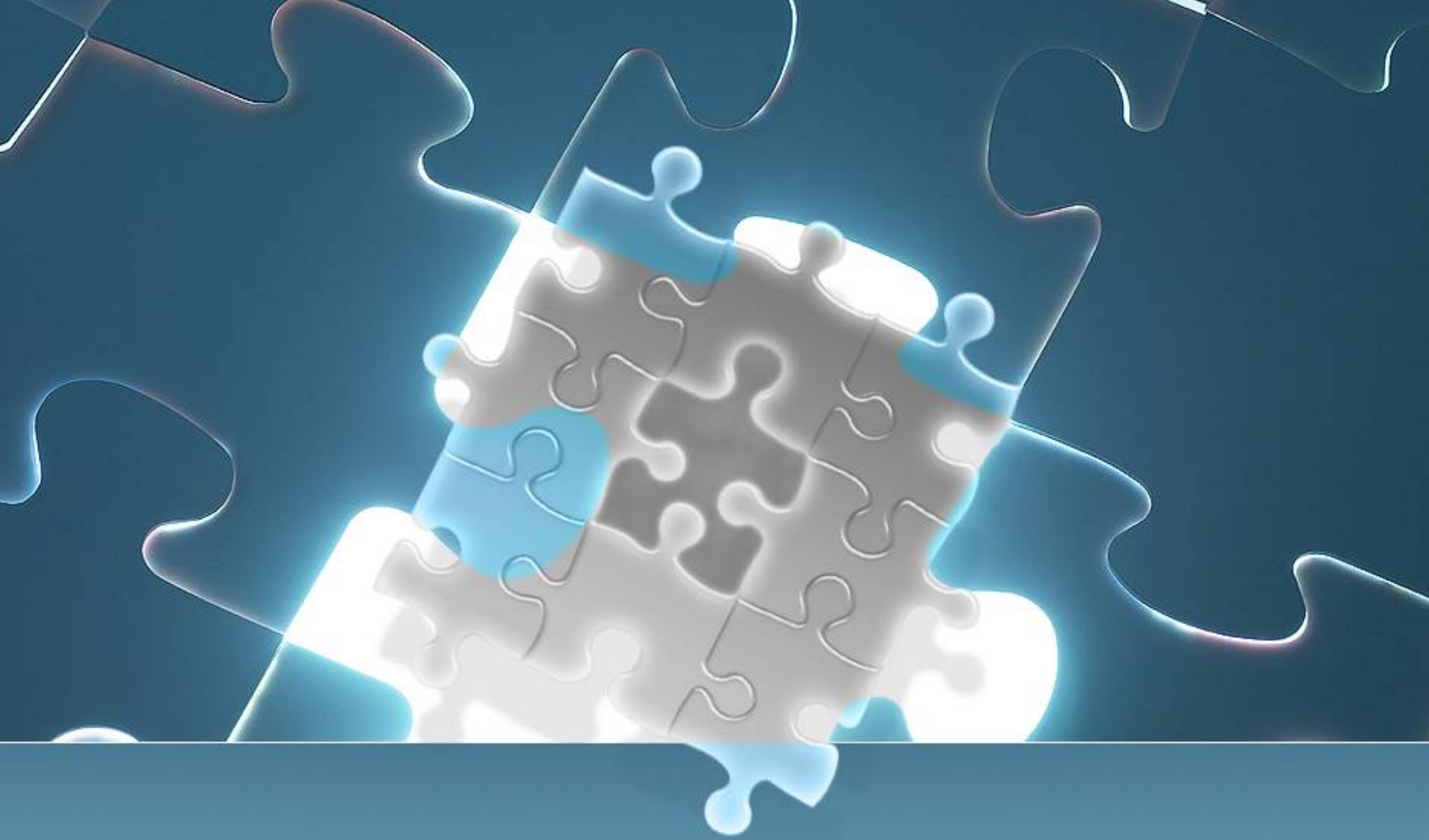

Programando Hilos

Grupo 2

Creación de Hilos

Para crear un hilo es necesario declarar la variable de tipo thread:

pthread_t este tipo de variable esta definida en **pthread.h**

ptread_t nombre_del_hilo

Creación de Hilos

- Ahora debe crearse el hilo mediante la función:
- **pthread_create()**
- La cual devolverá un valor de “0”, si se creo satisfactoriamente el hilo y “1” en caso contrario

Argumentos de pthread_create()

- **pthread_create(arg1, arg2, arg3, arg4)**

- **arg1**: Puntero a la variable de tipo thread creada anteriormente
- **arg2**: Usado para definir atributos al hilo, regularmente se usa NULL
- **arg3**: Nombre de la función que el hilo ejecutara cuando comience
- **arg4**: Paso de argumentos a la función que ejecutará el hilo

Generales

- Al momento de crear nuestro primer hilo en realidad tendremos dos hilos.
- Nuestro programa principal también se considera un hilo.
- ¿Que hace el hilo principal al crear nuestro hilo?

Generales

- Se mantiene y ejecuta secuencialmente la línea de programa que sigue.
- Para que el hilo principal espere la finalización del hilo que acaba de crear utilizaremos la función:

Espera de los hilos

- **pthread_join(arg1,arg2)**

- **arg1**: Nombre del hilo a esperar
- **arg2**: Argumentos que se recibirán del hilo

Generales

- **No manejar los terminos: hilo hijo e hilo padre.**
- En los hilos POSIX esta relación jerarquica no existe. Mientras que un hilo principal puede crear otro hilo y este hilo puede a su vez crear otro hilo, POSIX ve a todos los hilos como a un simple conjunto de elementos idénticos.

Generales

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

void *Hilo_Funcion(void *arg) {
  int i;
  for ( i=0; i<20; i++ ) {
  printf("%d: El hilo dice Hola!\n",i);
    sleep(1);
  }
  return NULL;
}

int main(void) {

  pthread_t mi_hilo;

  if ( pthread_create( &mi_hilo, NULL, Hilo_Funcion, NULL) ) {
    printf("Error al crear el hilo");
    abort();
  }

  if ( pthread_join ( mi_hilo, NULL ) ) {
    printf("Error al esperar el Hilo");
    abort();
  }
  exit(0);
}
```

MUTEX

- Los mutex permiten proteger secciones de código críticas, normalmente se usan para proteger estructuras de datos. Con esto se asegura que un solo hilo a la vez puede acceder a una cierta estructura de datos bloqueándola y desbloqueándola.

MUTEX

- Esta actividad de proteger la sección crítica de un código la conocemos como mutua exclusión, es decir dos hilos no pueden mantener la misma estructura de datos bloqueada al mismo tiempo.

Inicialización de un MUTEX

- Método de inicialización estático:
  - Necesitamos crear una variable de tipo: pthread_mutex_t y asignarle la constante:
  - PTHREAD_MUTEX_INITIALIZER

pthread_mutex_t mimutex=PTHREAD_MUTEX_INITIALIZER

Llamada de bloqueo

- **pthread_mutex_lock()** acepta un único puntero para bloquear un mutex. Si el mutex ya se encuentra bloqueado la llamada se duerme, cuando la función retorna la llamada despertará y a partir de ese momento mantendrá el bloqueo

Llamada de desbloqueo

- **pthread_mutex_unlock()** desbloquea un mutex que el hilo había bloqueado. Debe desbloquearse un mutex bloqueado en cuanto sea posible y seguro (para incrementar el rendimiento).