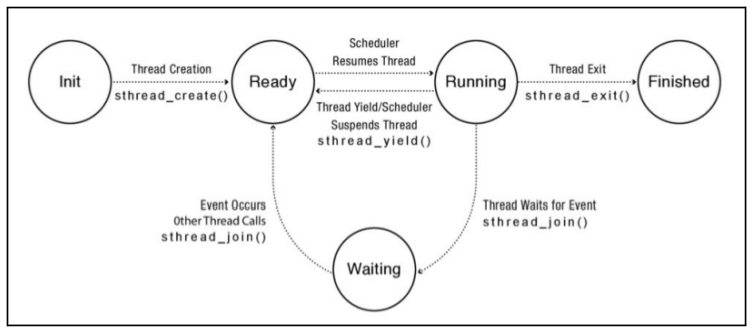
## ¿Qué es un thread, cual es su estructura y qué estados posee?

Un thread es una secuencia de ejecución atómica que representa una tarea planificable de ejecución. Su estructura es la de un proceso, contenido por: metadata, código, región de datos, stack y registros de CPU. Los estados del mismo se almacenan en la TCB (que por cada thread tiene una entrada que guarda un puntero al stack del mismo) y pueden ser:

* Init -> el thread es creado, y se genera toda la información necesaria en las estructuras del Sistema Operativo (se crea una entrada en la TCB)
* Ready -> el thread está en la cola de planificación, listo para ser ejecutado
* Running -> los registros de la TCB son cargados en el procesador y comienza la ejecución del thread
* Waiting -> el thread está esperando para ser planificado
* Finished -> el thread finalizó su ejecución y nunca más será planificado, de modo que el registro de la TCB se mueve a la lista de finished threads

****

## [Describa qué es un thread](#_cmshp183yse) y use su API para crear un programa que use 5 threads para incrementar una variable compartida por todos en 7 unidades/thread hasta llegar a 1000.

## Defina y dé ejemplos: race-condition, heisenbug, dead-lock, interleave.

* Race-Condition: Ocurre cuando la salida de un programa depende de cómo se intercalan las operaciones de los threads que ocurren dentro de un proceso.
* Heisenbug: Error no determinístico que depende de condiciones como el intercalado del scheduler.
* Dead-Lock: Ocurre cuando un thread lockea un recurso y no lo libera haciendo que el resto de los threads que necesitan utilizar ese mismo recurso queden esperando infinitamente.
* Interleave: Concepto que representa el intercalado de la ejecución de los threads en el CPU y depende del scheduler.

## Defina la abstracción que mejor maneja la **granularidad más fina de cómputo** (thread). Cómo se denomina, sus características principales. Y en un ejemplo que muestre cómo funciona y cuáles son las precauciones que un programador debe tener en cuenta a la hora de utilizar dicha abstracción. Sin el ejemplo es inválido el ejercicio.

Un thread es una secuencia de ejecución atómica que representa una tarea planificable de ejecución y está compuesto por: metadata, stack, registros de CPU, código y datos.

1. **Dado el siguiente tipo de dato. Identificar la sección crítica:**

**struct QNode {**

**int key;**

**struct QNode\* next;**

**};**

**struct Queue {**

**struct QNode \*front, \*rear;**

**};**

**void deQueue(struct Queue\* q){**

**if (q->front == NULL)**

**return;**

**struct QNode\* temp = q->front;**

**q->front = q->front->next;**

**if (q->front == NULL)**

**q->rear = NULL;**

**free(temp);**

**}**

1. **Muestre una implementación de un tipo de dato en el cual una variable que es compartida por varios threads es modificada y accedida de forma sincronizada.**
2. **Dado la siguiente estructura de datos, implemente la función insertar y la función crear, de forma tal que sea thread safe:**

**int List\_Insert(list\_t \*L, int key) {**

**node\_t \*new = malloc(sizeof(node\_t));**

**if (new == NULL) {**

**perror("malloc");**

**return -1; // fail**

**}**

**new->key = key;**

**new->next = L->head;**

**L->head= new;**

**return 0; // success**

**}**

**// nodo**

**typedef struct \_\_node\_t {**

**int key;**

**struct \_\_node\_t \*next;**

**} node\_t;**

**// lista**

**typedef struct \_\_list\_t {**

**node\_t \*head;**

**} list\_t;**

**Explique detalladamente que es un thread. Cuáles son sus componentes. Qué metadata se debe almacenar. Cuál es su relación con el paralelismo.**