

Actividad 6:

Investigación sobre sección crítica

Cómputo de Alto Rendimiento

Luis Fernando Izquierdo Berdugo

Fundamentos teóricos

En programación, la sección crítica es la parte del código que accede a recursos compartidos (como pueden ser variables, archivos o estructuras de datos), esta no debe ser ejecutada simultáneamente por más de un proceso con el fin de evitar condiciones de carrera.

Las condiciones de carrera son un problema que se da en los sistemas cuando dos o más procesos acceden de manera simultánea a recursos compartidos, como su nombre lo dice, es una carrera por acceder a un recurso y si no hay mecanismos de sincronización adecuados puede haber errores impredecibles o difíciles de identificar. En programación paralela es importante controlar el acceso simultáneo a estos recursos para preservar la integridad y consistencia del sistema.

Problemas

Los principales problemas que surgen en relación con la sección crítica son:

- Condiciones de carrera
- Exclusión mutua (mutex), que garantiza que solamente un procesador pueda acceder a la sección crítica.
- Deadlock, que es una situación donde dos o más procesos esperan que el otro libere la sección crítica, lo cual puede provocar que ninguno de los procesos pueda continuar.
- Starvation, cuando un proceso le es negado el acceso a la sección crítica de manera repetida, haciendo que sea imposible que avance.
- Overhead, debido a que los procesos deben obtener y liberar candados de liberación y semáforos al usar las secciones críticas, esto puede impactar el desempeño del programa.

Técnicas y algoritmos

- Semáforos:

Variables enteras utilizadas para controlar el acceso a recursos compartidos. Propuestos por Dijkstra, pueden ser Binarios (como mutexes) o Contadores (permiten más de un acceso simultáneo)

- Mutex (Exclusión Mutua):

Mecanismo de bloqueo utilizado para garantizar que solo un hilo accede a la sección crítica.

- Monitores

Estructuras de alto nivel que combinan variables, procedimientos y la sincronización de acceso en un mismo componente.

- Algoritmos distribuidos

En sistemas distribuidos, donde no hay memoria compartida, se emplean algoritmos como:

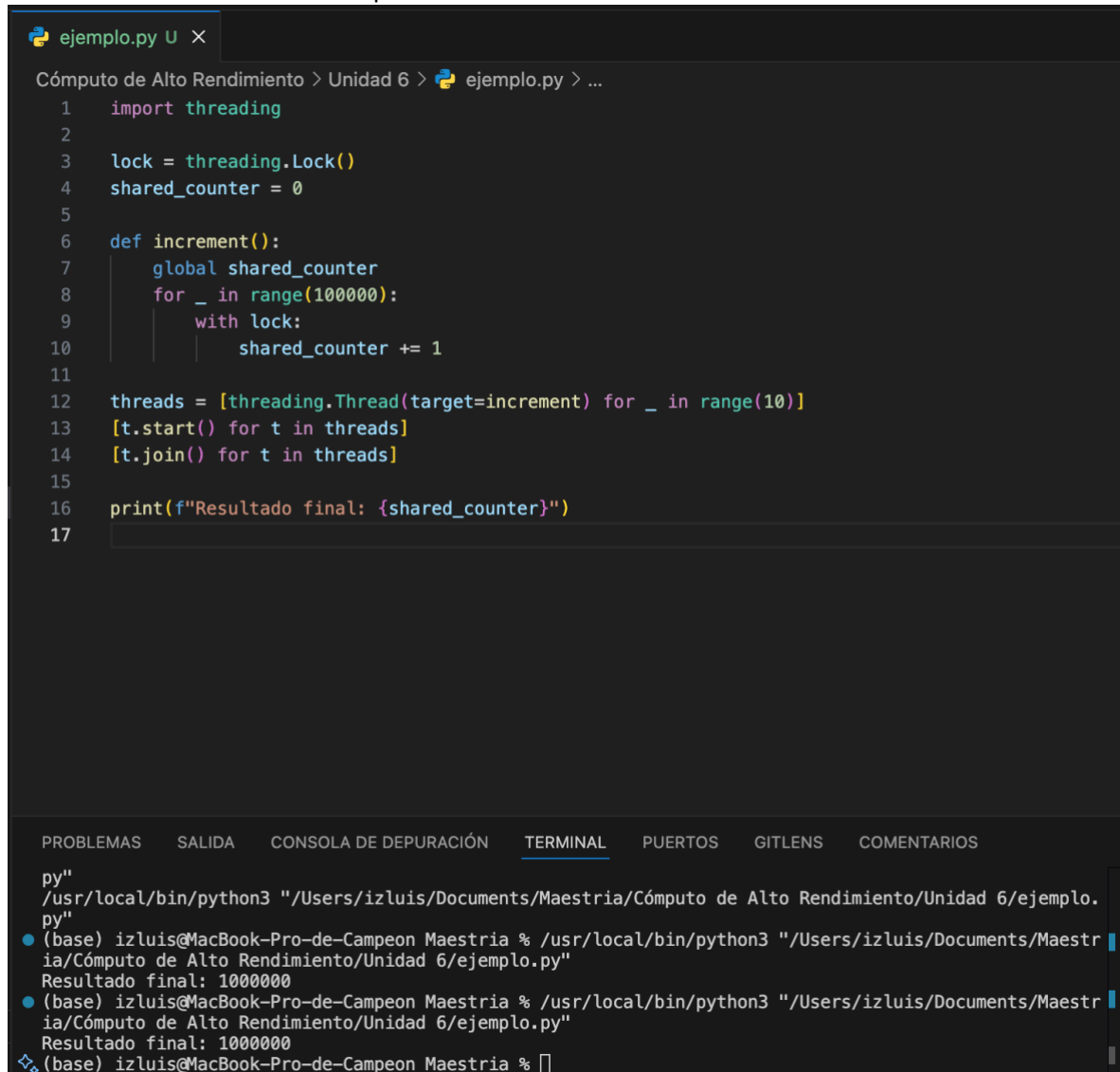
- Ricart-Agrawala
- Token Ring
- Lamport Timestamps

Estos algoritmos usan mensajes y relojes lógicos para coordinar el acceso a la sección crítica entre nodos de una red.

Ejemplo práctico

Para el ejemplo práctico se implemento un contador compartido por múltiples hilos, este utiliza un bloqueo para evitar las condiciones de carrera. La variable “shared_counter” irá incrementando por varios hilos por medio de la función “increment”. Dentro de esta función se utiliza el bloqueo “lock” para proteger la sección crítica, que en este caso es el aumento del contador.

Para ejecutarlo se crean 10 hilos, se inician y se espera a que todos terminen, finalmente imprime el resultado final del contador compartido.



```
ejemplo.py U X
Cómputo de Alto Rendimiento > Unidad 6 > ejemplo.py > ...
1  import threading
2
3  lock = threading.Lock()
4  shared_counter = 0
5
6  def increment():
7      global shared_counter
8      for _ in range(100000):
9          with lock:
10             shared_counter += 1
11
12  threads = [threading.Thread(target=increment) for _ in range(10)]
13  [t.start() for t in threads]
14  [t.join() for t in threads]
15
16  print(f"Resultado final: {shared_counter}")
17

PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS  GITLENS  COMENTARIOS

py"
/usr/local/bin/python3 "/Users/izluis/Documents/Maestria/Cómputo de Alto Rendimiento/Unidad 6/ejemplo.py"
(base) izluis@MacBook-Pro-de-Campeon Maestria % /usr/local/bin/python3 "/Users/izluis/Documents/Maestria/Cómputo de Alto Rendimiento/Unidad 6/ejemplo.py"
Resultado final: 1000000
(base) izluis@MacBook-Pro-de-Campeon Maestria % /usr/local/bin/python3 "/Users/izluis/Documents/Maestria/Cómputo de Alto Rendimiento/Unidad 6/ejemplo.py"
Resultado final: 1000000
(base) izluis@MacBook-Pro-de-Campeon Maestria %
```

Bibliografía

Facultad de Ingeniería - Universidad de la República Uruguay. (s.f.). *Concurrencia y sincronización* [PDF]. <https://www.fing.edu.uy/tecnoinf/mvd/cursos/so/material/teo/so07-concurrencia.pdf>

GeeksforGeeks. (2022, noviembre 30). *Race condition in operating system*.
<https://www.geeksforgeeks.org/g-fact-70/>

Universidad Nacional del Sur. (s.f.). *Sincronización – Problemas clásicos* [PDF].
<https://cs.uns.edu.ar/~gd/soyd/clases/04-SincronizacionProbClasicos.pdf>