Concurrencia de Procesos: Exclusión Mutua y Sincronización

En la concurrencia existen **términos clave**:

* Sincronización, los procesos coordinan sus actividades.
* Sección crítica, región de código que solo puede ser accedida por un proceso simultáneamente (variables compartidas).
* Exclusión mutua, solo un proceso puede estar en la sección crítica accediendo a recursos compartidos.
* Interbloqueo, varios procesos, todos tienen algo que otros esperan, y a su vez esperan algo de otros.
* Circulo vicioso, los procesos cambian continuamente de estado como respuesta a cambios en otros procesos sin que sea útil.
* Condición de carrera, varios hilos/procesos leen y escriben dato compartido.
* Inanición, proceso que está listo y se le deniega siempre el acceso a un recurso compartido.

En la concurrencia hay que tener en cuenta que la velocidad relativa de los procesos no puede predecirse.

Para la concurrencia de procesos se puede usar la **memoria compartida**.

* La función **shmget(key, longitud, shmflag)** crea un segmento de memoria compartida o solicita accesos a un segmento de memoria existente. La key identifica el segmento, la longitud determina el tamaño de la región compartida y la shmflag es un código que determina si se puede crear un segmento si no existe (IPC\_CREAT), o que devuelva error si ya existe el segmento (IPC\_EXCL | IPC\_CREAT).
* La función **shmat(shmid, shmaddr, shmflag)** añade el segmento de memoria compartida a la memoria del proceso. shmaddr si es 0, el SO trata de encontrar una zona donde “mapear” el segmento compartido.

En la **sincronización**, el sistema operativo se encarga de:

1. Seguir la pista de los procesos activos.
2. Asignar y retirar recursos.
3. Proteger los datos y los recursos físicos.
4. Los resultados de un proceso deben ser independientes de la velocidad relativa a la que se realiza la ejecución de otros procesos concurrentes.

Los procesos **interaccionan** de tres formas:

1. Los procesos no tienen conocimiento de los demás. **Competencia**.

Cuando varios procesos entran en competencia se pueden producir las siguientes situaciones:

* Exclusión mutua. Solo un programa puede acceder a su sección crítica en un momento dado.
* Interbloqueo.
* Inanición.

1. Los procesos tienen un conocimiento indirecto de los otros. **Cooperación por comportamiento**.

Para que los recursos puedan compartir recursos adecuadamente las operaciones de escritura deben ser mutuamente excluyentes.

1. Los procesos tienen un conocimiento directo de los otros (conocen el PID). **Cooperación por comunicación**.

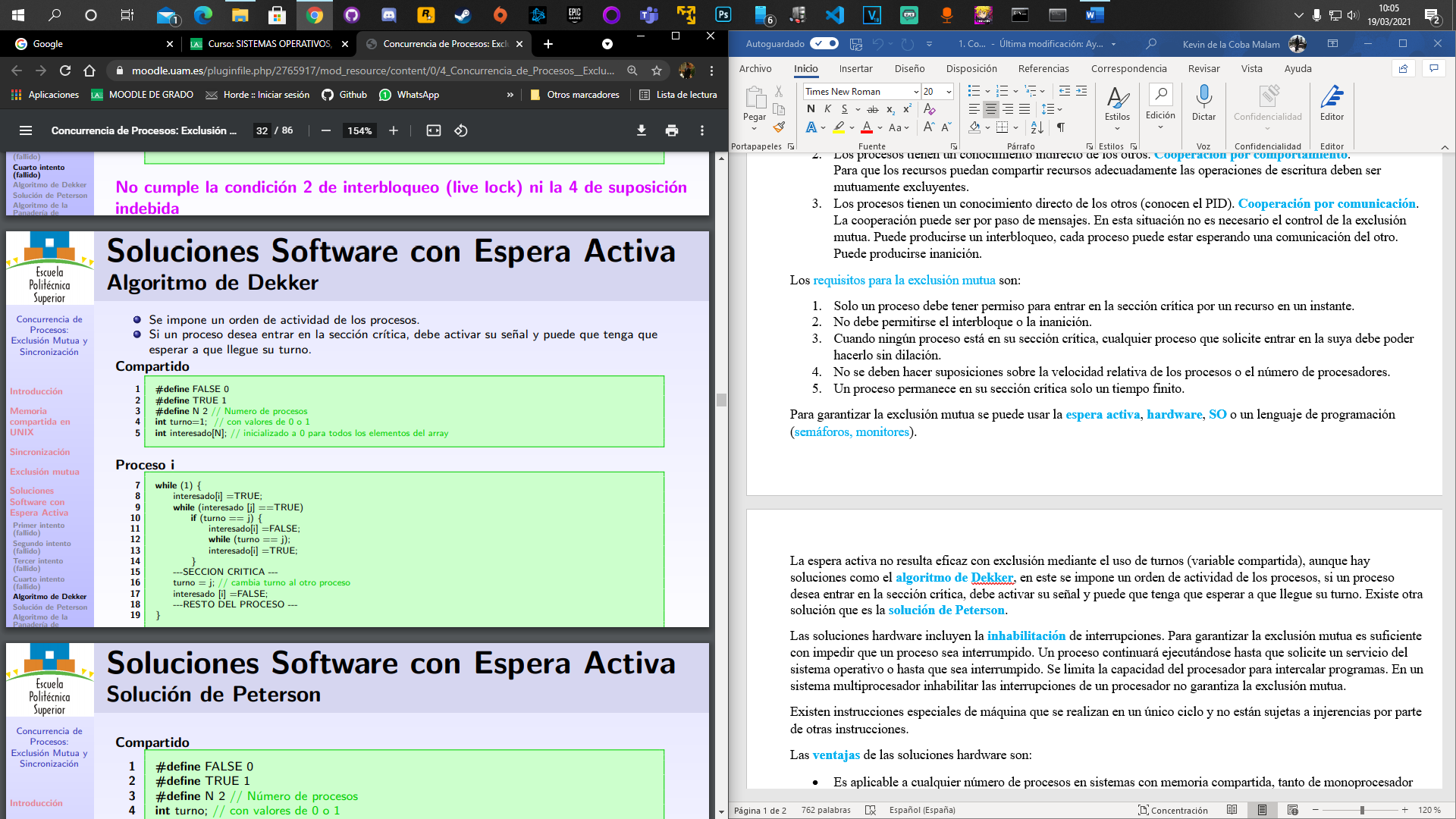
La cooperación puede ser por paso de mensajes. En esta situación no es necesario el control de la exclusión mutua. Puede producirse un interbloqueo, cada proceso puede estar esperando una comunicación del otro. Puede producirse inanición.

Los **requisitos para la exclusión mutua** son:

1. Solo un proceso debe tener permiso para entrar en la sección crítica por un recurso en un instante.
2. No debe permitirse el interbloque o la inanición.
3. Cuando ningún proceso está en su sección crítica, cualquier proceso que solicite entrar en la suya debe poder hacerlo sin dilación.
4. No se deben hacer suposiciones sobre la velocidad relativa de los procesos o el número de procesadores.
5. Un proceso permanece en su sección crítica solo un tiempo finito.

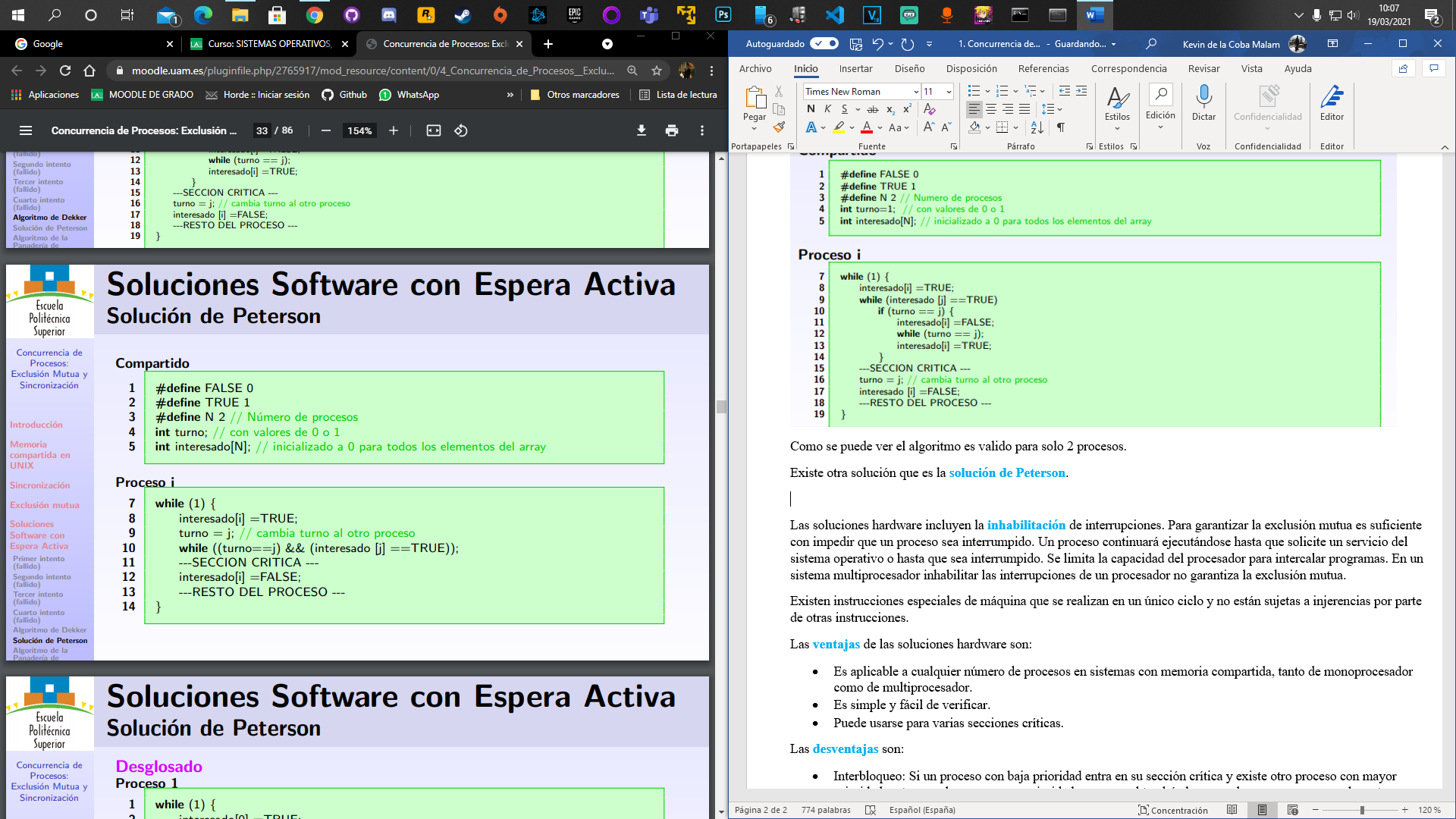
Para garantizar la exclusión mutua se puede usar la **espera activa**, **hardware**, **SO** o un lenguaje de programación (semáforos, monitores).

La espera activa no resulta eficaz con exclusión mediante el uso de turnos (variable compartida), aunque hay soluciones como el **algoritmo de Dekker**, en este se impone un orden de actividad de los procesos, si un proceso desea entrar en la sección crítica, debe activar su señal y puede que tenga que esperar a que llegue su turno.

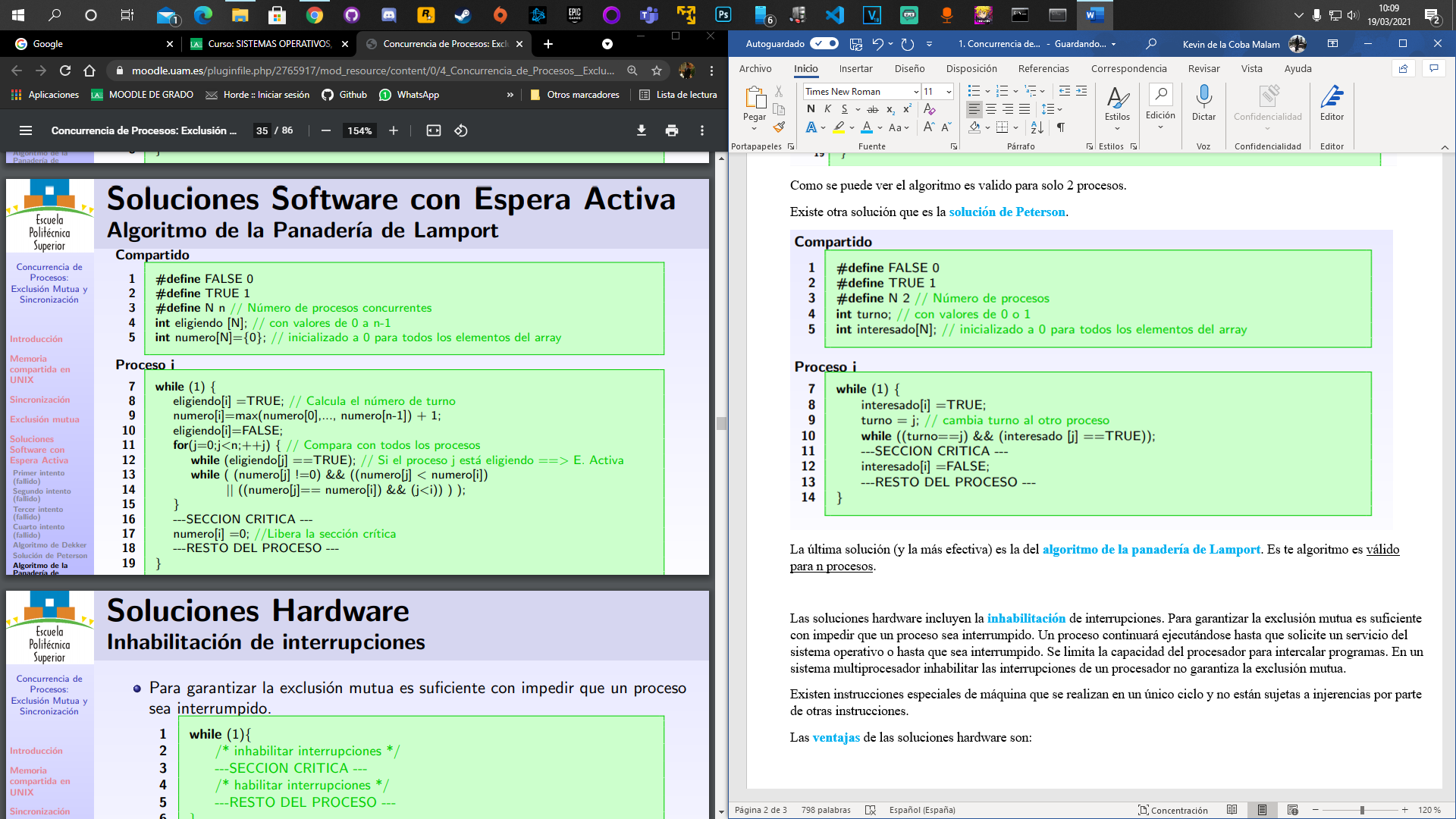


Como se puede ver el algoritmo es valido para solo 2 procesos.

Existe otra solución que es la **solución de Peterson**.

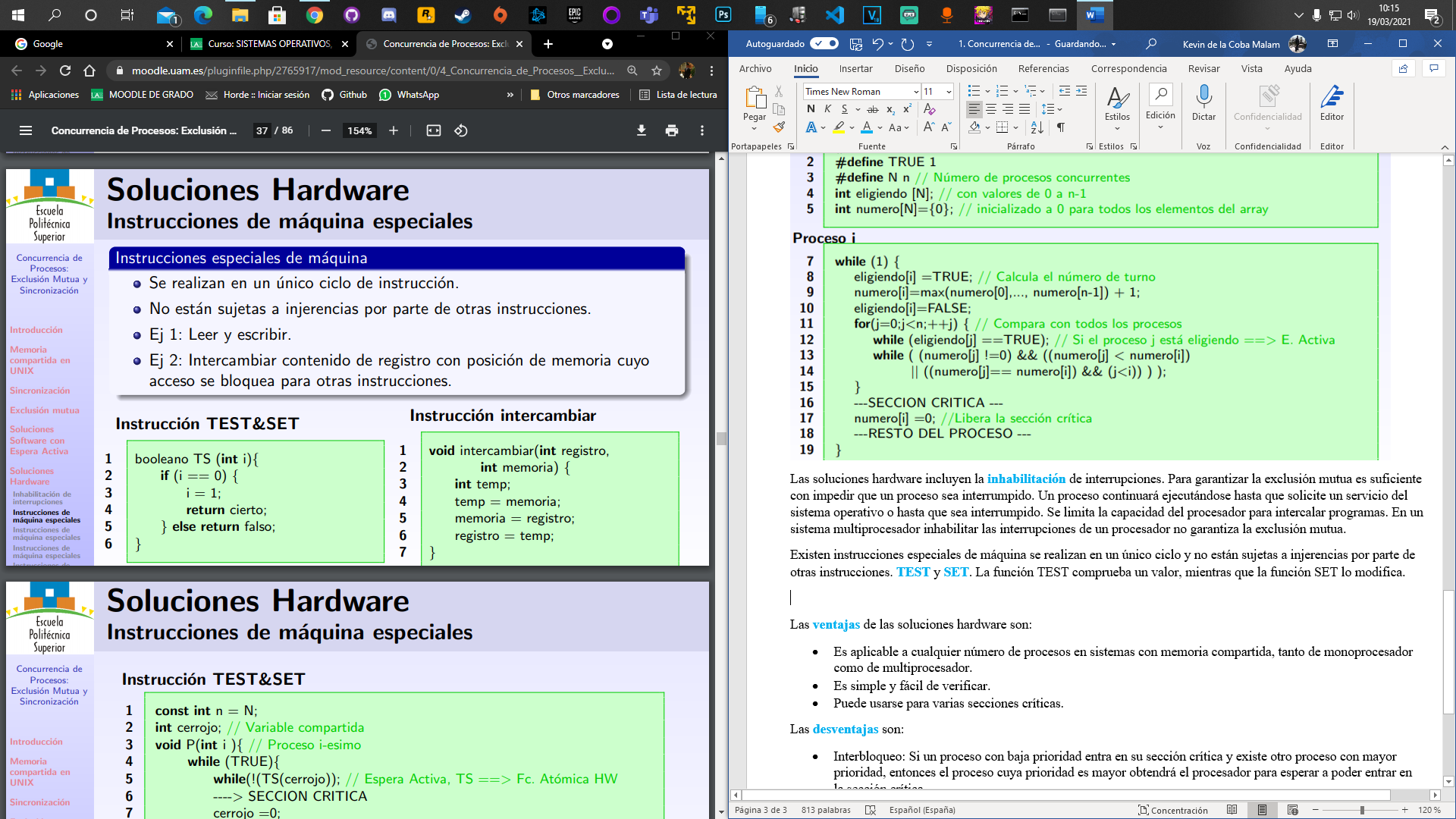


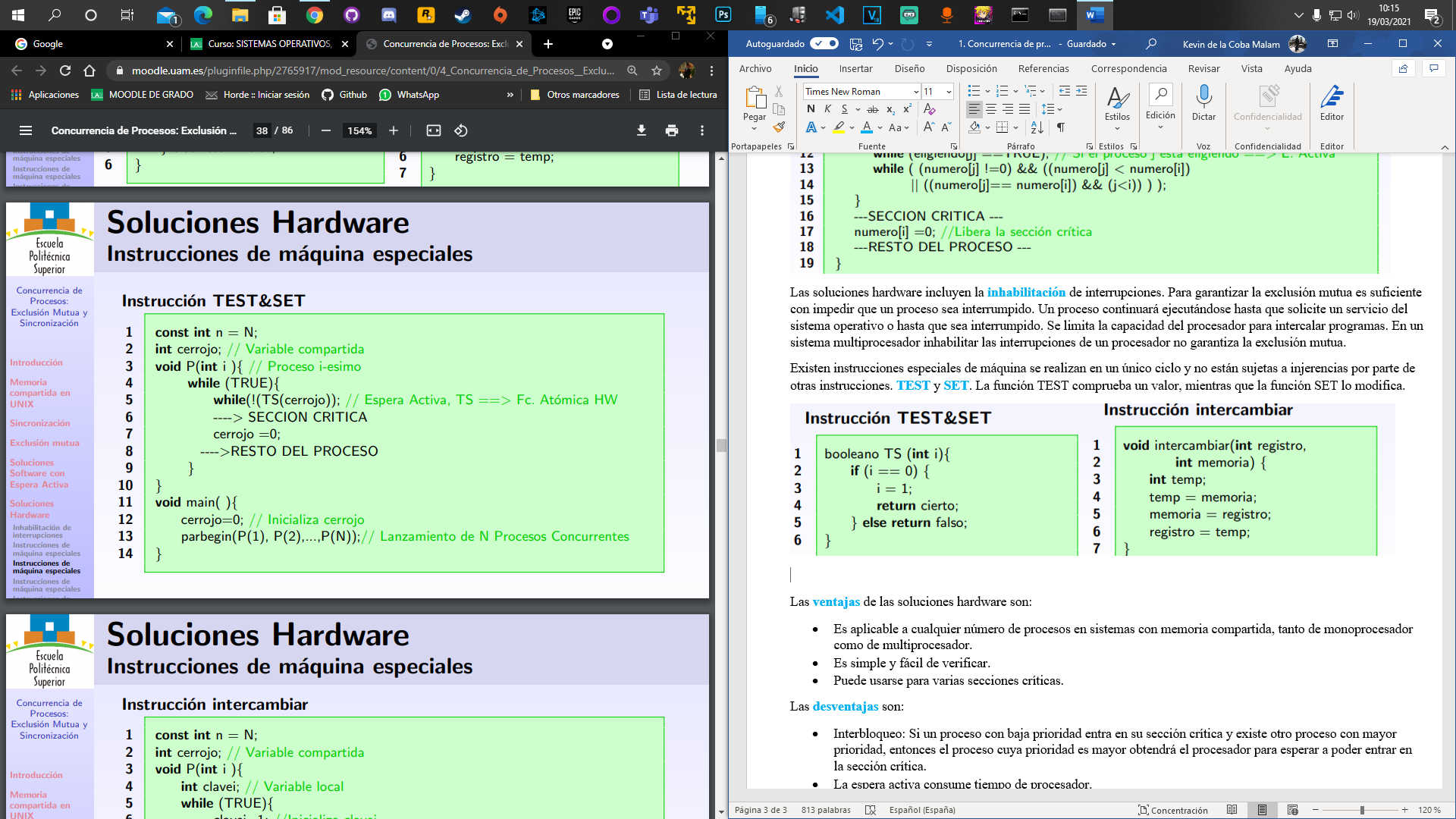
La última solución (y la más efectiva) es la del **algoritmo de la panadería de Lamport**. Es te algoritmo es válido para n procesos.



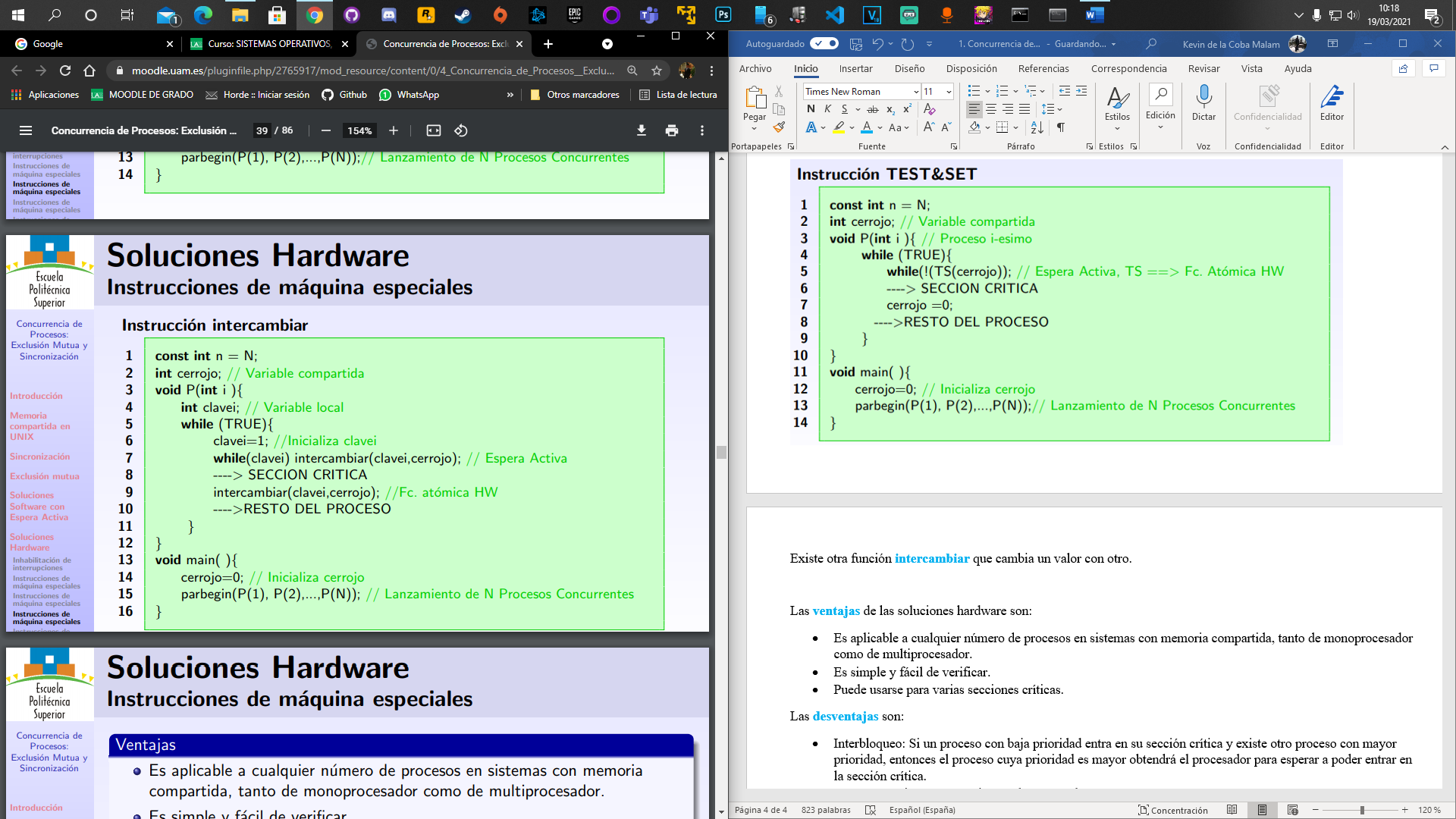
Las soluciones hardware incluyen la **inhabilitación** de interrupciones. Para garantizar la exclusión mutua es suficiente con impedir que un proceso sea interrumpido. Un proceso continuará ejecutándose hasta que solicite un servicio del sistema operativo o hasta que sea interrumpido. Se limita la capacidad del procesador para intercalar programas. En un sistema multiprocesador inhabilitar las interrupciones de un procesador no garantiza la exclusión mutua.

Existen instrucciones especiales de máquina se realizan en un único ciclo y no están sujetas a injerencias por parte de otras instrucciones. **TEST** y **SET**. La función TEST comprueba un valor, mientras que la función SET lo modifica.





Existe otra función **intercambiar** que cambia un valor con otro.



Las **ventajas** de las soluciones hardware son:

* Es aplicable a cualquier número de procesos en sistemas con memoria compartida, tanto de monoprocesador como de multiprocesador.
* Es simple y fácil de verificar.
* Puede usarse para varias secciones críticas.

Las **desventajas** son:

* Interbloqueo: Si un proceso con baja prioridad entra en su sección crítica y existe otro proceso con mayor prioridad, entonces el proceso cuya prioridad es mayor obtendrá el procesador para esperar a poder entrar en la sección crítica.
* La espera activa consume tiempo de procesador.