

# Fonaments dels Sistemes Operatius (FSO)

Departament d'Informàtica de Sistemes i Computadors (DISCA)

*Universitat Politècnica de València*

Bloc temàtic 2: Gestió de processos

Unitat temàtica 5

SUT5: Programació amb fils

POSIX

fSO

DISCA



UNIVERSITAT  
POLITÈCNICA  
DE VALÈNCIA

- **Objectius**

- Treballar les **crides al sistema POSIX** relacionades amb la **creació i gestió bàsica de fils**
- Treballar el concepte de **condició de carrera** per comprendre la problemàtica que introdueix, la seua existència, en la **programació concurrent**

- **Bibliografia**

- [UNIX Programación Práctica](#)”, Kay A. Robbins, Steven Robbins. Prentice Hall. ISBN 968-880-959-4

- **Contingut**
  - Introducció
  - Creació de fils
  - Terminació i espera de fils
  - Identificació de fils
  - Condició de carrera

- **Procés POSIX**

- Es crea un **fil** (thread) **inicial** que executa la funció **main()**
  - Qualsevol fil pot crear més fils per executar altres funcions dins de l'espai d'adreces del procés
- **Tots els fils** d'un procés estan **al mateix nivell**
  - Això significa que són "germans", a diferència dels processos amb relació "pare-fill"
- Els **fils d'un procés comparteixen les variables i recursos globals** (fitxers, manejadors de senyals, etc.) del procés
  - A més, cadascun té una **còpia privada dels seus paràmetres inicials i de les variables locals** de la funció que executa

- Crides bàsiques per a **gestió de fils/threads**

	Fils d'execució (pthreads)
<b>pthread_create</b>	Crea un fil per a executar una funció específica
<b>pthread_attr_init</b>	Inicialitza un objecte atribut de fil als valors per omissió
<b>pthread_attr_destroy</b>	Destruïx l'objecte atribut de fil
<b>pthread_join</b>	Espera la terminació del fil especificat
<b>pthread_exit</b>	Termina l'execució del fil invocador
<b>pthread_self</b>	Torna la identitat del fil que l'invoca
<b>pthread_attr_setdetachstate</b>	Modifica l'atribut de desconnectat
<b>pthread_attr_getdetachstate</b>	Consulta l'atribut de desconnectat

- **Contingut**
  - Introducció
  - **Creació de fils**
  - Terminació i espera de fils
  - Identificació de fils
  - Condició de carrera

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

- `pthread_create ()`
  - Crea un **nou fil en estat preparat**
  - El fil creat i el creador **competeixen per la CPU** segons la política de planificació del sistema
  - Pot ser invocada per qualsevol fil del procés (no només pel "fil inicial") per a crear un altre fil
- arguments
  - **attr** és l'atribut que conté les característiques del nou fil
  - **start\_routine** és la funció que executarà el fil
  - **arg** és un punter als paràmetres inicials del fil
  - en **thread** es retorna l'identificador del fil creat

- Atributs per a creació de fils :

```
int pthread_attr_init(pthread_attr_t *attr);  
int pthread_attr_destroy(pthread_attr_t *attr);
```

on

- **attr** és l'atribut ha de ser creat/destruït

- Aquestes crides:

- Creen/destrueixen un atribut de creació de fils
- La funció "**init**" inicialitza **attr** amb els valors per defecte
  - Aquests valors es poden modificar amb funcions específiques
- Es poden crear múltiples fils amb la mateixa variable **attr**



- Per a modificar els atributs de creació de fils:

```
int pthread_attr_setdetachstate(pthread_attr_t *attr,  
                                int detachstate);  
int pthread_attr_getdetachstate(const pthread_attr_t *attr,  
                                int *detachstate);
```

- on
  - detachstate indica si un altre fil podrà esperar a la finalització d'aquest fil (mitjançant una instrucció `pthread_join`):
    - `PTHREAD_CREATE_JOINABLE`
    - `PTHREAD_CREATE_DETACHED`

- Exemple: Hello World. Sense espera de fils

```
#include <stdio.h>
#include <string.h>
#include <pthread.h>
```

```
void *My_Print(void *ptr )
{ char *message;
  message = (char *) ptr;
  write(1,message,strlen(message));
}

int main()
{
  pthread_t thread1, thread2;
  pthread_attr_t attr;

  pthread_attr_init(&attr);
  pthread_create(&thread1, &attr, My_Print, "Hello ");
  pthread_create(&thread2, &attr, My_Print, " World\n");

  return 0;
}
```

```
//fitxer: pthread_hello.c
//compilar:gcc pthread_hello.c -o pthread_hello -lpthread
//mostrar fils en el shell: ps -lT
```

- **Contingut**
  - Introducció
  - Creació de fils
  - **Terminació i espera de fils**
  - Identificació de fils
  - Condició de carrera

- **Finalització de fils POSIX**

- Un fil finalitza (voluntariament) la seua execució quan:
  - Finalitza l'execució de les instruccions de la seua funció, o be
  - invoca `pthread_exit`

```
#include <pthread.h>
```

```
int pthread_exit(void *exit_status);
```

on

- **exit\_status** és un punter a una variable mitjançant la qual un fil que finalitza (**pthread\_exit**) comunica un valor a altre que està esperant la seua terminació (**pthread\_join**)
- la finalització de l'últim fil d'un procés finalitza el procés
- Si `main()` no finalitza amb `pthread_join` el procés acabarà i, per tant, els fils del mateix que es troben en execució també acabaran

- Espera de fils:
  - Mitjançant la crida `pthread_join` es pot **esperar la finalització d'un fil**, sempre que s'haja creat amb l'atribut `PTHREAD_CREATE_JOINABLE`

```
#include <pthread.h>
```

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

- `exit_status` és el valor que el fil terminat amb `exit` comunica al fil que invoca a `pthread_join`
- Se suspén l'execució del fil que invoca la crida, fins que el fil especificat (**thread**) finalitze

- Exemple

```
...  
void *funcio(void *p) {  
    printf("Sóc un fil feliç!\n");  
    sleep(10);  
}  
...  
int main( void ) {  
    pthread_t      id_fil;  
    pthread_attr_t atributs;  
  
    printf("Fil principal: principi\n");  
    pthread_attr_init(&atributs);  
    pthread_create(&id_fil, &atributs, funcio, NULL);  
    printf("Fil principal: He creat un germà\n");  
    pthread_join(id_fil, NULL);  
    printf("Fil principal: Acabe!");  
}
```

¡Avis! Hem de declarar la variable de tipus fil. Una variable per cada fil a crear

¿Quin seria el resultat d'execució si eliminem el pthread\_join?

- **Contingut**
  - Introducció
  - Creació de fils
  - Terminació i espera de fils
  - **Identificació de fils**
  - Condició de carrera

- Identificació de fils:

```
pthread_t pthread_self(void);  
int pthread_equal(pthread_t th1, pthread_t th2);
```

on

- **pthread\_self** torna l'identificador intern del fil invocant
- donat que aquest identificador pot no ser escalar, la funció **pthread\_equal** aprofita per a comparar dos identificadors.
- La funció retorna
  - **zero** (0) si dos identificadors **no** són iguals
  - Un valor distint si són iguals els identificadors



Exemple: creació periòdica de fils

```
int main ()
{
    pthread_t t1,t2;
    pthread_attr_t attr;
    int period1=1, period2=2;

    if (pthread_attr_init(&attr) != 0) {
        printf("Error: atributtes\n");
        exit(1);
    }

    if (pthread_create(&t1, &attr,func_period, &period1) != 0) {
        printf("Error: creating first pthread\n");
        exit(1);
    }

    if (pthread_create(&t2, &attr,func_period, &period2) != 0) {
        printf("Error: creating second pthread\n");
        exit(1);
    }

    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
}
```

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

void *func_period (void *arg) {
    int period, i;
    period= *((int *)arg);
    for (i=0; i<10; i++) {
        printf("Pthread(period %d):", period);
        printf(" %ld\n", (long) pthread_self());
        sleep (period);
    }
}
```

```
//fitxer: th_periodic.c
//compilar:gcc th_periodic.c -o th_periodic -lpthread
//mostrar fils en el shell: ps -lT
```

- **Contingut**
  - Introducció
  - Creació de fils
  - Terminació i espera de fils
  - Identificació de fils
  - **Condió de carrera**

- Exemple: “globalvar.c”

- Procés que incrementa en 40000000 unitats una variable global
- El resultat d’execució correcte seria: **Globalvariable=40000000**

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int Globalvariable;

int main()
{   int i;
    long iterations = 40000000;
    for (i=0; i<(iterations); i++){
        variableGlobal ++;
    }
    printf("Globalvariable= %d\n",Globalvariable);

    return 0;
}
```

//fitxer: globalvar.c  
//compilar:gcc globalvar.c -o globalvar

¿Resultat de  
l’execució??

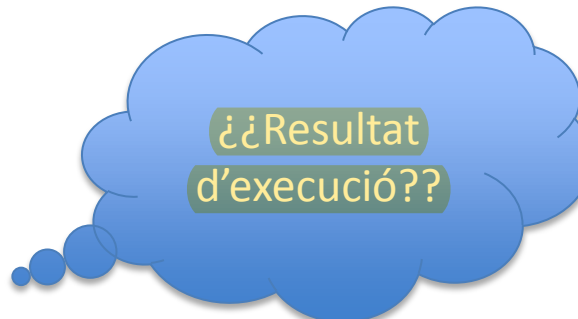
- Exemple “**race\_condition.c**”

- Versió concurrente de “globalvar.c”
- **Dos fils** colaboren per a **incrementar variable**
- Cada fil fa **20000000 operacions**
- Resultat d’execució esperat  
**Globalvariable=40000000**

```
//fitxero: race_condition.c  
//compilar: gcc race_condition.c -o race_condition -lpthread
```

```
int main()  
{ long iterations = 20000000;  
  pthread_t t1, t2;  
  pthread_attr_t attr;  
  
  pthread_attr_init(&attr);  
  pthread_create(&t1, &attr, Addition, &iterations);  
  pthread_create(&t2, &attr, Addition, &iterations);  
  
  pthread_join(t1, NULL);  
  pthread_join(t2, NULL);  
  printf("Globalvariable= %d\n", Globalvariable);  
  return 0;  
}
```

```
#include <stdio.h>  
#include <pthread.h>  
  
int Globalvariable;  
  
void *Addition(void *ptr )  
{int i, aux_variable;  
  int *iter = (int *)ptr;  
  for (i=0; i<*iter; i++){  
    aux_variable = Globalvariable;  
    aux_variable++;  
    Globalvariable = aux_variable;  
  }  
}
```



¿¿Resultat  
d’execució??