1. Resume de forma **individual** las funciones que fueron utilizadas en C para crear, comunicar y manipular procesos e hilos en C.

fork() Se utiliza para crear un proceso hijo el cual se ejecuta de manera concurrente con el proceso que lo mando a yamar también conocido como el proceso padre.

sleep() La función sleep espera al hilo principal por un periodo de tiempo esceficio (ingresado). Pondra a dormir el hilo asignado por el tiempo del ejecutable actual.

perror() Imprime un mensaje de error ingresado por el usuario.

exit() Termina la llamada del proceso.

getpid() Regresa el ID del proceso hijo.

getppid() Regresa el ID del proceso padre.

getuid() Regresa el ID del usuario en la terminal.

pthread_t inicializa un hílo.

pthread_create(pthread_t thread, const pthread_attr_t attr, void (start_routine) (void), voidarg) crea un nuevo hilo asignado a **thread** con los atributos **attr**.

pthread_join(pthread_t thread, void retval)la función espera a que el hilo especificado con **thread** termine.

pthread_exit(void retval) termina la llamada al hílo.

pipe() Es una conexión entre 2 procesadores, tal que la salida estandar de uno de los procesadores es la entrada estandar del otro procesador. En UNIX los pipes son utiles para la comunicación entre procesos relacionados (comunicación inter-procesador).

2. Investiga de forma **individual** los siguientes conceptos:

a) *Global Interpreter Lock (GIL)* en el contexto del multiprocesamiento en python. Resume los conceptos más importantes.

Un *Global interpreter Lock* es un mecanismo usado por interpretadores para sincornizar la ejecución de hilos para que solo un hilo nativo sea ejecutado a la vez. Esto permite una ejecución más rapida de hilos y es facil de implementar.
en python, un *Global Interpreter Lock* es un *mutex* que protege el acceso a los objetos de python preveniendo que multiples hílos se ejecuten todos al mismo tiempo. Este es necesario porque la administración de memoria de Python no es inicialmente segura de multihilos.

b) *Ley de Amdahl:* Concepto, terminología relacionada, fórmula de la ley.

La ley de Amdahl es una formula la cual da el incremento de velocidad teorico en la latencia de una tarea a una carga de trabajo que puede ser esperada de un sistema cuando sus recursos son mejorados.
Se utiliza en la computación paralela para predecir el incremento en la velocidad teoríca cuando se utilizan multiples procesadores. La formula se define como:

$$S_{latencia} = \frac{1}{(1-p)+\frac{p}{s}}$$

donde

- $S_{latencia}$ es el incremento teorico de la velocidad de ejecucion de una una tarea completa.
- s es el incremento de velocidad de la parte de la tarea que se beneficia de la mejor de los recursos del sistema.
- p es la proporcion del tiempo de ejecucion de la parte que se beneficia de la mejora de los recursos del sistema.

c) *multiprocessing:* Resume la descripción de este módulo de python, enumera y describe los métodos o funciones más importantes del módulo, elige al menos 5 métodos y enlista los argumentos que reciben.

multiprocessing es un paquete que soporta la creación de procesos usando una API similar a la que se crea para el modulo de threading. El paquete multiprocessing ofrece concurrencia local y remota, evadiendo de manera efectiva el GIL mediante la utilización de subprocesos en vez de hilos. Debido a esto, el modulo de multiprocessing permite que el programador manipule multiples procesadores en una maquina. Funciona tanto en Unix como en Windows.
El modulo de multiprocessing también introduce APIs las cuales no tienen analogos in el modulo threading, lo cual permite la implementación paralela de la ejecución de una función a través de multples valores de entrada, distrribuyendo las entradas a través de procesos.
Sus principales metodos son:

run() Método que representa la actividad del proceso.

start() Inicia la acivación del proceso.

join() Método que une un proceso con otro cuando este es terminado en su tiempo indicado.

name Es el nombre del proceso.

pid() Método que devuelve el ID del proceso.

3. Investiga de forma **individual** la manera de crear procesos como un objeto que hereda la clase `multiprocessing.process` y muestra un ejemplo.

El multiprocesamiento funciona creando un objeto del `process` y luego llamando a su método `start()` como se muestra a continuación:

```
1  from multiprocessing import Process
2
3
4  def greeting():
5      print 'hello world'
6
7  if __name__ == '__main__':
8      p = Process(target=greeting)
9      p.start()
10     p.join()
```

El multiprocesamiento admite dos tipos de canales de comunicación entre procesos: Tubos (pipes) que se utilizan principalmente para comunicación entre procesos, y colas(queue) se utilizan para pasar datos entre procesos:

```
1  #ejemplo cola (queue)
2  def is_even(numbers, q):
```

```
    for n in numbers:
        if n % 2 == 0:
            q.put(n)

if __name__ == "__main__":

    q = multiprocessing.Queue()
    p = multiprocessing.Process(target=is_even, args=(range(20), q))

    p.start()
    p.join()

    while q:
        print(q.get())

#ejemplo tubo (pipe)
def f(conn):
    conn.send(['hello world'])
    conn.close()

if __name__ == '__main__':
    parent_conn, child_conn = Pipe()
    p = Process(target=f, args=(child_conn,))
    p.start()
    print parent_conn.recv()
    p.join()
```

También podemos asegurarnos de que solo un proceso de ejecute en un momento dado con el método `Lock()` o aplicar el método `release` para terminar el Lock.

4. Referencias


```
https://en.wikipedia.org/wiki/Global_interpreter_lock

https\:\/\/wiki.python.org\/moin\/GlobalInterpreterLock

https://docs.python.org/3.4/library/multiprocessing.html?highlight=process#module-
    multiprocessing.sharedctypes

https://www.geeksforgeeks.org/fork-system-call/

https://www.poftut.com/what-is-sleep-function-and-how-to-use-it-in-c-program/

https://man7.org/linux/man-pages/man3/pthread_create.3.html

https://www.geeksforgeeks.org/pipe-system-call/
```