

# STR : Threads

TP :

Cette présentation contient des extraits du code, les programmes sources complets ainsi qu'un fichier Makefile pour les compiler peuvent être retrouvés dans l'archive rendu.

## Question 1 :

1 - En exécutant la commande :

**man 1 time**

```

TIME(1)                                General Commands Manual                                TIME(1)

NAME
    time - run programs and summarize system resource usage

SYNOPSIS
    time [ -apqvV ] [ -f FORMAT ] [ -o FILE ]
        [ --append ] [ --verbose ] [ --quiet ] [ --portability ]
        [ --format=FORMAT ] [ --output=FILE ] [ --version ]
        [ --help ] COMMAND [ ARGS ]

DESCRIPTION
    time run the program COMMAND with any given arguments ARG.... When
    COMMAND finishes, time displays information about resources used by
    COMMAND (on the standard error output, by default). If COMMAND exits
    with non-zero status, time displays a warning message and the exit
    status.

    time determines which information to display about the resources used
    by the COMMAND from the string FORMAT. If no format is specified on
    the command line, but the TIME environment variable is set, its value
    is used as the format. Otherwise, a default format built into time is
    used.

Manual page time(1) line 1 (press h for help or q to quit)

```

**time** affiche selon l'argument les ressources utilisées pour l'exécution d'une commande notamment la durée d'exécution. **time** affiche également un warning si la commande ne se termine pas d'une manière normale c-à-d en retournant un 0.

2 - En exécutant un make dans le dossier : *Fork\_vs\_thread*, nous obtenons :

```

e@ubuntu:~/Bureau/TP_Threads/Mod0/Fork_vs_Thread$ make
gcc -std=c99 -c testFork.c -o testFork.o
gcc testFork.o -o testFork
gcc -std=c99 -c testThread.c -o testThread.o
gcc testThread.o -o testThread -lpthread
Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz

+-----+-----+-----+
| Durée (s) | effective | user  | sys. |
+-----+-----+-----+
| fork      | 4.46      | 2.94  | 1.70 |
+-----+-----+-----+
| thread    | 1.05      | 0.21  | 0.85 |
+-----+-----+-----+

```

Nous remarquons que la version à base de Thread est environ 4 fois plus rapide que la version fork (à base de processus).

## Question 2 :

- On peut afficher le contenu de pthread.h avec la commande :  
**cat \$(locate pthread.h)** ou **la commande less** si fichier très grand
- Programme thread1.c qui crée deux threads, et utilise la fonction atexit(void fonction) qui exécute la fonction en argument dès qu'un appel system exit() se produit pour signaler la fin de chaque thread.

```
#define DUREE_MAX 5
#define CHECK_T(status, msg) \
    if (0 != (status)) { \
        fprintf(stderr, "pthread erreur : %s\n", msg); \
        exit (EXIT_FAILURE); \
    }

void *monThread (void * arg);
void bye(void);

int main(int argc, char *argv[])
{
    pthread_t tid1, tid2; /* Identifiants de thread */

    atexit(bye);
    printf("Le processus %d va créer deux threads\n",
        getpid());

    CHECK_T(pthread_create (&tid1, NULL, monThread, NULL),
        "pthread_create(1)");

    CHECK_T(pthread_create (&tid2, NULL, monThread, NULL),
        "pthread_create(2)");

    printf("\tattente de la terminaison des threads\n");

    CHECK_T(pthread_join (tid1, NULL), "pthread_join(1)");
    CHECK_T(pthread_join (tid2, NULL), "pthread_join(2)");
    return EXIT_SUCCESS;
}

void *monThread (void * arg)
{
    printf("Thread fils : \tDébut de traitement\n");
    sleep (rand() % DUREE_MAX + 1);
    printf("Thread fils : \tFin de traitement\n");
    pthread_exit(NULL);
}

void bye (void)
{
    printf("Fin du processus %d\n", getpid());
}
```

## Question 3 :

a - En exécutant la commande : `gcc thread1.c -Wall -o thread1` , on oublie de demander une liaison avec la librairie **"pthread.so"** qui contient les fonctions **"pthread\_join"** et **"pthread\_create"**.

b - Pour demander la liaison avec la librairie **"pthread.so"**, on ajoute **-lnomdelalibraire** à la compilation. On compile donc avec la commande :

**gcc thread1.c -Wall -o thread1 -lpthread**

```
elMehdi_C@UNSH:~/Bureau/TP_threads/Mod0/3_threads$ ./thread1
Le processus 8636 va créer deux threads
    attente de la terminaison des threads
Thread fils : Début de traitement
Thread fils : Début de traitement
Thread fils : Fin de traitement
Thread fils : Fin de traitement
Fin du processus 8636
```

#### **Question 4 :**

Programme du thread2.c :

```
void *monThread (void * arg);
void bye(void);

pthread_t tid[NBMAX_THREADS];

int main(int argc, char *argv[])
{
    int i;

    atexit(bye);
    printf("Le processus %d va créer %d threads\n",
        getpid(), NBMAX_THREADS);

    for (i = 0; i < NBMAX_THREADS; i++)
        CHECK_T(pthread_create (&tid[i], NULL, monThread, NULL),
            "pthread_create()");

    printf("\tattente de la terminaison des threads\n");

    for (i = 0; i < NBMAX_THREADS; i++)
        CHECK_T(pthread_join (tid[i], NULL),"pthread_join()");
    return EXIT_SUCCESS;
}

void *monThread (void * arg)
{
    pthread_t me = pthread_self();
    int no, i;
    //Incréméntation de no jusqu'à trouver le nombre du thread actuel
    for (no = 0; no < NBMAX_THREADS && !pthread_equal(me, tid[no]); no++);
    //printf f de no * indentations
    for (i = 0; i <= no; i++) printf(" ");
    //Impression du numéro du thread
    printf("Début du thread n°%d\n", no + 1);
    sleep (rand() % DUREE_MAX + 1);
    for (i = 0; i <= no; i++) printf(" ");
    printf("Fin du thread n°%d\n", no + 1);
    pthread_exit(NULL);
}

void bye (void)
{
    printf("Fin du processus %d\n", getpid());
}
```

résultat de l'exécution :

```
elMehdi_C@UNSH:~/Bureau/TP_threads/Mod0/3_threads$ ./thread2
Le processus 9634 va créer 4 threads
    Début du thread n°1
        Début du thread n°2
            Début du thread n°3
                attente de la terminaison des threads
                    Début du thread n°4
                        Fin du thread n°4
                    Fin du thread n°2
                        Fin du thread n°3
                            Fin du thread n°1
                                Fin du processus 9634
```



### Question 5 :

On s'inspirant des programmes fournis par l'enseignant sur Moodle, on écrit le programme du thread3.c exploitant un forçage de type avec `typedef void * (*pf_t)(void *)`:

```
typedef void * (*pf_t)(void *);

void monThread (long no);
void bye(void);

int main(int argc, char *argv[])
{
    pthread_t tid[NBMAX_THREADS];
    long i;

    atexit(bye);
    printf("Le processus %d va créer %d threads\n",
           getpid(), NBMAX_THREADS);

    for (i = 0; i < NBMAX_THREADS; i++)
        CHECK_T(pthread_create (&tid[i], NULL, (pf_t)monThread,
                                (void *)i), "pthread_create()");

    printf("\tattente de la terminaison des threads\n");

    for (i = 0; i < NBMAX_THREADS; i++)
        CHECK_T(pthread_join (tid[i], NULL), "pthread_join()");
    return EXIT_SUCCESS;
}

void monThread (long no)
{
    long i;

    for (i = 0; i <= no; i++) printf(" ");
    printf("Début du thread n°%ld\n", no + 1);
    sleep (rand() % DUREE_MAX + 1);
    for (i = 0; i <= no; i++) printf(" ");
    printf("Fin du thread n°%ld\n", no + 1);
    pthread_exit(NULL);
}

void bye (void)
{
    printf("Fin du processus %d\n", getpid());
}
```

exécution du programme :

```
elMehdi_C@UNSH:~/Bureau/TP_threads/Mood/3_threads$ ./thread3
Le processus 10387 va créer 4 threads
Début du thread n°1
Début du thread n°2
    attente de la terminaison des threads
Début du thread n°4
Début du thread n°3
Fin du thread n°3
Fin du thread n°2
Fin du thread n°4
Fin du thread n°1
Fin du processus 10387
```

**Question 6 :**

De la même manière, on utilise cette fois un pointeur vers une variable "status" dans la fonction main() qui sera utilisée par : *pthread\_join()* pour récupérer la variable retournée "retVal" dans *pthread\_exit()*. (On oubliera de désallouer l'adresse allouée dans la boucle).

```
int main(int argc, char *argv[])
{
    pthread_t tid[NBMAX_THREADS];
    long i;
    long * status;

    atexit(bye);
    printf("Le processus %d va créer %d threads\n",
           getpid(), NBMAX_THREADS);

    for (i = 0; i < NBMAX_THREADS; i++)
        CHECK_T(pthread_create (&tid[i], NULL, (pf_t)monThread,
                                (void *)i), "pthread_create()");

    printf("\tattente de la terminaison des threads\n");

    for (i = 0; i < NBMAX_THREADS; i++) {
        CHECK_T(pthread_join (tid[i], (void **) &status), "pthread_join()");
        printf("--> fin du thread n°%ld avec le code %ld\n", i + 1, *status);
        free(status);
    }
    return EXIT_SUCCESS;
}

void monThread (long no)
{
    long i;
    long * retVal = (long *) malloc (sizeof(long));

    for (i = 0; i < no; i++) printf(" ");
    printf("Début du thread n°%ld\n", no + 1);
    sleep (rand() % DUREE_MAX + 1);
    for (i = 0; i < no; i++) printf(" ");
    printf("Fin du thread n°%ld\n", no + 1);

    *retVal = (long) no + 1;
    pthread_exit((void *) retVal);
}

void bye (void)
{
    printf("Fin du processus %d\n", getpid());
}
```

exécution :

```
etMehdi_C@QNSH:~/Bureau/TP_threads/mood/5_threads$ ./thread4
Le processus 11345 va créer 4 threads
Début du thread n°1
  Début du thread n°2
    Début du thread n°3
      attente de la terminaison des threads
        Début du thread n°4
          Fin du thread n°4
        Fin du thread n°2
      Fin du thread n°3
    Fin du thread n°1
  --> fin du thread n°1 avec le code 1
  --> fin du thread n°2 avec le code 2
  --> fin du thread n°3 avec le code 3
  --> fin du thread n°4 avec le code 4
Fin du processus 11345
```