Complément de cours IDU – INFO 632

Threads POSIX

Sébastien Monnet

Norme POSIX

«**P**ortable **O**perating **S**ystem **I**nterface (as in uni**X**)»

BUT :

garantir la **portabilité des applications** au niveau du **code source** entre des Systèmes d'Exploitation (« OS ») respectant cette norme (« POSIX compliant »).

MOYEN :

définir les services offerts par le Système d'Exploitation et les interfaces avec les langages de programmation (« API : Application Program Interface »).

Execution concurrente avec POSIX

- Deux mécanismes : fork et **pthreads**
- fork : création d'un nouveau ***processus***
- **pthreads** : extension à POSIX pour permettre de créer des ***threads*** (dans un même processus)
  - **Utilisation de thread dans ce cours**
  - ***Threads*** sont crées en utilisant un objet avec des attributs appropriés
- Librairie :
  - `#include <pthread.h>`
  - Compilation : en utilisant –lpthread.

Thread : fil d'exécution d'un programme

- Pour quoi faire : avoir plusieurs fils d'exécution en même temps
- Threads d'un processus partagent :
  - En mémoire : text (code), data (variables globales)
  - Descripteur : PID, gestion signaux et fichiers ouverts
- Threads d'un processus ne partagent pas :
  - En mémoire : pile (variables locales)
  - Descripteur : copie registres

Création de threads POSIX : pthread_create()

- Synopsis :

```
#include <pthread.h>
int pthread_create(pthread_t * thread,
                   const pthread_attr_t * attr,
                   void * (*start_routine)(void *)
                   void * arg);
```

- Description :
  - Permet de créer un nouveau thread
  - Thread exécute procédure `start_routine` en lui passant `arg` comme argument
  - `attr` est généralement `NULL`
  - En cas de succès dans la création, identifiant thread placé dans `thread`

Sortie de threads POSIX : pthread_exit()

- Synopsis :

```
#include <pthread.h>
int pthread_exit(void *value_ptr);
```

- Description :
  - Permet de savoir quand un thread se termine
  - `value_ptr` est généralement `0`
  - Utilisation dans la procédure du thread associé

Attente de la fin thread POSIX : pthread_join()

- Synopsis :

```
#include <pthread.h>
int pthread_join(pthread_t th,
                 void **thread_return);
```

- Description :
  - Suspend l'exécution du thread **appelant** jusqu'à la fin du thread `th`
  - `thread_return` est généralement `NULL`
  - Tout thread doit être attendu si on veut éviter des fuites mémoires
  - Utilisation dans le programme principal

Gestion variables

- Variables partagées : déclaration en globale et volatile

```
volatile int total = 0;
```

- Passage de paramètre à la procédure exécutée :
  - dans le processus du thread :

```
int mon_numero = *(int *) arg ;
```

  - dans la procédure principale :

```
int numero = 2;
pthread_create(&mon_thread, NULL,ma_procedure,
(void *)&numero)
```

Exemple complet 1 :

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <stdlib.h>
#include <pthread.h>


void *fonction_thread (void *
arg)
{
   int i;
   for (i = 0 ; i < 5 ; i++) {
      printf ("%s thread:
%d\n", (char*)arg, i);
      usleep(5000000);
   }
pthread_exit(0);
}
```

```
int main (void) {
   pthread_t th1, th2;
   void *ret;


   if (pthread_create (&th1,
NULL, fonction_thread,
"Premier") < 0) {
      perror("premier
(pthread_create)");
     exit (-1);
   }
   if (pthread_create (&th2,
NULL, fonction_thread,
"Second") < 0) {
      perror("second
(pthread_create)");
      exit (-1);
   }
   pthread_join (th1, &ret);
   pthread_join (th2, &ret);
   return 0;
}
```

Exemple complet 2

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <stdlib.h>
#include <pthread.h>


volatile int total = 0;


void *fonction_thread (void *
arg)
{
   int mon_numero=*(int *)arg;
   for (i = 0 ; i < 5 ; i++)
      total = total + 1 ;
   printf ("Je suis le thread:
%d\n", mon_numero);
pthread_exit(0);
}
```

```
int main (void) {
   pthread_t th;
   int numero = 2 ;


   pthread_create (&th, NULL,
fonction_thread, (void
*)&numero);
   pthread_join (th1, NULL);
   printf("total=%d\n",total);
   return 0;
}
```