Librería pthread (POSIX)

- La librería de pthread es una librería incluida en el estándar POSIX que nos permite construir programas concurrentes.
- Ya que pthread es una librería POSIX, los programas concurrentes en donde se utiliza se pueden portar a cualquier sistema operativo que soporte el estándar POSIX.
- Para usarla se debe incluir el archivo de cabecera **`pthread.h`** y enlazar la librería añadiendo a la llamada a **gcc** el parámetro **`-lpthread`**.
- Todos los procesos son creados con un thread inicial que crea a los demás.
- Los threads comparten la memoria y los demás recursos del proceso entre ellos.
- Es una buena práctica que el hilo que creó a otro thread, espere a que se termine
- En las siguientes sesiones estudiaremos las principales funciones de la librería.

Elizabeth Pérez Cortés

Creación de un hilo: pthread_create

```
#include <pthread.h>

int pthread_create(pthread_t *restrict thread,
    const pthread_attr_t *restrict attr,
    void *(*start_routine)(void*), void *restrict arg);
```

`pthread_create` Crea un hilo nuevo dentro del proceso en donde se le invoca con los atributos `attr`, si `attr=NULL`, se crea con los atributos por defecto. Los atributos no se pueden modificar después. Si el llamado es exitoso, el parámetro `thread` contendrá el ID del thread creado, la invocación regresa 0 y el thread ejecutará *`start_routine`* con *`arg`* como único argumento.

Si la rutina invocada ejecuta `return`, el efecto será el mismo que el de una invocación a `pthread_exit` y se tomará el valor regresado como el estatus de salida. El hilo padre (que ejecuta el main) se comporta diferente, si ejecuta la instrucción `return`, entonces se tomará como si invocara a `exit` y el valor regresado se tomará como el estatus de salida.

Elizabeth Pérez Cortés

Creación de un hilo: pthread_exit

```
#include <pthread.h>

void pthread_exit(void *value_ptr);
```

La función **pthread_exit** termina el hilo que la invoca y el valor **value_ptr** es puesto a disposición de cualquier invocación exitosa a **pthread_join** con el hilo en cuestión.

Se hace un llamado implícito a esta función cuando un hilo (diferente del hilo padre) regresa de la rutina que se usó para crearlo. El valor de regreso servirá como el estatus del hilo.

Después de que un hilo termina, el acceso a sus variables locales (automáticas) es indefinido.

El proceso debe terminar con un estatus 0 después de que el último hilo ha terminado. El comportamiento debe ser como si la implementación hubiese invocado **exit** con 0 al tiempo de terminación del hilo.

Elizabeth Pérez Cortés

Creación de un hilo: pthread_join

```
#include <pthread.h>

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

La función **pthread_join** suspende la ejecución del hilo que la invoca hasta que el hilo objetivo (indicado en el primer parámetro) termina. A menos que el hilo objetivo ya haya terminado. Si se invoca con **value_ptr** diferente de NULL, el valor enviado por el hilo objetivo en **pthread_exit** se podrá a disposición del hilo que invoca en la localidad referenciada por ese parámetro. Cuando el llamado regresa de manera exitosa, el hilo objetivo ha terminado.

Solo debe ser invocado por un hilo, de otra manera el resultado es indefinido.

Elizabeth Pérez Cortés

Ejemplo:

```
#include <pthread.h>
...
#define HILOS 4
#define ITERACIONES 3
void Escribe(void *);

int main()
{
    int i, ids[HILOS];
    pthread_t hilos[HILOS];

    for(i = 0; i < HILOS; i++){
         ids[i] = i;
         pthread_create(&hilos[i],NULL,
            (void *)&Escribe,
            (void*)(intptr_t)ids[i]);
    }

    for(i = 0;i < HILOS; i++)
         pthread_join(hilos[i], NULL);

    printf("Hilo padre: todos han terminado\n");
}
```

```
void Escribe(void *ptr){

  int j, numHilo = (intptr_t) ptr;

  for(j=0; j<ITERACIONES; j++){

    switch(j){
     case 0: printf("%d: Hola!!!\n",numHilo);
           break;
     case 1: printf("%d: Estoy feliz\n",numHilo);
           break;
     default: printf("%d: Nos vemos
pronto\n",numHilo);
           break;
    }

  }
  pthread_exit(0);
}
```