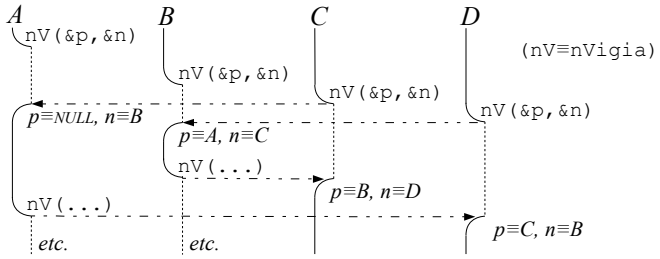
Pregunta 1

Parte a.- (4 puntos) En el Titanic se necesita que siempre hayan 2 vigías en busca de icebergs. Los vigías son representados por tareas que llaman cada cierto tiempo a *nVigia*, una nueva función de nSystem:

```
void nVigia(nTask *pprev, nTask *pnext);
```

Cuando una tarea *T* invoca *nVigia*, la tarea *T* se supende hasta que sea relevada luego de que otras 2 tareas llamen a *nVigia*. Mientras tanto la tarea *T* actúa como vigía junto a la tarea que había llamado previamente a *nVigia* y más tarde junto a la próxima tarea que llame a *nVigia*. En ***pprev* y **pnext*** se entregan los identificadores de las 2 tareas que acompañaron a *T* durante su vigilancia. El siguiente diagrama de tareas muestra un ejemplo de ejecución:



Observe que la primera tarea que invoca nVigia (tarea A) no tiene acompañante previo y por lo tanto *p≡NULL*. La primera flecha punteada indica que en ese instante la tarea C releva a A, la segunda flecha indica cuando D releva a B, etc. Los acompañantes de B fueron primero A y luego C. Note además que eso de ser vigía es solo una metáfora: en la práctica un vigía no hace nada, solo duerme.

Implemente esta API usando los procedimientos de bajo nivel de nSystem (*START_CRITICAL*, *Resume*, *PutTask*, etc.). Ud. *no puede usar* otros mecanismos de sincronización ya disponibles en nSystem, como semáforos, monitores, mensajes, etc. Ud. necesitará agregar nuevos campos al descriptor de tarea de nSystem, nuevos estados, nuevas variables globales, etc.

Parte b.- (1 punto) ¿Cuál es la ventaja número 1 de un núcleo clásico por sobre un núcleo moderno? ¿Y cuál es la desventaja número 1?

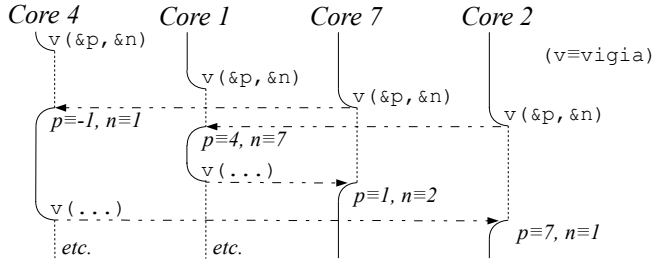
Parte c.- (1 punto) ¿Tiene sentido usar spin-locks en una máquina monocore? Explique. Responda entonces cómo implementaría la exclusión mutua en un núcleo moderno de Unix para una máquina monocore.

Pregunta 2

I. (4 puntos) Resuelva el mismo problema de los vigías del Titanic considerando ahora una máquina *octa-core* en la que no existe un núcleo de sistema operativo y por lo tanto la única herramienta de sincronización preexistente es el *spin-lock*. La función que Ud. debe implementar es la siguiente:

```
void vigia(int *pprev, int *pnext);
```

En ***pprev* y **pnext*** se entregan los números de los cores que acompañaron al core que invocó *vigia*. El siguiente es un ejemplo de ejecución (equivalente al de la pregunta 1):



Observe que el primer core que invoca *vigia* (Core 4) no tiene acompañante previo y por lo tanto *p≡-1*.

Para programar su solución Ud. dispone de *spin-locks* y la función *coreId()* que entrega el número del core que la invoca. Necesitará declarar variables globales.

Restricción: Dado que no hay un núcleo de sistema operativo, la única forma válida de esperar a ser relevado es utilizando un spin-lock. Otras formas de *busy-waiting* no están permitidas.

II. (1 punto) ¿Cuantos timers se necesitan en una máquina octa-core para implementar el núcleo de un sistema operativo? Explique por qué.

III. (1 punto) ¿Cómo se evita que un proceso del usuario tenga acceso a la cola de procesos ready, a los descriptores de proceso, al vector de interrupciones, etc.? ¿Pero si no tiene acceso, cómo logra la función *fork* crear un nuevo descriptor de proceso y agregarlo a la cola de procesos ready? Explique.