Lab 5: Creació de *Threads*

Objectius:

- Entendre el concepte de programació multifil ("*multithread*").
- Aprendre les comandes bàsiques de creació i de control de *threads*.

Fitxers dèxemple:

`exemples_P2.1/multifil.c`

Funcions bàsiques de llibreria:

```
int pthread_create(pthread_t *thread,
                   pthread_attr_t *attr,
                   void * (*start_routine)(void *),
                   void *arg);
```

La funció `pthread_create` crea un nou fil que executarà la rutina `start_routine`, passant-li com a argument el paràmetre `arg`. Si tot ha anat bé, la funció retorna zero i copia l'identificador del fil que s'ha creat dins de la variable `thread` (pas per referència). El paràmetre `attr` serveix per especificar les característiques d'execució del nou fil, però habitualment es posa a `NULL` per a què el SO utilitzi els valors per defecte.

```
int pthread_join(pthread_t thread, void **status);
```

La funció `pthread_join` bloqueja l'execució del fil que la invoca, a l'espera de que el fil especificat en el paràmetre `thread` acabi la seva execució. Quan el fil indicat acaba, el valor que retorna es copia dins de la variable `status` (pas per referència).

```
pthread_t pthread_self(void);
```

La funció `pthread_self` retorna l'identificador del fil que la invoca.

```
pthread_t pthread_exit(void *value_ptr);
```

La funció `pthread_exit` acaba immediatament l'execució del fil que la invoca, on `value_ptr` serà el codi de retorn del fil.

Programa dèexemple:

```
/*****************************************************************/
/*                           multifil.c                          */
/*                                                               */
/*    Compilar i executar:                                       */
/*          $ gcc -Wall  multifil.c -o multifil  -lpthread       */
/*          $ ./multifil  num_threads  n_vegades  n_lletres      */
/*                                                               */
/*    Executa tants threads com indica el parametre num_threads, */
/*    on cada thread visualitza un caracter identificatiu        */
/*    (thread 0 -> 'a', thread 1 -> 'b', ...) tantes vegades com */
/*    indiqui el segon parametre n_vegades, esperant un temps    */
/*    aleatori entre dues visualitzacions; el programa acaba quan*/
/*    s'han escrit un total de n_lletres entre tots els fils.    */
/*****************************************************************/

#define _REENTRANT
#include <pthread.h>
#include <stdio.h>
#include <stdint.h>   /* definició de intptr_t per màquines de 64 bits */
#include <unistd.h>
#include <stdlib.h>

#define MAX_THREADS     10
#define MAX_VEGADES     50
#define MAX_LLETRES     100
                              /* Variables Globals */
pthread_t tid[MAX_THREADS];   /* taula d'identificadors dels threads */
int lletres;                  /* numero de lletres escrites */
int max_iter;                 /* numero maxim d'iteracions */

/* escriure una marca corresponent al num. de thread ('a'+i_thr) en */
/* intervals aleatoris entre 1 i 3 segons, les vegades que indiqui */
/* la variable global max_iter */
void * caracter(void *i_thr)
{
  int i;
  char ic='a'+ (intptr_t) i_thr;

  for (i=0; i < max_iter; i++)      /* per a totes les vegades */
  {
    if (lletres > 0)                /* si falten lletres */
    {
      sleep(1 + rand() % 3);              /* dormir entre 1 i 3 segons */
      printf("%c",ic);              /* escriure marca del thread */
      lletres--;                    /* una lletra menys */
     }
     else pthread_exit((void *)(intptr_t) i);   /*sino, forcar sortida thread */
  }
  return((void *) (intptr_t) i); /*retorna numero lletres que ha impres el fil*/
}
```

```
int main(int n_args, char * ll_args[])
{
    int i,n,t,t_total,n_thr;

    if (n_args != 4)
    {   fprintf(stderr,"comanda: multifil num_threads max_iter n_lletres\n");
        exit(1);
    }
    n_thr = atoi(ll_args[1]);       /* convertir arguments a num. enter */
    max_iter = atoi(ll_args[2]);
    lletres = atoi(ll_args[3]);
    if (n_thr < 1) n_thr = 1;       /* i filtrar el seu valor */
    if (n_thr > MAX_THREADS) n_thr = MAX_THREADS;
    if (max_iter < 1) max_iter = 1;
    if (max_iter > MAX_VEGADES) max_iter = MAX_VEGADES;
    if (lletres < 1) lletres = 1;
    if (lletres > MAX_LLETRES) lletres = MAX_LLETRES;

    srand(getpid());          /* inicialitza la "llavor" dels aleatoris */
    setbuf(stdout,NULL);      /* anular buffer de sortida estandard */
    printf("Main thread del proces (%d) : ", getpid());
    n = 0;
    for ( i = 0; i < n_thr; i++)
    {
      if (pthread_create(&tid[n],NULL,caracter,(void *)(intptr_t) i) == 0)
            n++;
    }
    printf("he creat %d threads, espero que acabin!\n\n",n);

    t_total = 0;
    for ( i = 0; i < n; i++)
    {
      pthread_join(tid[i], (void **)&t);
      printf("el thread (%d) ha escrit %d lletres\n",i,t);
      t_total += t;
    }
    printf("\nJa han acabat tots els threads creats!\n");
    printf("Entre tots els threads han escrit %d lletres.\n",t_total);
    return(0);
}
```

Exercici:

Realitzar la pràctica 2.1.