Universidad Nacional Autónoma de México

IMASS

COMPUTACIÓN CONCURRENTE

EXAMEN COLEGIADO

Alumno:
Peralta Rionda Gabriel Zadquiel

14 de septiembre del 2022

14 de septiembre de 2022

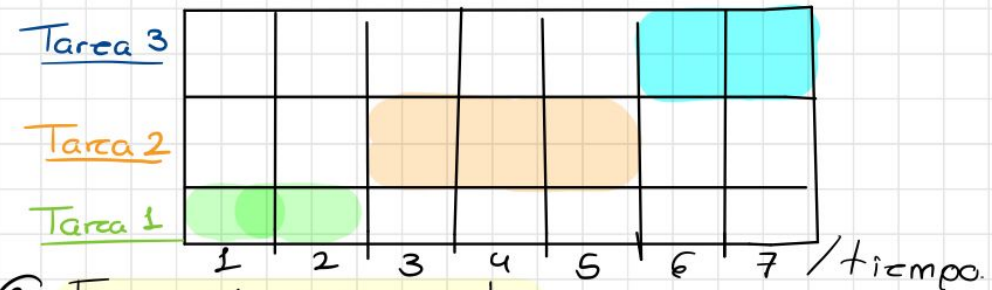
1. Elabora con tus palabras una descripción lo más completa posible del concepto de concurrencia.

   Respuesta: Definimos concurrencia en el ambito computacional como la capacidad del sistema operatico de ejecutar multiples tareas (asi como algorimos o progreamas) que se van ejecutando en cualquier instante de manera independiente creando una ilusión de ejecución en paralelo. Es decir que el sistema operativo va administrando las tareas de tal forma que ejecuta una tarea, pausa esa tarea y ejecuta otra distinta, esto lo hace con algoritmos que tiene el sistema operativo. Este tipo de ejecución de un programa ayuda a que tengamos programas más eficientes.

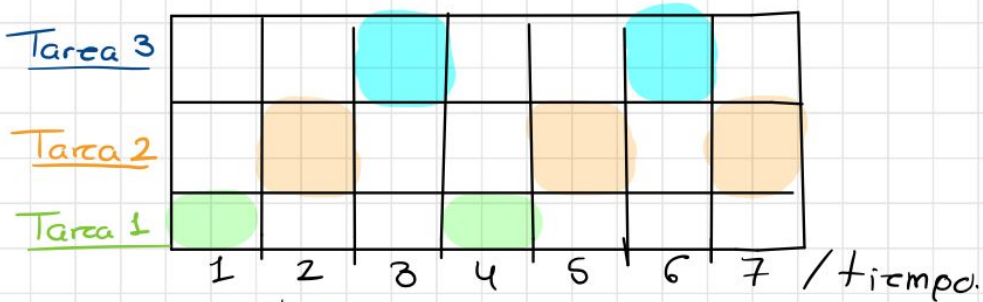
2. Utiliza diagramas de tiempos y tareas (diagram de Gantt) donde muestres los tres tipos de ejecución: secuencial, concurrente y paralela.

Diagrama de Gantt.

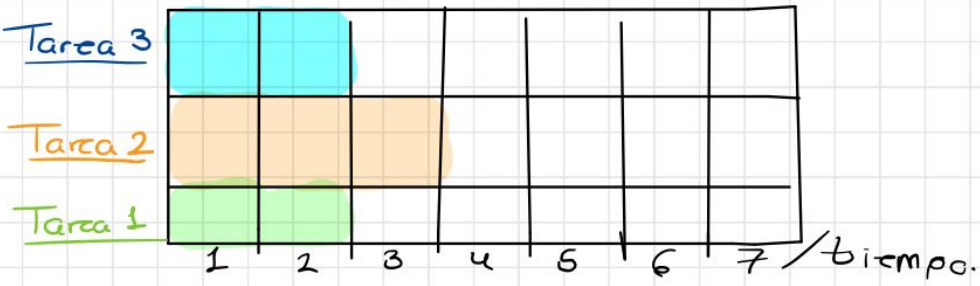
1. Ejecución secuencial.
   → El sistema operativo ejecuta multiples tareas una detras de otra con un orden previamente establecido.



2. Ejecución concurrente
   → El sistema operativo ejecuta multiples tareas donde estas se ejecutan en cualquier instante (es decir de manera simultanea pero un orden indeterminado).



3. Ejecución en paralelo
   → El sistema operativo ejecuta multiples tareas al mismo tiempo y de manera independiente.



3. Menciona lo que entiendes por llamadas al sistema, el estándar POSIX y las funciones básicas para implementar concurrencia en C-Linux.

   -Las llamadas al sistema es un método en que el programador se comunica con el sistema operativo. En los sistemas operativos modernos, esto es necesario cuando un proceso de usuario necesita transmitir a o leer información del hardware, de otros procesos o del propio núcleo.

   -El estándar POSIX es un conjunto de reglas o normas que nos indican como definir la interfaz de nuestro sistema operativo y el entorno, incluyendo la terminal.

   Funciones para implementar concurrencia:

   a) UID es la función que nos regresa el identificador de usuario.

   b) PID es la función que nos regresa el indetificador del proceso actual.

   c) PPID es la función que nos regresa el identificador del proceso padre del proceso actual.

   d) fork() es una función del lenguaje C que nos permite crear un proceso y que El proceso hijo y el proceso padre se ejecutan en espacios de memoria separados. En el momento de 'fork()' ambos espacios de memoria tienen el mismo contenido. En caso de éxito, el PID del proceso hijo se devuelve en el padre y se devuelve 0 en el hijo. En caso de error, se devuelve -1 en el padre, no se crea ningún proceso hijo.

   e) pipe() Es una función que nos permite crear una tuberia un canal de datos unidireccional que se puede utilizar para la comunicación entre procesos.

   f) exit(estado). La función de biblioteca exit finaliza un proceso, haciendo que todos los recursos (memoria, descriptores de archivos abiertos, etc.) utilizados por el proceso disponible para posterior reasignación por parte del kernel

   g) wait(estado). La llamada al sistema wait(estado) tiene dos propósitos. Primero, si un hijo de este proceso aún no ha terminado llamando a exit(), entonces wait() suspende la ejecución del proceso hasta que uno de sus hijos haya terminado.

4. Implementa en Python tres versiones de un programa que calcula el factorial de un número entero proporcionado desde el teclado.

   a) Implementa la versión secuencial.

```python
#Definimos la funcipon que calcula el factorial de un número de forma secuencial.
def factorial(x):
  multi = 1
  for i in range(1, x+1):
    multi *= i
  print(f'{multi}\n')
#Probamos la ejecución secuencial de la función factorial.
print('Introduce el número al cual quieres calcular su factorial:')
n = int(input())
print('-----')
print('Resultado:')
factorial(n)
print('------')
```

```
Introduce el número al cual quieres calcular su factorial:
10
-----
Resultado:
3628800

------
```

b) Implementa la versión concurrente en la cual se dividen la tarea dos procesos.

```python
#Importamos la libreria de python para realizar procesos.
import multiprocessing as mp
#Definimos las funciones que realizaran cada proceso.
def fac_concurrente1(fact,x):
  for i in range(1,x//2):
    fact.value *= i
def fac_concurrente2(fact,x):
  for i in range(x//2,x+1):
    fact.value *= i
print('Introduce el número que quieres calcular el factorial')
x = int(input())
print("**********************")
fact = mp.Value('i',1)
# Creacion de 2 procesos
p1 = mp.Process(target=fac_concurrente1, args=(fact,x))
p2 = mp.Process(target=fac_concurrente2, args=(fact,x))
# Iniciando los procesos
p1.start()
p2.start()
# Haciendo que el padre espere a los dos procesos
p2.join()
p2.join()
print('Resultado:')
print(fact.value)
print('**********************')
```

```
Introduce el número que quieres calcular el factorial
10
**********************
**********************
Resultado:
3628800
```

c) Implementa la versión en la que se crean cinco hilos que reciben un número real desde el teclado, cada hilo calcula el factorial del número recibido.

```python
#Importamos la libreria para hacer hilos en python.
import threading as th
nums = []
threads = []
n = 5
#Guardamos los valores que queremos caclular.
for i in range(n):
  print('Introduce el {} facotorial que quieres calcular: '.format(i))
  nums.append(int(input()))
  print("-----")
# Creacion de hilos e inicio del proceso.
for i in range(n):
  thread = th.Thread(target=factorial, args=(nums[i],))
  threads.append(thread)
  threads[i].start()
# El padre espera a que los hijos terminen.
for i in range(n):
  threads[i].join()
```

```
Introduce el 0 facotorial que quieres calcular:
2
-----
Introduce el 1 facotorial que quieres calcular:
3
-----
Introduce el 2 facotorial que quieres calcular:
4
-----
Introduce el 3 facotorial que quieres calcular:
5
-----
Introduce el 4 facotorial que quieres calcular:
6
-----
2

6

24

120
720
```

5. Describe la acción de bloqueo, la razón por la cual se utiliza y los métodos que ofrece Python para implementarla

   El bloque es un método que nos permite restringir el acceso de procesos a una cierta sección critia mediante la llamada de la función bloqueo (en el código nosotros indicamos en que parte del código queremos hacer un bloqueo), este método no permite que otro proceso acceda a la parte bloqueada

hasta que el proceso que esta dentro de acceso. Este nos permite que evitar la corrupción de datos en las condiciones de carrera de dos procesos que quieren utilizar variables compartidas.

Python ofrece la clase Lock para la condición de carrera, ofrecera dos estados locked o unlocked, de esta manera poder sincronizar los procesos, esta clase continen los metodos:

• lock.acquire(): Cuando el estado es unlocked, acquire() cambia el estado a locked. Cuando el estado está locked, acquire() bloquea hasta que una llamada a release() en otro hilo lo cambie a unlocked, luego la llamada de acquire() lo restablece a bloqueado y regresa.

• lock.release(): El metodo release(),se utiliza para liberar un bloqueo adquirido, deberia ser solo llamado cuando esta en estado locked ,si no lanzará una exepción a RuntimeError.

6. Describir los siguientes conceptos: Starving, Lock, Sección crítica:

   -Starving: Es cuando un proceso no puede contunuar su ejecución dado que necesita recursos que están asifnados a otros procesos y estos se le son negados.

   -Lock: Es cuado uno o más procesos no pueden acceder a cierta sección del código dado que un proceso esta haciendo uso de esa parte del código y necesita estar ejecundo esa parte del código sin la interrupción de otros procesos.

   -Sección crítica; Es una sección del código en la que se comparte recursos los diferentes procesos o hilos en una ejecución concurrente.

7. Métodos de hilos que nos indican propiedades, características o estado actual de un hilo:

   [x] currentThread() [x] getName() [x] isAlive() [] yield() [] sleep() [] start()

8. Estados de un hilo

   [x]Nuevo [x]Bloqueado [x]En espera [x]Ejecutable []En espera temporizada