E

laSalle ENG
Universitat Ramon Llull

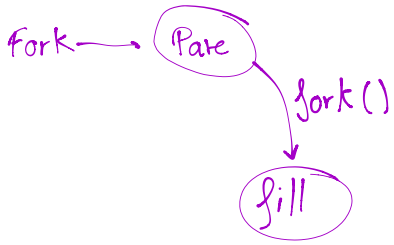
# Sistemes Operatius: threads

# Threads

laSalle ENG
Universitat Ramon Llull

- Generen fils d'execució.
- Comparteixen molts recursos (és un risc).
- Creació de threads: **pthread_create()**
- Espera finalització: **pthread_join()** equival·lent a wait().
- Si el pare mor, tots els threads es destrueixen.
- Compilar amb -lpthread.

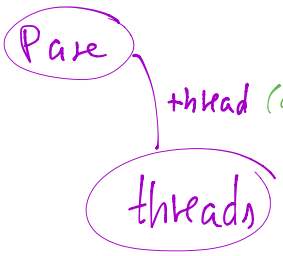
diferències FORKS VS THREAD




És lliat compartir info entre pare i fill

Si mor el pare, el fill és orfe!

Thread:




←No té identitat pròpia

Ens és fàcil compartir info → a través de var global tots els threads poden accedir-hi

Un thread NO és un fill, tots els threads són "germans" amb el mateix pare

#include <pthread.h>

# 1. Exemple 1

laSalle ENG
Universitat Ramon Llull

```
void * printf_xs(void *unused){
while (1){
  fputc('x',stderr);}
return NULL;
}

int main() {
pthread_t thread_id;
int estat;
estat=pthread_create(&thread_id,NULL,printf_xs,NULL);
if (estat!=0) exit(-1);
while (1)
  fputc('o',stderr);
return 0;
}
```

obligatori void * ← Aquesta és la funció que farà de thread ← la podem llençar tants cops com vulguem

tipus thread (de moment sempre NULL)

↳ Nom funció thread

Paràmetre que li passem (en aquest exemple CAP)

si retorna != 0 és que NO s'ha creat

Millor write, això és cutre!

# NOMÉS PODEM PASSAR 1 ÚNIC PARÀMETRE

quin thread esperem
↓
pthread_join (t1, &res)

↓

Variable que reb el que el thread retorna

↳ aquesta variable ha de ser void * res; ← després com que ja sabem que rebem fem x = (int) res; i ale ja l'hem castejat.

un pthread_join per thread, si volem fer 2 esperes fem 2 pthreadjoin

# 2. Exemple 2

laSalle ENG
Universitat Ramon Llull

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

void *threadFunc (void *arg){

char *s = (char *) arg;
printf ("%s", s);
return (void *) strlen (s);

}
```

# 2. Exemple 2

laSalle ENG
Universitat Ramon Llull

```
int main (){
  pthread_t t1;
  void *res;
  int s;

s = pthread_create (&t1, NULL, threadFunc, "Hello world\n");
if (s != 0){
   printf("pthread_create\n");
   exit (EXIT_FAILURE);
}
printf ("Missatge des del main()\n");
s = pthread_join (t1,&res);
if (s != 0){
   printf("pthread_join\n");
   exit (EXIT_FAILURE);
}
printf ("Thread retorna %ld\n", (long)res);
exit (EXIT_SUCCESS);
}
```

# 2. Exemple 2 (v2)

laSalle ENG
Universitat Ramon Llull

```
int main (){
  pthread_t t1;
  void *res;


pthread_create (&t1, NULL, threadFunc, "Hello world\n");

printf ("Missatge des del main()\n");

pthread_join (t1,&res);

printf ("Thread retorna %ld\n", (long)res);
exit ();

}
```

# 3. Exemple 3: paràmetres

laSalle ENG
Universitat Ramon Llull

```
static void *threadFunc (void *arg){
int *numeret = (int *) arg;

*numeret=*numeret+5;

return (void *) arg;
}
```

# 3. Exemple 3: paràmetres

laSalle ENG
Universitat Ramon Llull

```
int main (){
pthread_t t1;
void *res;
int s,x;

  x=7;
  printf ("Valor de x=%d\n", x);
  s = pthread_create (&t1, NULL, threadFunc, &x);
  if (s != 0){
    printf("pthread_create\n");
    exit (EXIT_FAILURE);
  }
  s = pthread_join (t1, &res);
  if (s != 0){
    printf("pthread_join\n");
    exit (EXIT_FAILURE);
  }
  printf ("Thread retorna %X\n", (long) res);
  printf ("Valor de x=%d\n", x);
  exit (EXIT_SUCCESS);
}
```

# 3. Exemple 3: paràmetres (v2)

laSalle ENG
Universitat Ramon Llull

```
int main (){
pthread_t t1;
void *res;
int x;

  x=7;
  printf ("Valor de x=%d\n", x);

  pthread_create (&t1, NULL, threadFunc, &x);

  pthread_join (t1, &res);

  printf ("Thread retorna %X\n", (long) res);
  printf ("Valor de x=%d\n", x);

  exit ();
}
```

# 4. Exemple 4: threads i variables

laSalle ENG
Universitat Ramon Llull

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

int y;

static void *threadFunc (void *arg){
int *numeret = (int *)arg;

*numeret=*numeret+5;
y=y+1;
printf("Des del thread y=%d\n",y);

return (void *) arg;
}
```

# 4. Exemple 4: threads i variables

laSalle ENG
Universitat Ramon Llull

```
int main (){
pthread_t t1;
void *res;
int s,x;

x=7;
y=5;
printf ("[main]Valor Inicial x=%d\n",x);
printf("[main] Valor Inicial y=%d\n",y);
s = pthread_create (&t1, NULL, threadFunc, &x);
if (s != 0){
   printf("pthread_create\n");
   exit (EXIT_FAILURE);
}
```

# 4. Exemple 4: threads i variables

laSalle ENG
Universitat Ramon Llull

```
y=y+2;
printf("[main] Valor Mig y=%d\n",y);

s = pthread_join (t1, &res);
if (s != 0){
    printf("pthread_join\n");
    exit (EXIT_FAILURE);
}
printf ("Thread retorna %X\n", (long) res);
printf("[main] Valor Final x=%d\n",x);
printf("[main] Valor Final y=%d\n",y);
exit (EXIT_SUCCESS);}
```

# 4. Exemple 4: threads i variables (v2)

laSalle ENG
Universitat Ramon Llull

```
int main (){
pthread_t t1;
void *res;
int x;

x=7;
y=5;
printf("[main]Valor Inicial x=%d\n",x);
printf("[main]Valor Inicial y=%d\n",y);
pthread_create (&t1, NULL, threadFunc, &x);
y=y+2;
printf("[main] Valor Mig y=%d\n",y);
pthread_join (t1, &res);
printf ("Thread retorna %X\n", (long) res);
printf("[main] Valor Final x=%d\n",x);
printf("[main] Valor Final y=%d\n",y);
exit ();
}
```