Ejercicio 1
El sistema operativo gestiona diversas estructuras de datos, cuyos tamaños dependen de las características específicas de cada una de ellas. Vamos a estimar el tamaño de algunas de estas estructuras en un sistema como el siguiente:

- Se trata de un multiprocesador de memoria compartida con *N* procesadores.
- Se gestionan pilas de sistema separadas: la de interrupción para manejar las interrupciones de los dispositivos y la de sistema, propiamente dicha, para manejar los eventos restantes.
- Utiliza un esquema de planificación del procesador que minimiza la congestión en el acceso a las estructuras de datos del planificador y aprovecha lo mejor posible la afinidad de los procesos.
- Usa un esquema de memoria virtual basado en paginación por demanda, con tabla de páginas de usuario y de sistema independientes.
- En un determinado instante, hay *P* procesos de usuario con un número medio de *T threads* (incluido el implícito) por proceso, estando T_L en estado de listos y T_B bloqueados.
- Además, existen *K* procesos de núcleo, cada uno de los cuales sólo tiene el *thread* implícito asociado, habiendo K_L listos para ejecutar y K_B bloqueados.

Responda de forma razonada, pero breve, a las siguientes cuestiones:
a) ¿Cuántos BCP y BCT habrá reservados en el sistema?
b) Responda a la pregunta anterior suponiendo que se trata de un sistema que no soporta directamente *threads*, sino que lo hace indirectamente usando procesos de peso variable, como ocurre en Linux.
c) ¿Cuántas pilas de sistema y de interrupción existirán en el sistema?
d) En cuanto a la información que se almacena en dichas pilas, ¿en qué estados del proceso el contenido de la pila puede ser distinto dependiendo de que el núcleo sea expulsivo o no expulsivo, y en qué consistiría esa diferencia? Para responder, tenga en cuenta sólo los estados de bloqueado, listo habiendo estado previamente bloqueado y listo habiendo estado previamente en ejecución.
e) ¿Cuántas colas de planificación habrá en el sistema y cuál es el número total de *threads* incluidos en las mismas?
f) Mientras un *thread* en modo sistema mantiene un *spinlock* durante una sección crítica, ¿cuál es el número máximo de *threads* que pueden estar realizando una espera activa sobre el mismo? Y si usa un semáforo interno, ¿cuántos *threads* pueden llegar a estar bloqueados (espera pasiva) en el mismo mientras el *thread* que ejecuta la sección crítica lo tiene asignado?
g) Suponiendo que todos los programas que se están ejecutando en este momento son distintos, ¿cuántos ficheros ejecutables estarán involucrados?
h) ¿Cuántas tablas de páginas habrá en el sistema?
i) ¿Cuántas tablas de marcos existirán en el sistema?
j) Supóngase que en un instante dado se están ejecutando simultáneamente *M threads* ($M \leq N$) del mismo proceso. ¿Qué valores tomarían los registros base de la tabla de páginas de cada procesador involucrado? Considere que en un determinado momento se expulsa a memoria secundaria una página del mapa de ese proceso, ¿qué acciones se deberían realizar en el nivel de tabla de páginas y cuáles en el nivel de TLB (recuerde que hay una TLB en cada procesador)?

Solución

a) Cada proceso, sea de usuario o de núcleo, requiere un bloque de control de proceso (BCP). Por tanto, se usarán (P+K) BCP en la situación descrita. En cuanto a bloques de control de *thread* (BCT), se necesita uno por cada *thread*, incluido el propio *thread* implícito de cada proceso, por lo que habrá tantos como *threads*, con independencia de si pertenecen a proceso de usuario o de núcleo (P*T+K).

b) En este caso, no se usan bloques de control de *thread*, puesto que en estos sistemas la abstracción de *threads* del mismo proceso se implementa con procesos que comparten todos los recursos requeridos por esta abstracción (ficheros abiertos, mapa de memoria, manejo de señales, etc.). Por tanto, en este caso todo *thread* requiere un BCP (P*T+K).

c) Cada *thread* requiere su propia pila de sistema (en total, P*T+K), puesto que la necesita para poder reanudar correctamente su ejecución después de haber estado bloqueado. En cuanto a pilas de interrupción, dado que no puede haber un cambio de contexto durante el tratamiento de las interrupciones de dispositivos, no se requiere tener una pila de este tipo por cada *thread*. Sin embargo, sí se necesita una pila de interrupción por cada procesador, puesto que en un instante dado, se puede estar sirviendo una interrupción en cada procesador (por tanto, *N* pilas de interrupción).

d) En un núcleo expulsivo, un *thread* listo que previamente ha estado en ejecución puede haberse quedado en medio de la ejecución de una llamada al sistema debido a un cambio contexto involuntario, por lo que el contenido de su pila de sistema incluirá la información necesaria para proseguirla cuando vuelva a ejecutar. Sin embargo, en un núcleo no expulsivo no va a estar presente en ningún caso este tipo de información en la pila, puesto que el cambio de contexto involuntario sólo se lleva a cabo cuando el proceso va a retornar a modo usuario.
En el caso de un proceso bloqueado o listo habiendo estado previamente bloqueado, al corresponder a un cambio de contexto voluntario, la información que se almacena en la pila es la misma con independencia de si se trata de un núcleo expulsivo o no.

e) Al tratarse de un esquema de planificación que minimiza la congestión en el acceso a las estructuras de datos del planificador y aprovecha lo mejor posible la afinidad de los procesos, usará una cola de planificación por cada procesador (*N* colas de planificación). En esas colas de planificación estarán incluidos todos los *threads* listos para ejecutar que hay en el sistema (un total de $P*T_L + K_L$).

f) Mientras un *thread* en modo sistema ejecutando en un determinado procesador mantiene un *spinlock* durante una sección crítica, puede haber espera activa en todos los procesadores restantes si en cada uno de ellos está ejecutando un *thread* que realiza una llamada al sistema que requiere el uso de ese *spinlock* (un total de *N-1 threads*).
Si se utiliza un semáforo para la sincronización interna, en el peor caso, todos los *threads*, excepto el que posee el semáforo, pueden estar bloqueados esperando en el mismo. En el escenario planteado, todos los *threads* bloqueados podrían estar esperando en el mismo semáforo ($P*T_B + K_B$).

g) Cada proceso de usuario tiene asociado un programa (en este caso, distinto, según especifica el enunciado). Sin embargo, los procesos de núcleo no están vinculados a ningún programa, sino que su ejecución se restringe a código del propio sistema operativo. Por tanto, en el escenario planteado, habrá *P* ficheros ejecutables involucrados.

h) Cada proceso de usuario tiene asociada una tabla de páginas de usuario, pero comparten la misma tabla de páginas de sistema. En cuanto a los procesos de núcleo, no requieren una tabla de páginas de usuario, puesto que su ejecución se circunscribe al propio sistema operativo, compartiendo la tabla de páginas de sistema con todos los demás procesos. En consecuencia, se necesitan *P* tablas de páginas de usuario y 1 de sistema.

i) Dado que se trata de un multiprocesador con memoria compartida, sólo se requiere 1 tabla de marcos para mantener el estado de la memoria física disponible en el sistema.

j) Todos los *threads* del mismo proceso comparten el mapa de memoria, por lo que en los *M* procesadores que los ejecutan estará instalada la misma tabla de páginas de usuario. Por tanto, el registro base de la tabla de páginas de usuario de cada procesador involucrado hará referencia a la misma tabla de páginas (nótese que el registro base de la tabla de páginas de sistema también apuntará a la misma tabla, pero eso ocurre con todos los procesos).
Si se produce la expulsión de una página que pertenece al proceso especificado, habrá que invalidar la entrada correspondiente de la tabla de páginas del proceso. Este cambio es visible inmediatamente en los *M* procesadores involucrados, puesto que comparten la tabla de páginas. Además, es necesario eliminar de la TLB de cada uno de esos procesadores la entrada asociada a esa página, en caso de que existiera, puesto que la información de traducción ya no es válida al haber sido expulsada. Para ello, habrá que enviar una IPI a cada uno de los procesadores afectados, de manera que en el tratamiento de dicha interrupción se lleve a cabo la invalidación en la TLB de la entrada correspondiente a la página expulsada (nótese que, en caso de que el procesador donde se tomó la decisión de expulsar la página fuera uno de los *M* afectados, no se requeriría una IPI para que dicho procesador invalidara la entrada correspondiente de su TLB, sino que esa operación se llevaría a cabo directamente en el contexto de la expulsión).