Lab. Programación Concurrente

7° Programación Valenzuela/Varela

Programación concurrente

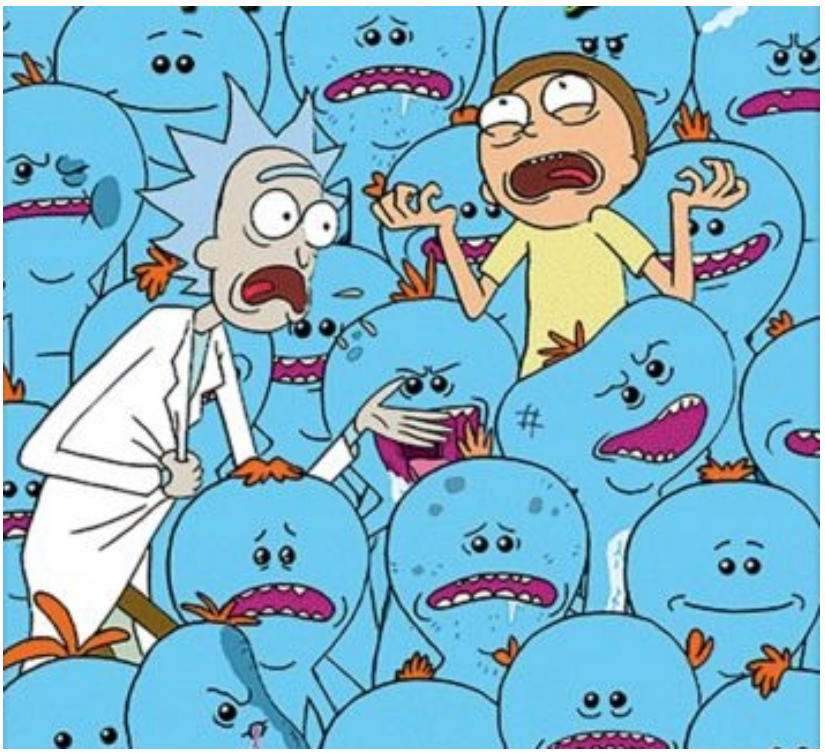
Introducción

Hasta ahora veníamos viendo en las diversas materias de programación programas que de una u otra manera se ejecutaban de manera secuencial o al menos nunca nos preocupamos por que pasaba dentro del cpu al ejecutar un programa. Incluso hemos escuchado hablar de la cantidad de procesadores que tiene una pc pero no lo asociamos directamente a cómo afectaba a los programas que escribimos… hasta hoy

¿Qué significa ***programación concurrente***? Que algo sea concurrente es que ciertos sucesos tienen que acontecer en el mismo momento. Si hablamos de programación los sucesos que van a acontecer en el mismo momento son los programas.

Acá hacemos una pausa para empezar a meter conceptos de la materia y vamos a hacer una distinción entre *programas* y *procesos*. Un programa es algo escrito, algo que se puede ejecutar, de esa ejecución nace un proceso. De esto se deduce que podemos tener un programa y uno o más procesos del mismo. Y justamente esta condición es la que nos va a importar.

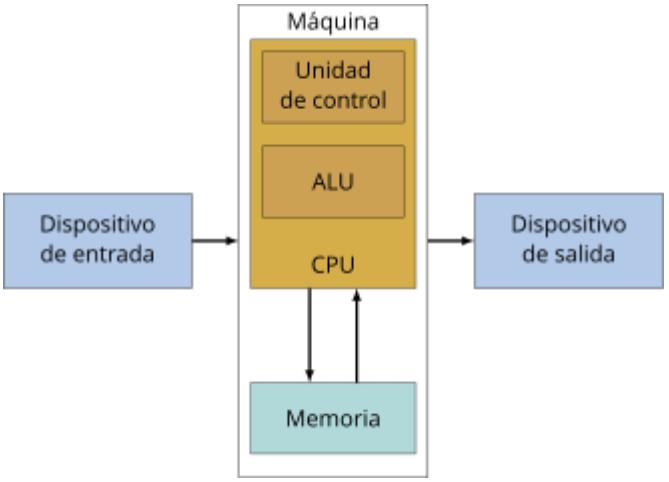
Vamos con un ejemplo de la vida misma. Chrome es un programa, y podemos abrir varias pestañas e incluso varias ventanas con varias pestañas cada una.



Trazas

Otra cosa que puede pasar es que tengamos varios programas ejecutándose al mismo tiempo cada uno con su proceso. Un ejemplo de esto es una pestaña de Chrome reproduciendo música en youtube y un editor de imágenes como puede ser Gimp. ¿Qué pasa en estos casos? ¿Cómo funciona esa concurrencia de programas?

Por si no lo vieron les dejamos un esquema simplificado de la arquitectura de Von Newmann de una computadora.



Lo que nos importa de acá es que el CPU (procesador) solo puede procesar UNA instrucción a la vez. Para ordenar esto el CPU tiene un programa (scheduler) que da tiempos de ejecución a los programas y si no terminan de ejecutarse las instrucciones del programa en ese tiempo le da lugar a otro programa para ser ejecutado. **Es muy importante destacar que en esta materia nuestros scheduler SIEMPRE darán tiempo y posibilidad de ejecutarse a todos los programas que quieran hacerlo.** Lo que no podemos saber es cuándo lo va a hacer, ni cuánto tiempo le va a asignar a cada programa.
En otras palabras el scheduler es un chico caprichoso que da tiempos de ejecución a los programas cuando a él le pinten ganas!!! Scheduler rules!!!

Después de esta ensalada conceptual, vayamos a un ejemplo un poco más concreto. Tenemos 2 programas que están siendo ejecutados, Pepito y Jose. Cada proceso (thread a partir de ahora) es una copia del programa pero está en ejecución y son los siguientes:

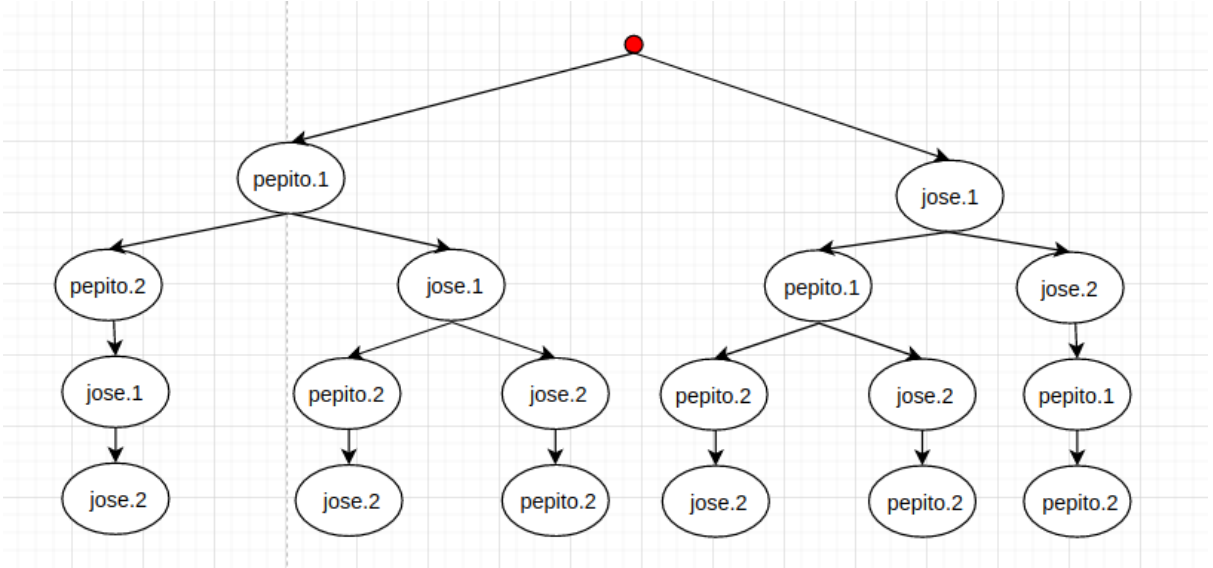
```
thread jose(){
    printf("Hola don Pepito?")
    printf("Adios don Pepito")
}

thread pepito(){
    printf("Hola don Jose?")
    printf("Adios don Jose")
}
```

Ahora viene la parte interesante del asunto. ¿Cuál es la salida de ejecutar esto? Recordemos que el scheduler les va a ir asignando tiempos de ejecución aleatoriamente a ambos threads.
la respuesta esperada sería:

```
(jose.1)    "Hola don Pepito?"
(pepito.1)  "Hola don Jose?"
(jose.2)    "Adios don Pepito"
(pepito.2)  "Adios don Jose?"
```

¿Ahora es real esto?¿Es cierto que el scheduler le va a dar a cada thread el tiempo para ejecutar únicamente una línea de código? Bueno ya anticipamos que esto se iba a complicar…. la respuesta a ambas preguntas claramente en NO!

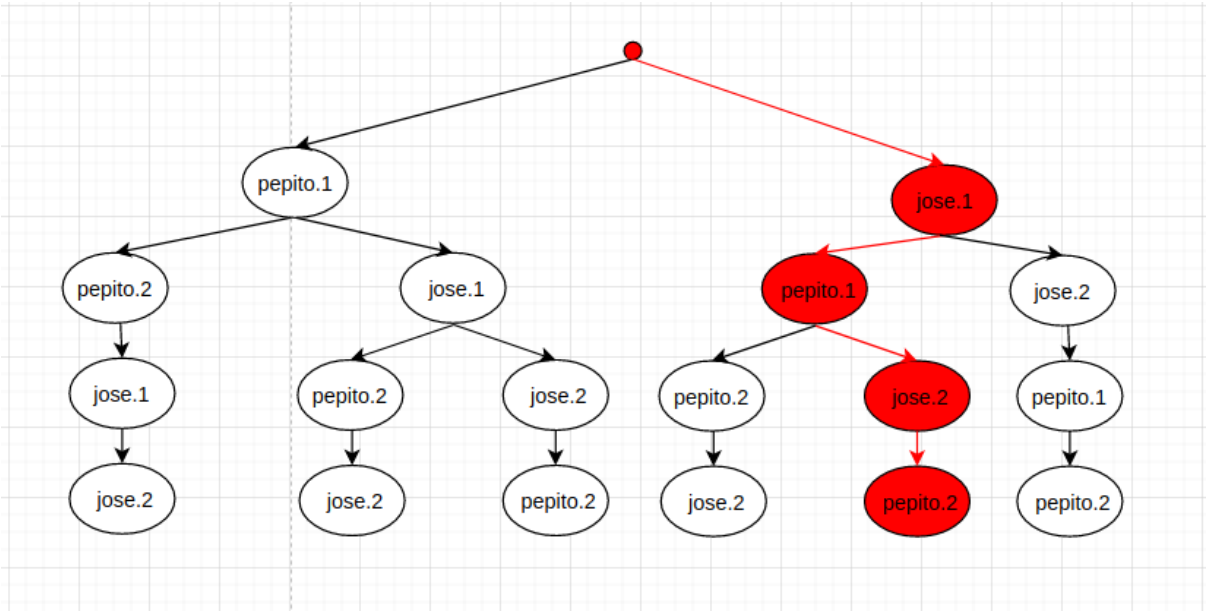
Veamos qué puede pasar con un gráfico de ***trazas***. Para que sea más legible el gráfico vamos a usar por convención nombre.númeroDeLínea y no escribir el resultado del print.



Como vemos la solución es mucho más compleja de lo que aparentaba en un principio y eso que nuestros programas originales tenían solamente 2 líneas de código cada uno de ellos y que no tienen más complejidad que un print…

Último detalle para cerrar esta primera parte.
Llamamos trazas a cada una de las soluciones que pueden conseguirse empezando por el punto rojo y llegando a uno de los óvalos finales.

Dejo como ejemplo la traza que esperábamos obtener



Hasta acá llegamos por esta semana.
Relean el texto e identifiquen los conceptos importantes del mismo:

- Scheduler
- Programa
- Proceso
- Traza