



# SISTEMAS OPERATIVOS

3004610 - 1

**German Sánchez Torres, I.S., M.Sc., Ph.D.**

Profesor, Facultad de Ingeniería - Programa de Sistemas

Universidad del Magdalena, Santa Marta.

Phone: +57 (5) 4214079 Ext 1138 - 301-683 6593

Edificio Docente, Cub 3D401.

Email: [sanchez.gt@gmail.com](mailto:sanchez.gt@gmail.com) - [gsanchez@unimagdalena.edu.co](mailto:gsanchez@unimagdalena.edu.co)



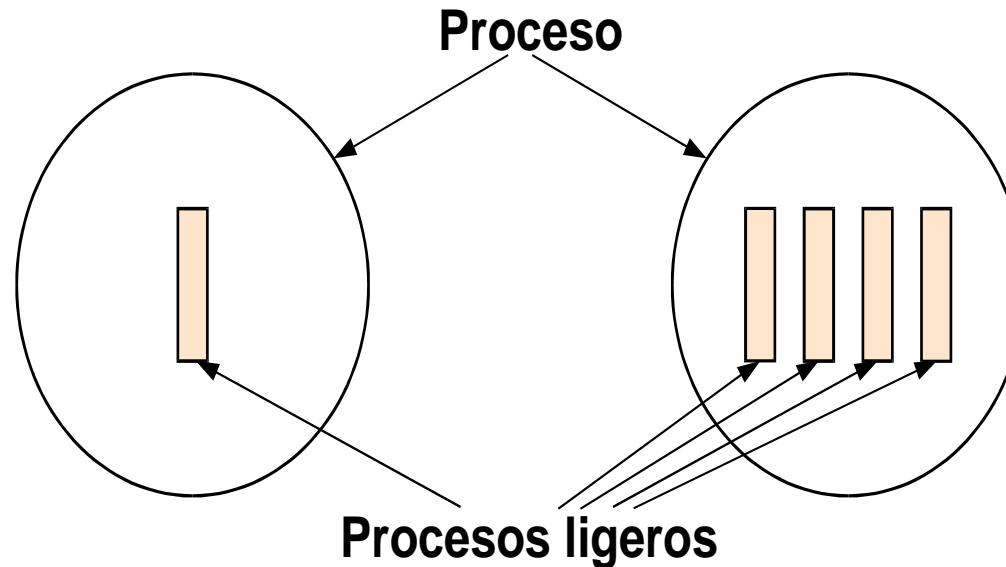
# Hilos de Ejecución - thread

**SISTEMAS OPERATIVOS**

# PROCESOS LIGEROS

## • INTRODUCCION

- Un proceso ligero (*thread o hebra*) es un programa en ejecución que **comparte la imagen de la memoria** y otras informaciones con otros procesos ligeros.



# PROCESOS LIGEROS

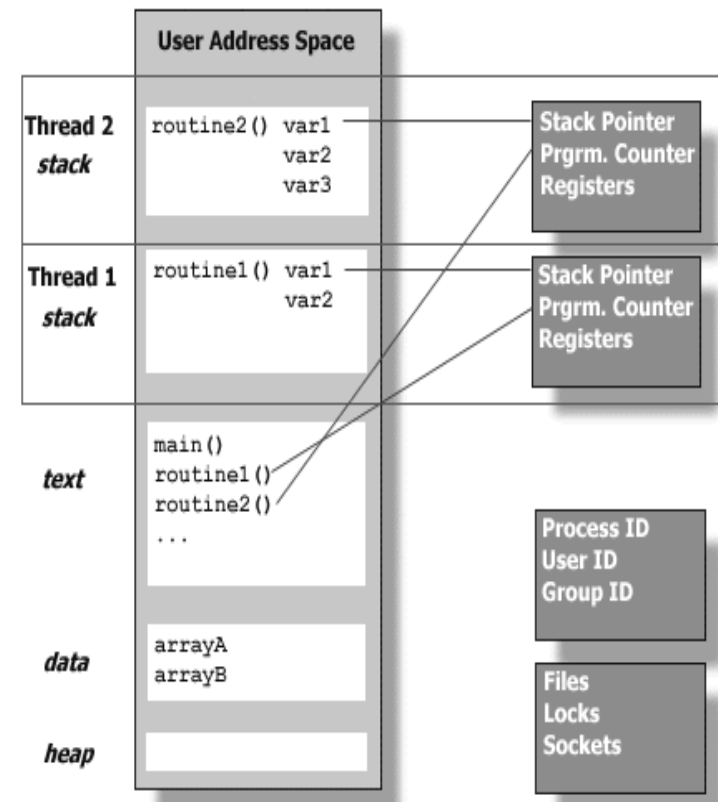
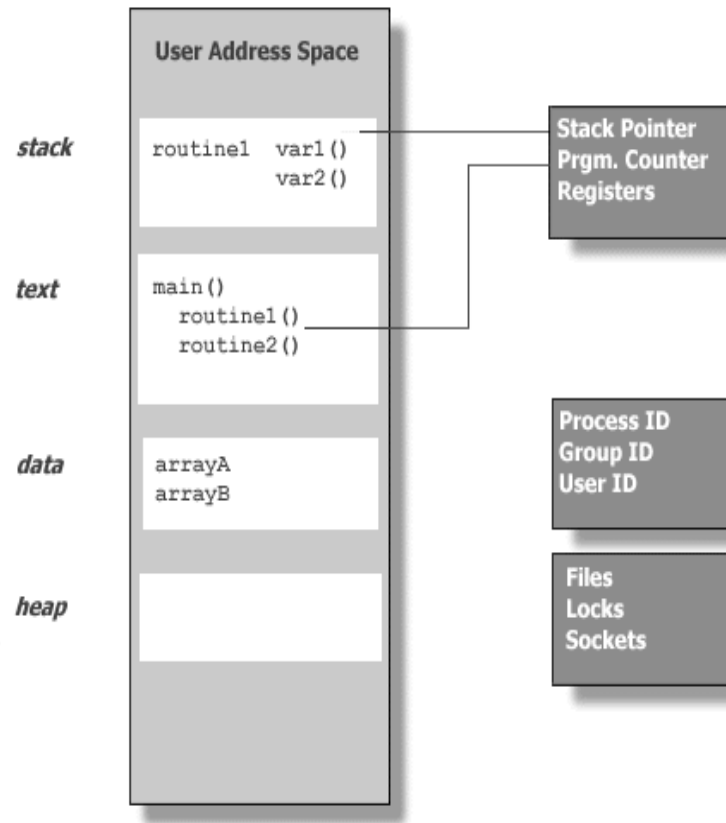
## Información compartida

- **Espacio de memoria**
- **Variables globales**
- **Archivos abiertos**
- **Procesos hijos**
- **Temporizadores**
- **Señales y semáforos**
- **Contabilidad**

## Información no compartida

- **Contador de programa**
- **Pila**
- **Registros del procesador**
- **Estado del proceso ligero**

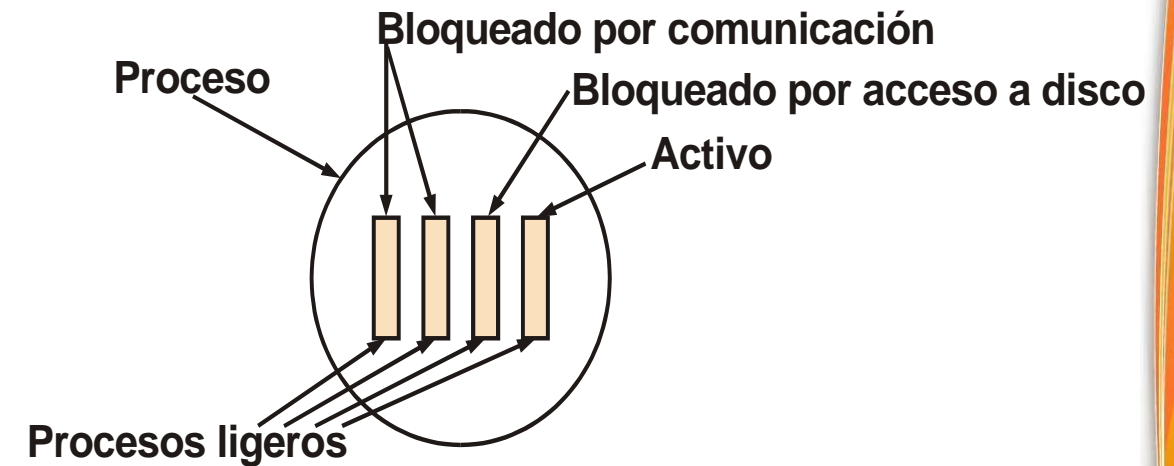
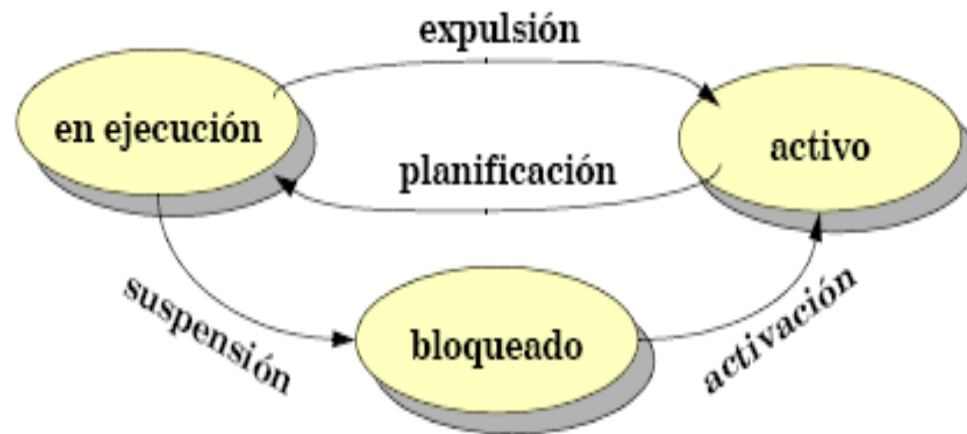
# PROCESOS LIGEROS



# PROCESOS LIGEROS

## ESTADO DE LOS PROCESOS LIGEROS

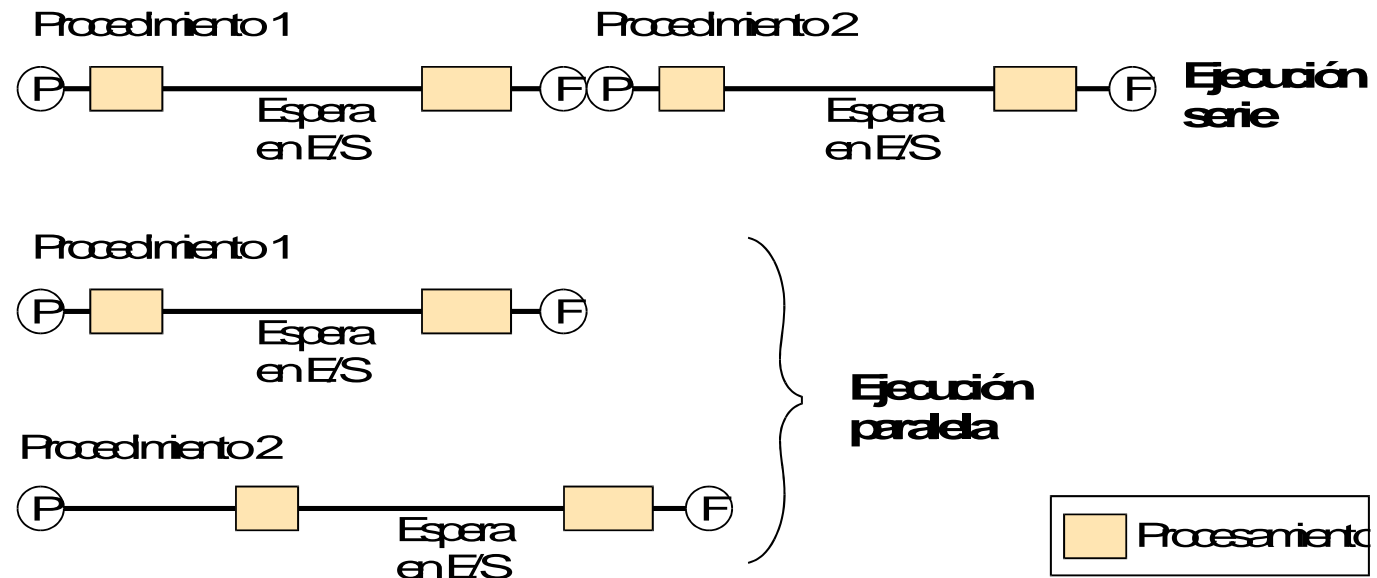
- Un proceso ligero puede estar **ejecutando, listo o bloqueado**.





## PARALELISMO

Los procesos ligeros permiten paralelizar una aplicación.



# Tipos de un hilo



- Los hilos Pthread tienen dos tipos
  - *Acoplado (joinable)*
  - *Desacoplado (detached)*
- Por defecto los hilos son acoplados
  - Los recursos se mantienen hasta el **pthread\_join**
- Los hilos desacoplados no pueden unirse
  - Los recursos pueden reclamarse en la terminación
  - No se pueden resetear a ser *unibles*



# pthread\_create



```
int pthread_create(tid, attr, function, arg) ;
```

**pthread\_t \*tid**

- descriptor del hilo creado

**const pthread\_attr\_t \*attr**

- atributos del hilo a crearse

**void \*(\*function) (void \*)**

- función que será mapeada al hilo

**void \*arg**

- argumento que se envía a la función

# pthread\_create Detalles

- Inicia un hilo ejecutando la función
- Descriptor del hilo retornado por medio de la estructura **pthread\_t**
  - Especifica **NULL** para usar los atributos por default
- Un solo argumento enviado a la función
  - Si no tiene argumentos, especifica **NULL**
- Se debe verificar los códigos de error!

**EAGAIN** – recursos insuficientes para crear el hilo  
**EINVAL** – atributo inválido

# Esperando un Thread

```
int pthread_join(tid, val_ptr);
```

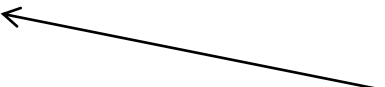
`pthread_t tid`

- manejador de un hilo *a esperar*

`void **val_ptr`

- valor de salida devuelto por un hilo

`int pthread_create(tid, attr, function, arg);`

An arrow pointing from the 'tid' parameter in the 'pthread\_create' function signature to the 'tid' parameter in the 'pthread\_join' function signature, indicating that the thread ID returned by create is used by join.

# pthread\_join Detalles



- Un hilo espera a que un hilo con descriptor **tid** termine
  - Solo espera a que un hilo se *una*
  - El hilo debe ser *acoplado*
- Un valor de salida se devuelve del hilo unido
  - Tipo devuelto es (**void \***)
  - Usar **NULL** si no se espera un valor de retorno

**ESRCH** - hilo (**pthread\_t**) no encontrado

**EINVAL** - hilo (**pthread\_t**) no unible



Ejemplos

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

/* Global variable: accessible to all threads */
int thread_count;

void *Hello(void* rank); /* Thread function */

int main(int argc, char* argv[]) {
    long thread; /* Use long in case of a 64-bit system */
    pthread_t* thread_handles;

    /* Get number of threads from command line */
    thread_count = strtol(argv[1], NULL, 10);

    thread_handles = malloc (thread_count*sizeof(pthread_t));
```

```
for (thread = 0; thread < thread_count; thread++)
    pthread_create(&thread_handles[thread], NULL,
        Hello, (void*) thread);

printf("Hello from the main thread\n");

for (thread = 0; thread < thread_count; thread++)
    pthread_join(thread_handles[thread], NULL);

free(thread_handles);
return 0;
} /* main */

void *Hello(void* rank) {
    long my_rank = (long) rank; /* Use long in case of 64-bit system */

    printf("Hello from thread %ld of %d\n", my_rank, thread_count);

    return NULL;
} /* Hello */
```



# Compilación y ejecución de programas con hilos POSIX

```
gcc -o PthreadExample PthreadExample.c -lpthread
```

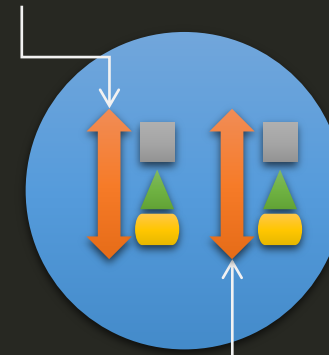
```
./PthreadExample
```

```

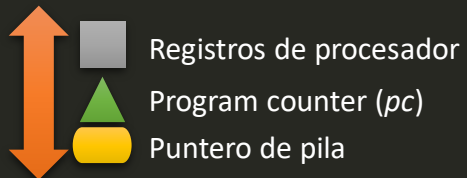
1  /*
2  Ejemplo Hilo1.c
3  */
4  #include <stdio.h>
5  #include <unistd.h>
6  #include <pthread.h>
7
8  void* funcion_manaja_hilo(void *);
9
10
11 int main(){
12     pthread_t pidhilo;
13     pthread_create(&pidhilo, NULL, funcion_manaja_hilo, NULL);
14     printf("Hilo principal (idthread -> [%ld])\n", (long int) pthread_self());
15     pthread_join(pidhilo, NULL);
16     return 0;
17 }
18
19 void* funcion_manaja_hilo(void *param){
20     printf("Hilo (idthread -> [%ld])\n", (long int) pthread_self());
21     pthread_exit(0);
22 }

```

Hilos principal ( main() )



pthread\_create  
( funcion\_manaja\_hilo() )



```
1  /*
2  Ejemplo Hilo2.c
3  Envío de parametros simple en la creacion de hilos
4  mediante cadena de caracteres
5  */
6  #include <stdio.h>
7  #include <unistd.h>
8  #include <pthread.h>
9
10 void* funcion_maneja_hilo(void *);
11
12
13 int main(){
14     pthread_t pidhilo;
15     pthread_create(&pidhilo, NULL, funcion_maneja_hilo, (void*) "hola");
16     printf("Hilo principal (idthread -> [%ld])\n", (long int) pthread_self());
17     pthread_join(pidhilo, NULL);
18     return 0;
19 }
20
21 void* funcion_maneja_hilo(void *param){
22     printf("Hilo param->%s (idthread -> [%ld])\n", (char *)param, (long int) pthread_self());
23     pthread_exit(0);
24 }
25
```

```
1  /*
2  Ejemplo Hilo2.c
3  Envío de parametros a un hilo mediante el uso
4  de una estructura de datos
5  */
6  #include <stdio.h>
7  #include <unistd.h>
8  #include <pthread.h>
9
10 void* funcion_maneja_hilo(void *);
11 typedef unsigned long int tipo_hilo;
12
13 struct nodo{
14     int val;
15 };
16
17 int main(){
18     pthread_t pidhilo;
19     struct nodo Nodo;
20     Nodo.val = 5;
21     pthread_create(&pidhilo, NULL, funcion_maneja_hilo, (void*) &Nodo);
22     printf("Hilo principal (idthread -> [%lu])\n", pthread_self());
23     pthread_join(pidhilo, NULL);
24     return 0;
25 }
26
27 void* funcion_maneja_hilo(void *param){
28     printf("Hilo param->%d (idthread -> [%lu])\n",((struct nodo *)param)->val, pthread_self());
29     pthread_exit(0);
30 }
31
```

```
1  /*
2  Ejemplo Hilo2.c
3  Error en el Envío de parametros a multiples hilos mediante el uso
4  de una estructura de datos
5  */
6  #include <stdio.h>
7  #include <unistd.h>
8  #include <pthread.h>
9
10 void* funcion_manaja_hilo(void *);
11 typedef unsigned long int tipo_hilo;
12
13 struct nodo{
14     int val;
15 };
16
17 int main(){
18     pthread_t pidhilo[2];
19     struct nodo Nodo;
20     int i;
21
22     for(i=0; i<2; i++){
23         Nodo.val = i;
24         pthread_create(&pidhilo[i], NULL, funcion_manaja_hilo, (void*) &Nodo);
25     }
26     printf("Hilo principal (idthread -> [%lu])\n", pthread_self());
27     for(i=0; i<2; i++){
28         pthread_join(pidhilo[i], NULL);
29     }
30     return 0;
31 }
32
33 void* funcion_manaja_hilo(void *param){
34     printf("Hilo param->%d (idthread -> [%lu])\n",((struct nodo *)param)->val, pthread_self());
35     pthread_exit(0);
36 }
```

```
1  /*
2  Ejemplo Hilo2.c
3  Correcto Envío de parametros a multiples hilos mediante el uso
4  de una estructura de datos
5  */
6  #include <stdio.h>
7  #include <stdlib.h>
8  #include <pthread.h>
9
10 void* funcion_manaja_hilo(void *);
11 typedef unsigned long int tipo_hilo;
12
13 struct nodo{
14     int val;
15 };
16
17 int main(){
18     pthread_t pidhilo[2];
19     struct nodo *Nodo;
20     int i;
21
22     for(i=0; i<2; i++){
23         Nodo = (struct nodo *)malloc(sizeof(struct nodo));
24         Nodo->val = i;
25         pthread_create(&pidhilo[i], NULL, funcion_manaja_hilo, (void*) Nodo);
26     }
27     printf("Hilo principal (idthread -> [%lu])\n", pthread_self());
28
29     //Esperar por la terminacion de los Hilos
30     for(i=0; i<2; i++){
31         pthread_join(pidhilo[i], NULL);
32     }
33     return 0;
34 }
35
36 void* funcion_manaja_hilo(void *param){
37     printf("Hilo param->%d (idthread -> [%lu])\n",((struct nodo *)param)->val, pthread_self());
38     pthread_exit(0);
39 }
```