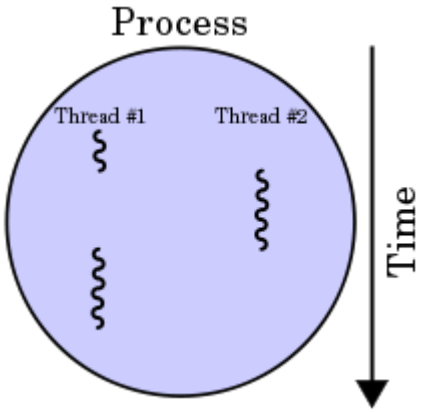
Communication entre deux threads

Introduction : *Un* ***thread*** *ou* ***fil (d'exécution)*** *ou* ***tâche*** *(terme et définition normalisés par ISO/CEI 2382-7:2000 ; autres appellations connues :* ***processus léger****,* ***fil d'instruction****,* ***processus allégé****,* ***exétron****, voire* ***unité d'exécution***[1] *ou* ***unité de traitement***[2,3][réf. nécessaire]*) est similaire à un processus car tous deux représentent l'exécution d'un ensemble d'instructions du langage machine d'un processeur. Du point de vue de l'utilisateur, ces exécutions semblent se dérouler en parallèle. Toutefois, là où chaque processus possède sa propre mémoire virtuelle, les threads d'un même processus se partagent sa mémoire virtuelle. Par contre, tous les threads possèdent leur propre pile d'appel.*



Un processus avec deux *threads*.

Enoncé : *Lorsqu'un programme a été décomposé en plusieurs threads, ceux-ci ne sont en général pas complètement indépendants et ils doivent communiquer entre eux. Cette communication entre threads est un problème complexe comme nous allons le voir.*

Et voici le programme en C qui permet la communication entre deux threads.

```
#include <stdio.h>

#include <pthread.h>

Void myfunc1 (void *ptr);

Void myfunc2 (void *ptr);


Int main(int argc,char *argv[ ]){
```

```
    Pthread_t thread1;
    Pthread_t thread2;

    Char *msg1 = "thread 1";
    Char *msg2 = "thread 2";

Pthread_create(&thread1 , NULL ,  (void *) &myfunc1 , (void *) msg1);
Pthread_create(&thread2 , NULL , (void *) &myfunc2 , (void *) msg2);

Pthread_join(thread1,NULL);
Pthread_join(thread2 , NULL);

Return 0;
}

Void myfunc1 (void *ptr )
{
Char *msg = (char *)  ptr;
```

```
Void myfunc2 (void *ptr )
{
Char *msg =(char *) ptr;
Printf("%s\n",msg );
Pthread_exit(0);
}
```