Hilos

Adaptación de múltiples referencias bibliográficas

Juan Felipe Muñoz Fernández

Definición

- Unidad básica de utilización de CPU. Información propia de cada hilo:
  - ID del hilo (thread)
  - Registro IP
  - Conjunto de registros
  - Pila
- Tradicionalmente: un proceso un hilo de ejecución: `main()`.
- Un proceso puede crear más de un hilo, los hilos comparten
  - Sección de código
  - Sección de datos
  - Archivos abiertos
  - Procesos hijos

Definición

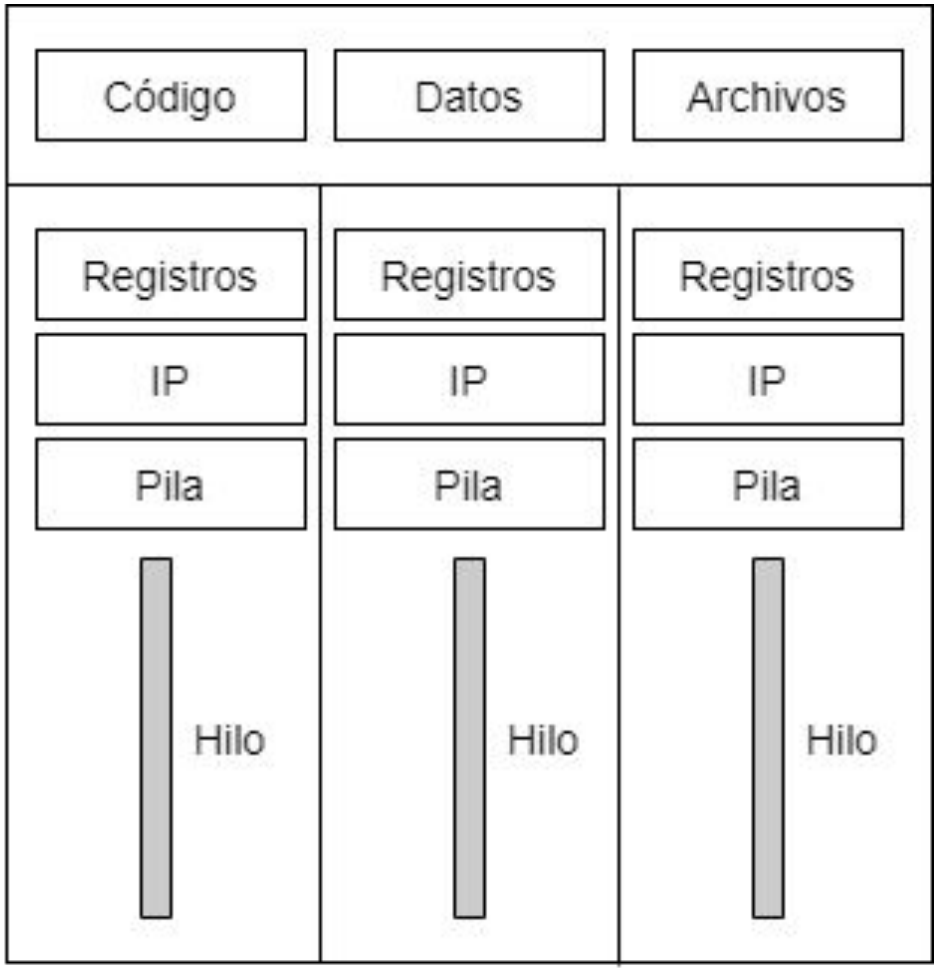
- Los hilos están dentro del mapa de memoria del proceso.
- Cada hilo es un flujo de ejecución.
- Todos los hilos comparten la imagen de memoria.
- Un hilo es una función que se puede ejecutar en paralelo
- Un hilo puede estar ejecutando, listo o bloqueado.

Motivación

- Aplicaciones modernas se implementan como procesos separados con varios hilos de control.
- Procesador de texto multihilo
  - Hilo que muestra documento
  - Hilo que revisa ortografía
  - Hilo que realiza operaciones de autoguardado cada cierto tiempo (en
  - Hilo que procesa las pulsaciones de teclado
- Procesador de texto un solo hilo
  - Todas las funciones anteriores pero frente a la ejecución de una de ellas, las demás deben esperar

Motivación


Código
Datos
Archivos
Registros
IP
Pila
Hilo principal


Código
Datos
Archivos
Registros
IP
Pila
Hilo
Registros
IP
Pila
Hilo
Registros
IP
Pila
Hilo

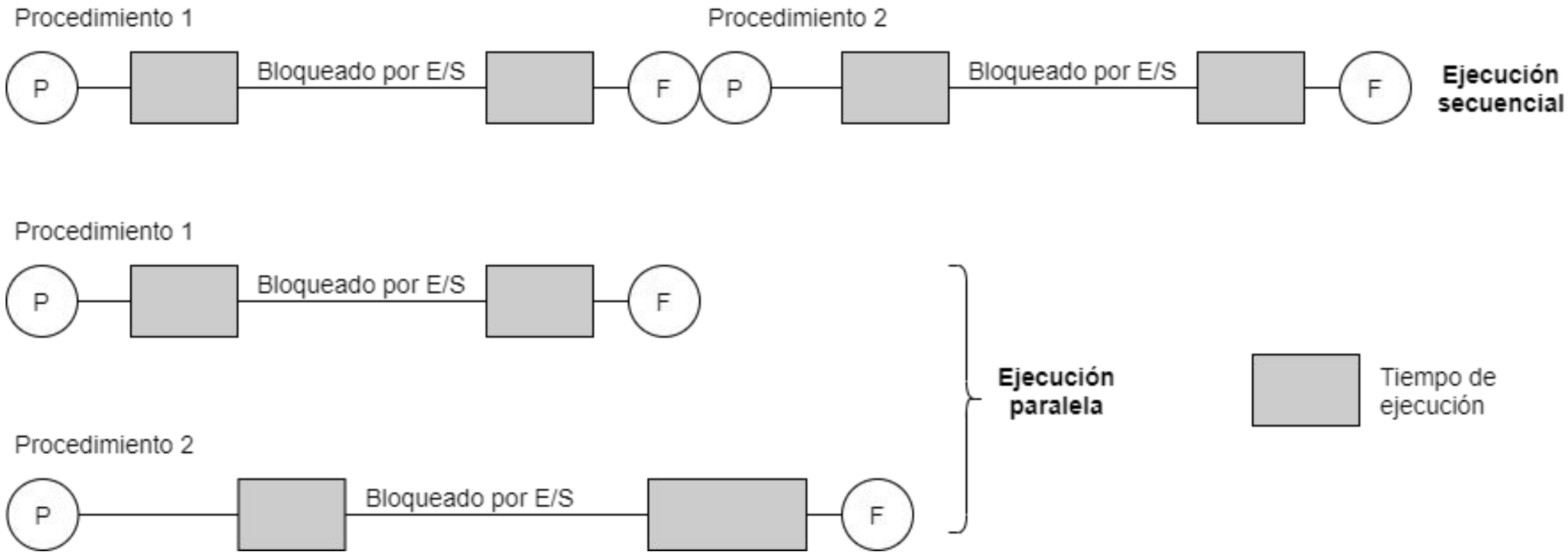
Motivación

- Aprovechar las capacidades de sistemas multinúcleo
- Es más eficiente crear hilos que procesos
  - La creación y terminación de un proceso tiene más *overhead* que la de un hilo.
- Programas intensivos de CPU hay que diseñarlos con capacidades de procesamiento paralelo.
  - Problemas de ordenamiento
  - Árboles
  - Gráficos
  - Problemas de minería de datos
  - Problemas de IA

Estados de los hilos

- Tres estados posibles
  - Ejecutando
  - Listo para ejecutar
  - Bloqueado
- El estado del proceso será la combinación de los estados de sus hilos
  - Si un hilo en ejecución entonces estado del proceso en ejecución.
  - Si no hay hilos en ejecución pero hay un hilo en estado de listo, entonces proceso está en estado de listo.
  - Si todos los hilos están bloqueados, entonces estado del proceso es bloqueado.

Paralelismo



Paralelismo

- La base del paralelismo está en que mientras exista un hilo bloqueado, otro hilo puede estar ejecutándose.
- Concurrencia y paralelismo son dos cosas diferentes
- Hilos
  - Permiten variables compartidas y paralelismo
  - Usan llamadas bloqueantes
- Proceso convencional de un solo hilo
  - No hay paralelismo
  - Usa llamadas al sistema bloqueantes
- Varios procesos convencionales cooperativos
  - Permiten paralelismo
  - No comparten variables
  - Se requieren mecanismos de IPC para compartir información.

Desafíos desde la programación

- Sistemas multinúcleo requieren diseños de software que aprovechen dichas capacidades
- Diseñadores de S.Os deben escribir algoritmos de despacho de procesos e hilos que aprovechen todos los núcleos.
- Cada núcleo se ve como un procesador independiente para el S.O.

Desafíos desde la programación

- Identificar tareas
  - Identificar en un proceso cómo se puede dividir en tareas concurrentes
  - Idealmente tareas independientes
- Balance
  - Tareas que hagan un trabajo igual de igual valor para el proceso general
  - Algunas tareas en paralelo puede que no aporten mucho al proceso en general
- División de datos
  - Datos deben dividirse para cada una de las tareas que se ejecutan en núcleos independientes
- Dependencia de datos
  - Sincronizar ejecución cuando existe dependencia de datos

Desafíos desde la programación

- Pruebas y depuración
  - Tareas paralelas tienen diferentes caminos de ejecución.
  - Depuración y pruebas es mucho más difícil.
- Se requieren nuevos paradigmas de diseño de software
  - No seguir pensando en modelos secuenciales
  - Pensar en modelos de ejecución paralela
- Programación con alto nivel de dificultad
  - Acceso a datos compartidos se haga de forma correcta.
  - Accesos incorrectos a variables: se comparten variables globales.
  - Implementar mecanismos de sincronización de procesos/hilos.

Ejecución sincrónica y asincrónica

- **Ejecución sincrónica**
  - Hilo padre crea hilo(s) y continua su ejecución.
  - Papá e hijo se ejecutan concurrentemente e independientemente.
  - Hilos independientes hay pocos datos compartidos por ambos hilos.
- **Ejecución asincrónica**
  - Hilo padre crea uno o más hilos hijos.
  - Hilo padre espera a que hijos terminen para poder continuar.
  - Hilo terminado se une a hilo padre.
  - Hilo padre continua ejecución cuando todos los hijos terminen
  - Comparten muchos datos entre todos lo hilos
  - Hilo padre combina resultados de hilos hijos.

Modelos de multihilos

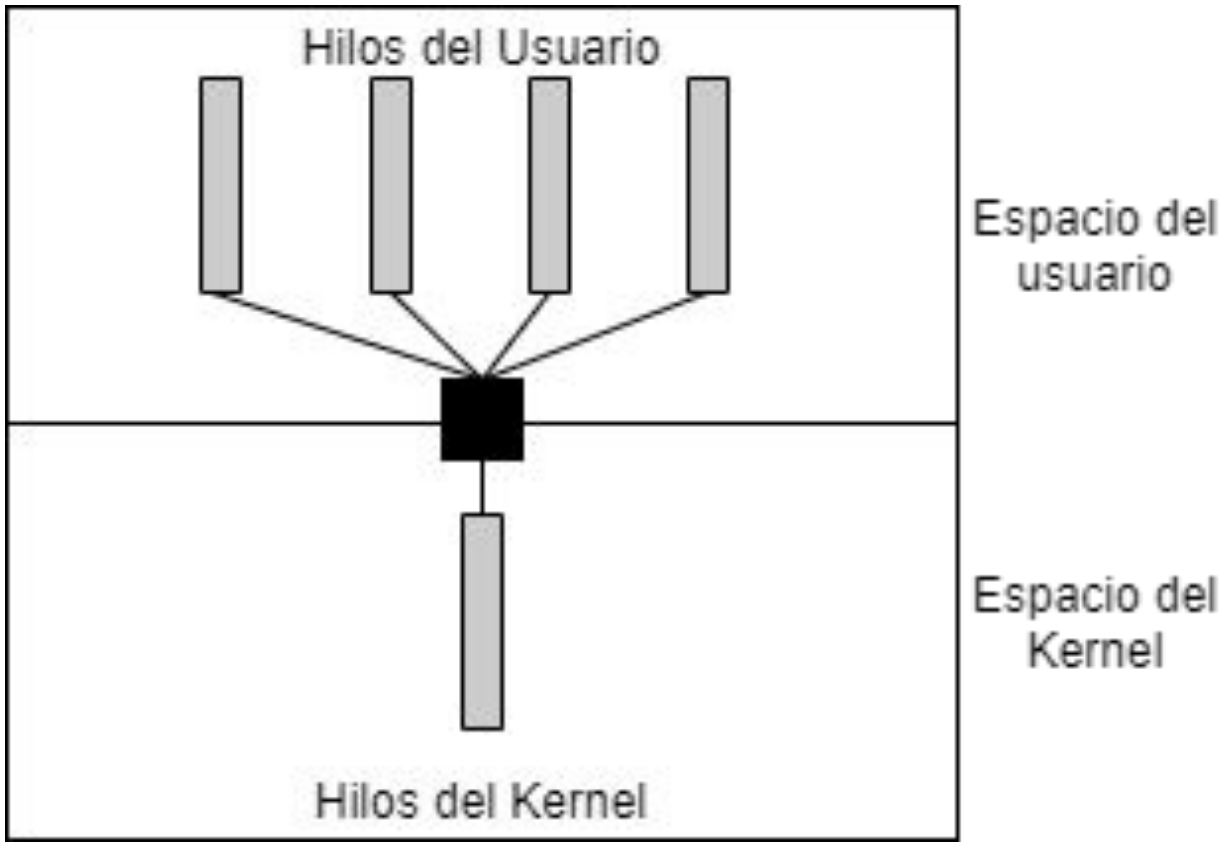
- Se deben suministrar mecanismos de soporte de hilos
  - A nivel del espacio del usuario
  - A nivel del núcleo (kernel) del sistema operativo
- Hilos a nivel de usuario
  - Se gestionen en el espacio del usuario, no requieren soporte del kernel.
- Hilos a nivel del kernel del sistema operativo
  - Se soportan y se administran directamente por el kernel del sistema operativo
- Debe existir una relación entre hilos en el espacio de usuario e hilos en el nivel del kernel.

Multithreading | Hyperthreading

- Capacidad de procesadores modernos de permitir paralelismo de instrucciones.
- Procesadores modernos tienen más de un núcleo
  - Cada núcleo es dividido en dos o más procesadores lógicos
  - Cada procesador lógico soporta un hilo de ejecución
- P. Ej.: Intel Core i5-6200u
  - Cantidad de núcleos: 2
  - Cantidad de subprocesos (hilos): 2
  - Sistema operativo Windows considera 4 procesadores lógicos

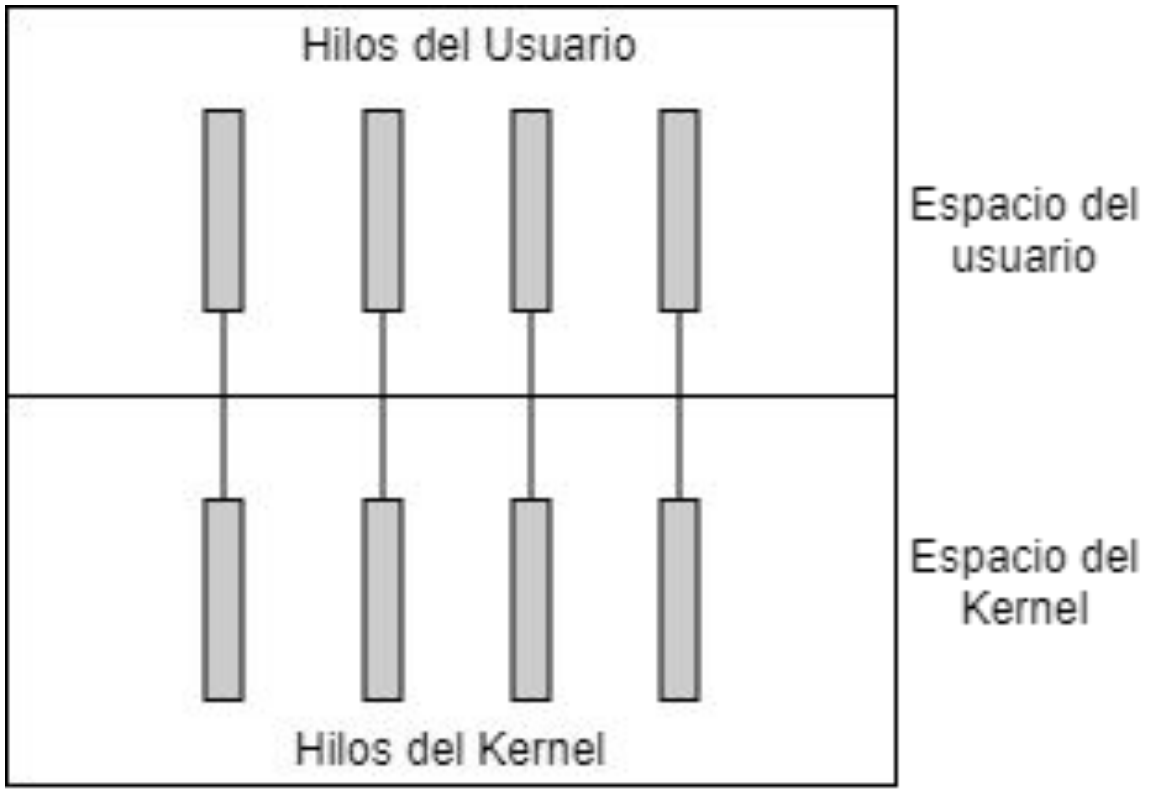
Modelos de multihilos: muchos a uno

- Hilos los gestionan librería en espacio del usuario
- Un hilo hace una *system call* bloqueante
  - Se bloquea todo el proceso
- Un solo hilo puede acceder al Kernel
  - No hay paralelismo en sistemas multi núcleo.
- Muy pocos sistemas ofrecen este modelo



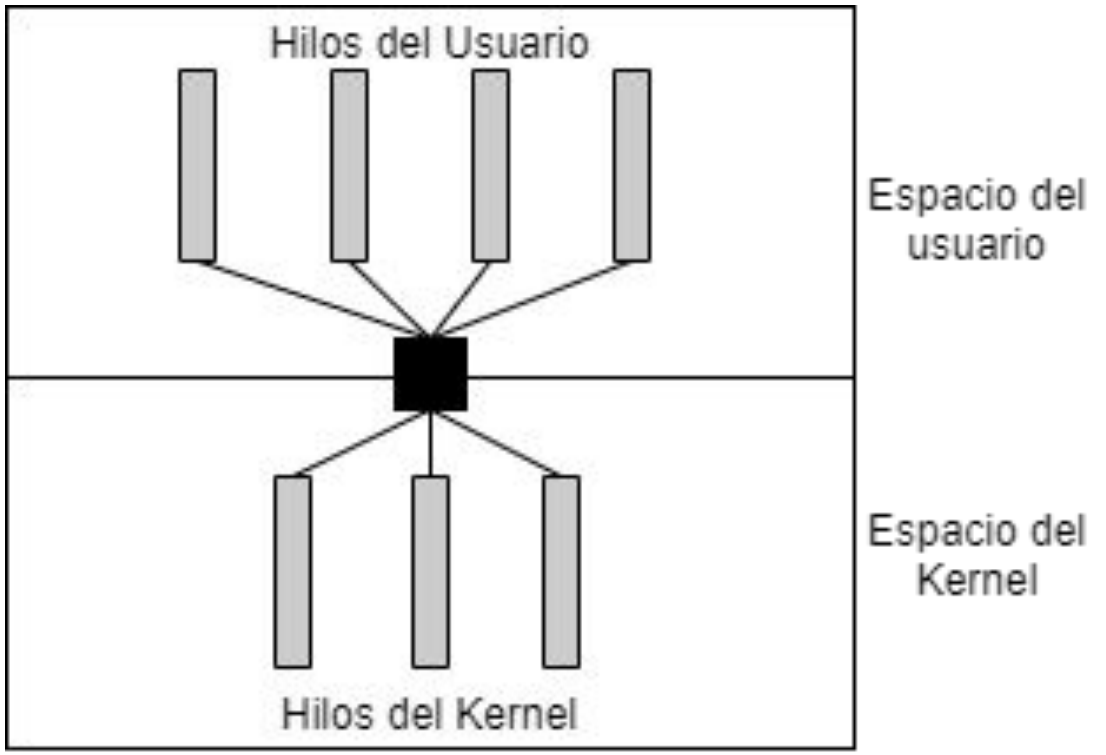
Modelos de multihilos: uno a uno

- Asociación entre un hilo del usuario contra un hilo del kernel.
- Un hilo bloqueado NO bloquea al resto.
- Se permite paralelismo en sistemas multinúcleo.
- Castiga desempeño:
  - Por cada hilo de usuario se crea un hilo en el kernel.
- Linux y Windows implementan este modelo.



Modelos de multihilos: muchos a muchos

- Se multiplexan los hilos del espacio de usuario con un número menor o igual de hilos en Kernel.
- Permite paralelismo
- Número de hilos en Kernel deben ser específicos para situaciones particulares.
  - Más hilos de Kernel en máquina con 8 núcleos que en máquina de 4.
- Difícil de implementar



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

void *mihilo(void *arg) {
    printf("%s\n", (char *) arg);
    return NULL;
}

int main(int argc, char *argv[]) {
    pthread_t p1, p2;
    printf("main(): Inicia\n");
    pthread_create(&p1, NULL, mihilo, "Hilo A");
    pthread_create(&p2, NULL, mihilo, "Hilo B");

    /* Esperar a que hilos terminen */
    pthread_join(p1, NULL);
    pthread_join(p2, NULL);
    printf("main(): Termina\n");
    return 0;
}
```

```
# ./hilos2
main(): Inicia
Hilo A
Hilo B
main(): Termina
```

main()	Hilo 1	Hilo 2
Inicia ejecución Imprime **`main(): Inicia`** Crea Hilo 1 Crea Hilo 2 Espera por Hilo 1		
	Se ejecuta Imprime **`Hilo A`** Retorna	
Espera por Hilo 2		
		Se ejecuta Imprime **`Hilo B`** Retorna
Imprime **`main(): Termina`**		

main()	Hilo 1	Hilo 2
Inicia ejecución Imprime **`main(): Inicia`** Crea Hilo 1		
	Se ejecuta Imprime **`Hilo A`** Retorna	
Crea Hilo 2		
		Se ejecuta Imprime **`Hilo B`** Retorna
Espera por Hilo1 *Retorna inmediatamente (ya terminó)*		
Espera por Hilo 2 *Retorna inmediatamente (ya terminó)*		
Imprime **`main(): Termina`**		

main()	Hilo 1	Hilo 2
Inicia ejecución Imprime **`main(): Inicia`** Crea Hilo 1 Crea Hilo 2		
		Se ejecuta Imprime **`Hilo B`** Retorna
Espera por Hilo 1		
	Se ejecuta Imprime **`Hilo A`** Retorna	
Espera por Hilo2 *Retorna inmediatamente (ya terminó)*		
Imprime **`main(): Termina`**		

Ejecución de hilos

- **No se puede asumir** que el hilo que se crea primero, es el primero en ejecutarse.
- La creación de hilo es una llamada a una función.
  - El sistema operativo crea un hilo de ejecución cuando se llama a la función.
  - El hilo se ejecuta de manera independiente del programa que llama la función.
  - Lo que se ejecuta después del llamado depende del planificador del S.O
    - Ya que los hilos también tiene diferentes estados pasan por el planificador del S.O.
  - Es difícil saber que se ejecutará en un momento dado.

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

void * funcion(void *arg) {
    printf("Hilo %d \n", (int) pthread_self());
    pthread_exit(0);
}

int main(int argc, char *argv[]) {
    pthread_t h1, h2;

    pthread_create(&h1, NULL, funcion, NULL);
    pthread_create(&h2, NULL, funcion, NULL);

    printf("main() sigue su ejecución\n");
    printf("main() sigue su ejecución\n");
    printf("main() sigue su ejecución\n");

       /* Se espera a que terminen */
    pthread_join(h1, NULL);
    pthread_join(h2, NULL);

    printf("main() termina\n");
    return 0;
}
```

Hilado sincrónico:

- Hilo padre crea hilos hijos y sigue su ejecución.
- Hilos se ejecutan con independencia: padre e hijos.
- En algún punto hilo padre espera a hilos hijos: **`pthread_join()`**
- Se usa usualmente cuando hay muchos datos compartidos entre los hilos: hilo padre consolida la información de los hilos hijos cuando llama a **`pthread_join()`.**

```
# gcc -pthread -o hilos_sync hilos_sync.c
# ./hilos_sync
main() sigue su ejecución
main() sigue su ejecución
main() sigue su ejecución
Hilo 225703680
Hilo 234096384
main() termina
```

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

void * funcion(void *arg) {
    printf("Hilo %d \n", (int) pthread_self());
    pthread_exit(0);
}

int main(int argc, char *argv[]) {
    int j;
    pthread_t hilos[10];
    pthread_attr_t attr;

    pthread_attr_init(&attr);
    pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_DETACHED);

    for (j = 0; j < 10; j++) {
        pthread_create(&hilos[j], &attr, funcion, NULL);
    }

    /* Se espera 5 segundos a que terminen hilos */
    sleep(5);

    printf("main() termina\n");
    return 0;
}
```

```
# gcc -pthread -o hilos_async hilos_async.c
# ./hilos_async
Hilo 208307968
Hilo 199915264
Hilo 241878784
Hilo 233486080
Hilo 225093376
Hilo 191522560
Hilo 183129856
Hilo 174737152
Hilo 216700672
Hilo 250271488
main() termina
```

Hilado asincrónico:

- Hilo padre crea hilos hijos y sigue su ejecución.
- Hilo padre e hilo(s) hijo(s) se ejecutan concurrentemente.
- No se comparten muchos datos entre los hilos.

Referencias

- Arpaci-Dusseau, R. H., & Arpaci-Dusseau, A. C. (2018). Concurrency: An Introduction. In *Operating Systems. Three Easy Pieces*. Arpaci-Dusseau Books.
- Carretero Pérez, J., García Carballeira, F., De Miguel Anasagasti, P., & Pérez Costoya, F. (2001). Procesos ligeros. In *Sistemas operativos. Una Visión Aplicada* (pp. 98–101). McGraw Hill.
- Silberschatz, A., Galvin B., P., & Gagne, G. (2018). Threads & Concurrency. In *Operating Systems Concepts* (10th ed., pp. 159–196). John Wiley & Sons, Inc.
- Tanenbaum, A. S. (2009). Hilos. In *Sistemas Operativos Modernos* (3rd ed., pp. 95–114). Pearson Educación S.A.