```c
/* gestion  des threads sous UNIX */

/* TP4  exercice  */
/* essai de _create, _join, _exit et return */

# include <stdlib.h>
# include <unistd.h>
# include <pthread.h>
# include <stdio.h>
# include <string.h>

#define NB_THREADS  2

void * traitThread1(void *);
void * traitThread2(void *);

int main(void)
{
        int err;
        pthread_t  idThread[NB_THREADS] ;
        void * ptrRetVal;

        int unNombre=1234;
        char uneChaine[ ]={"azerty"};

        /* main thread */

        printf("\nmain --> PID= %d\n",(int)getpid());

        /* creation du thread1 */

        err = pthread_create(&idThread[0], NULL, traitThread1, (void *)& unNombre) ;
        if (err != 0)
            { perror("echec pthread_create");exit(1); }

        /* creation du thread2 */

        err = pthread_create(&idThread[1], NULL, traitThread2, (void *)uneChaine) ;
        if (err != 0)
            { perror("echec pthread_create");exit(2); }

        /* attente de la terminaison du thread1 */

            err=pthread_join( idThread[0], &ptrRetVal);
            if (err != 0 )
            {   perror("echec join"); }
            else
            {
               printf("\nTerminaison du thread de TID= %d\n", idThread[0]);
               if (  ptrRetVal == PTHREAD_CANCELED )
               {
                 printf("Fin anormale: thread annulé\n");
               }
               else
               {
                 printf("Fin normale: nombre retourne = %d\n", *((int *)ptrRetVal) );
               }
            }

        /* attente de la terminaison du thread2 */

            err=pthread_join( idThread[1], &ptrRetVal);
            if (err != 0 )
            { perror("echec join"); }
            else
            {
               printf("\nTerminaison du thread de TID= %d\n", idThread[1]);
               if (  ptrRetVal == PTHREAD_CANCELED )
               {
                 printf("Fin anormale: thread annulé\n");
               }
```

```
                else
                {
                  printf("Fin normale: chaine retournee = %s\n", (char *)ptrRetVal);
                }
            }

        /* fin du main thread */

        pthread_exit(NULL);
}

/* traitement du thread1 */

void * traitThread1(void *nbre)
{
        static int valeur;     /* attribut static absolument necessaire !!! */

        printf("\n***thread1 --> TID= %d\n", pthread_self());
        valeur= *((int *) nbre);
        printf("\n***thread1 --> nombre recu= %d\n", valeur);
        valeur=valeur * 2;
        pthread_exit((void *)(&valeur));
}


/* traitement du thread2 */

void * traitThread2(void *chaine)
{
        static char mess[128]={"message de thread2: "};    /* attribut static absolument
necessaire !!! */

        printf("\n***thread2 --> TID= %d\n", pthread_self());
        printf("\n***thread2 --> chaine recue= %s\n", (char *)chaine);
        strcat(mess, chaine);

        return((void *)mess);
}
```