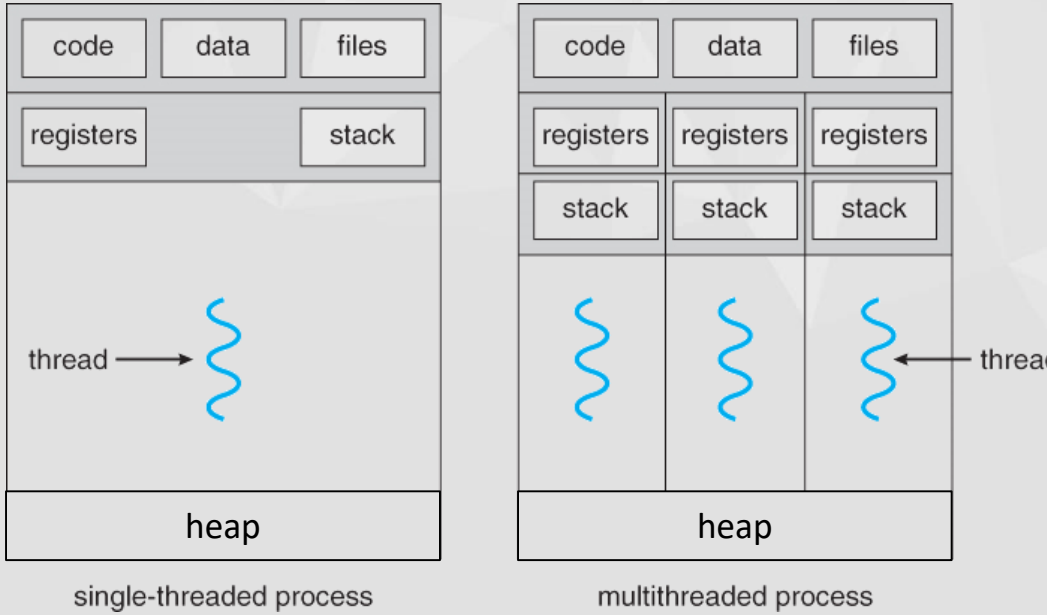
ehamelin@inseec-edu.com

aberne@inseec-edu.com

Séance 1

TP1 – pthreads

- PThreads: POSIX Threads
- Librairie de référence pour le multithreading en C



Rappels – le modèle mémoire en C

- Variables globales, et *static* :
  - segment *data*
- Variables allouées dynamiquement (malloc, calloc, free)
  - dans le tas *heap*

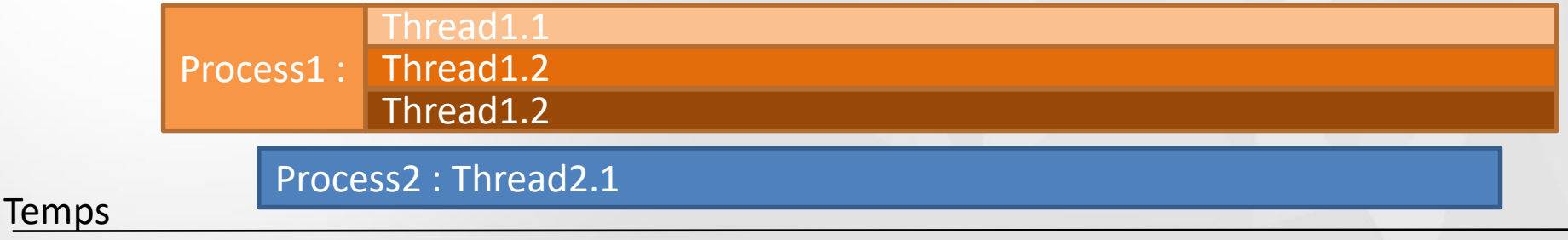
Mémoire partagée par les threads

- Variables automatiques (locales, non-*static*):
  - dans la pile *stack*

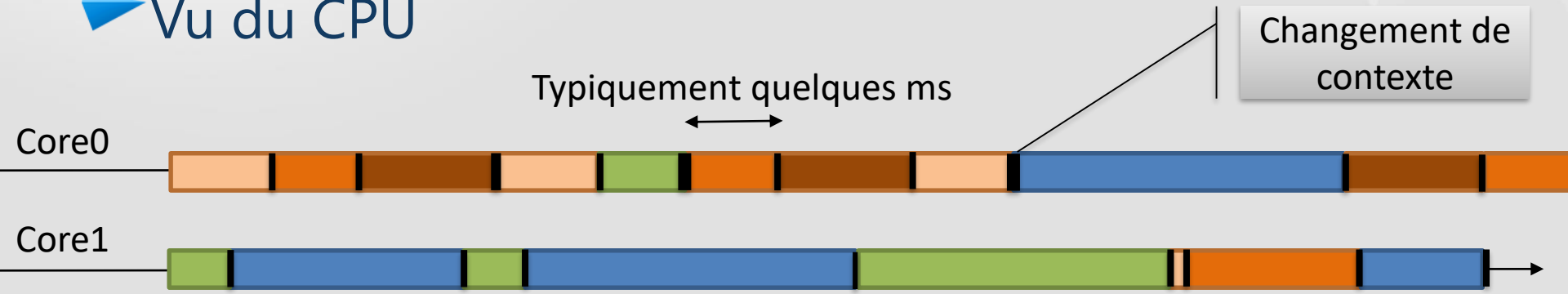
1 stack par thread

Etienne HAMELIN – ehamelin@inseec-edu.com
Alexandre BERNE – aberne@inseec-edu.com

Vu du programmeur



Vu du CPU



Etienne HAMELIN – ehamelin@inseec-edu.com
Alexandre BERNE – aberne@inseec-edu.com

Mesures de temps

- Mesurer le temps d'execution d'un programme:
  - `time -p` (mon programme) (paramètres du programme)

- Pour des mesures fiables et reproductibles ⚠
  - Power management → PC branché sur secteur
  - Préemptions → système peu chargé

Etienne HAMELIN – ehamelin@inseec-edu.com
Alexandre BERNE – aberne@inseec-edu.com

Commandes de base

- Éditer le code: vim, notepad++, VisualStudio Code, … votre choix.

```
code monprogramme.c &
```

- Compiler avec les librairies Math (-lm) et Pthread (-lpthread),

```
gcc -o monbinaire monprogramme.c malibrairie.c -lm -lpthread
```

(ou voir Makefile)

- Exécuter le programme

```
./monbinaire {arguments}
```

- Mesurer le temps d'exécution

```
time -p ./monbinaire {arguments}
```

Aide sur pthreads

https://computing.llnl.gov/tutorials/pthreads/

TP par binôme

- COVID-safe: chacun manipule sur 1 machine
- 1 rapport par binôme
- Toutes les mesures effectuées sur *la même* machine

Compte-rendu de TP

- Une introduction, une conclusion
- Réponses aux **questions**
- Des phrases courtes mais complètes (sujet, verbe, compléments…)

Annexes

- Votre **code source en annexe**
- Un pdf, nommé: « TP1-{nom1}-{nom2}.pdf »
- A déposer sur http://campus.ece.fr **avant** le mercredi soir.

PThreads : exemple

```c
#include <stdio.h>    // printf, etc.
#include <stdlib.h>   // malloc, ...
#include <pthread.h>  // pthreads !

typedef struct {
  int x;
  int y;
} thread_args_t;

void *thread_runner(void *thread_arg) {
  thread_args_t *my_args = (thread_args_t
*)(thread_arg);
  int z = my_args->x + my_args->y;
  printf("Coucou du thread %p\n",
pthread_self());
  return (void *)z;
}
```

```c
int main(int argc, char **argv) {
  int n_threads; int rc;
  pthread_t *my_threads;
  thread_args_t *my_args;
  void *thread_return;
  n_threads = 8;
  my_threads = calloc(n_threads, sizeof(pthread_t));
  my_args = calloc(n_threads, sizeof(thread_args_t));
  for (int i = 0; i < n_threads; i++ ) {
    my_args[i].x = ...;
    my_args[i].y = ...;
    pthread_create(&my_threads[i], NULL,
        thread_runner, (void *)&my_args[i]);
  }

  for (int i = 0; i < n_threads; i++) {
    rc = pthread_join(my_threads[i], &thread_return);
  }
  return 0;
}
```


Etienne HAMELIN – ehamelin@inseec-edu.com
Alexandre BERNE – aberne@inseec-edu.com

Récupérer un argument sur la ligne de commande

```
./monprogramme 123 un_autre_argument 'un troisieme'
  ↳ argv[0]    ↳ argv[1] ↳ argv[2]   ↳ argv[3]        argc = 4.
```

```
int main(int argc, char **argv)
{
      int num;
      if(argc<2)
      {
            printf("Pas d'argument specifie, j'utilise 10.\n");
            num = 10;
      }
      else {
            num = atoi(argv[1]);
      }
      printf("Nous utiliserons %d threads\n", num);
}
```


Etienne HAMELIN – ehamelin@inseec-edu.com
Alexandre BERNE – aberne@inseec-edu.com