# DISEÑO DE SISTEMAS OPERATIVOS. Febrero de 2003.

## Ejercicio 1

Responda razonadamente a las siguientes preguntas sobre gestión de procesos y memoria.

**a)** Considérese un sistema con las siguientes características:

- 2 interrupciones externas: de teclado y de reloj, esta última tiene mayor prioridad.
- S.O. no expulsivo (sin llamadas concurrentes); para ello se usa el mecanismo de interrupción software.
- Planificación expulsiva basada en prioridad con *round-robin* para procesos de igual prioridad.
- Hay paginación por demanda existiendo, por tanto, una excepción de fallo de página. Se supone que el código y datos del S.O. están residentes y no hay errores en el código del S.O. Por tanto, los fallos de página se corresponden siempre con accesos al mapa de memoria del proceso, aunque pueden producirse dentro de rutinas del S.O. cuando se intenta acceder desde ellas al mapa del proceso.

Se plantean las siguientes cuestiones sobre el nivel de anidamiento (o sea, el número de activaciones del S.O. anidadas; un nivel 0 corresponde con un proceso en modo usuario) que hay en el sistema en cada momento.

**a1) [Puntuación: 1,5/10]** Calcule cuál es el nivel máximo de anidamiento. A continuación, analice de forma razonada si puede ocurrir que la rutina de tratamiento de un fallo de página se anide dentro de las siguientes rutinas del S.O.: llamada al sistema, rutina de interrupción de teclado o rutina de interrupción software

**a2) [Puntuación: 3,5/10]** Cuando se produce un cambio de contexto, determine qué posibles niveles de anidamiento tendrá el proceso...

1. que deja el procesador en un cambio de contexto voluntario.
2. que deja el procesador en un cambio de contexto involuntario.
3. que obtiene el procesador en el cambio de contexto. Se debe distinguir si dicho proceso estaba previamente bloqueado o no. Además, para cada uno de estos dos casos (bloqueado o no), se debe explicar en qué punto de su ejecución estaba parado el proceso (o sea, adónde apuntaba la copia del contador de programa que estaba guardada en el BCP) en el momento de realizarse el cambio de contexto involuntario.

**b)** Sea un proceso `Pr1` en cuyo mapa de memoria existe una región privada con soporte en fichero.

**b1) [Puntuación: 1/10]** El proceso va invocar la llamada `EXEC`. ¿En qué distintas ubicaciones pueden estar las páginas de la región en el instante previo?¿Qué tratamiento se realiza sobre esa región durante la llamada `EXEC`?

**b2) [Puntuación: 3/10]** Supóngase que la región está formada por 3 páginas y que se ejecuta la siguiente traza:

Pr1: lee de página 1
Pr1: escribe en página 2 y página 3
Pr1: fork ® Pr2
Pr1: escribe en página 1 y página 2
Pr2: escribe en página 2
Pr2: fork ® Pr3
Pr2: escribe en página 1 y página 3
Pr3: lee de página 1, página 2 y página 3

Suponiendo que después de ejecutar esa traza entran a ejecutar otros procesos que expulsan las páginas de estos procesos, se debe calcular cuántos bloques de *swap* se dedican en total a esta región y, para cada proceso, cuál es la ubicación de cada página de la región, identificando en qué bloque del fichero o del *swap* está almacenada (numere los bloques del *swap* como considere oportuno).

**b3) [Puntuación: 1/10]** Dado cómo se usan habitualmente las llamadas `FORK` y `EXEC` en las aplicaciones, algunos diseñadores proponen que sería más eficiente que después del `FORK` se ejecutara primero el proceso hijo en vez del padre. Analice en qué puede basarse esta propuesta.

## Solución

**a1)** El máximo nivel de anidamiento es 4 y corresponde con una situación en la que se produce la siguiente secuencia de

eventos:

1. Se produce una llamada al sistema.
2. La llamada requiere el acceso al mapa de memoria del proceso (por ejemplo, una llamada *open* que intenta acceder al nombre del fichero o una llamada *read* o *write* que accede al buffer) y la página correspondiente no está residente. Se produce un fallo de página anidado.
3. Mientras se ejecuta la rutina de tratamiento del fallo, se produce una interrupción del teclado que causa la activación anidada de la rutina de tratamiento.
4. Durante la ejecución de la rutina, se genera una interrupción de reloj que, al ser de prioridad máxima, causa la activación inmediata de su rutina de tratamiento.

Con respecto al posible anidamiento del tratamiento de un fallo de página dentro de las rutinas planteadas en el enunciado:

1. Como se comentó previamente, durante la ejecución de una llamada al sistema se puede producir un fallo de página al intentar acceder al mapa del proceso.
2. La rutina de interrupción del teclado, como todas las interrupciones, tiene un carácter asíncrono. Esto implíca que el proceso que está en ejecución cuando se activa la rutina de interrupción no está relacionado con la misma. Por tanto, el código de la rutina no va a acceder a su mapa de memoria y no se podrá producir un fallo de página.
3. Este mismo razonamiento es aplicable a la rutina de tratamiento de la interrupción software. Este tipo de interrupción de baja prioridad se usa para ejecutar de forma diferida las operaciones menos urgentes asociadas a una interrupción de mayor prioridad (entre ellas, acciones de planificación), por lo que tiene también carácter asíncrono y no accederá al mapa del proceso.

**a2)**

1. El cambio de contexto voluntario se produce dentro del ámbito de una llamada al sistema o dentro de un fallo de página .El nivel de anidamiento del proceso que deja el procesador en un cambio de contexto voluntario puede ser: o bien una sola activación del sistema operativo, si el proceso se ha bloqueado en una llamada al sistema o en un fallo de página producido en modo usuario, o bien dos activaciones, en el caso de que se haya bloqueado dentro del tratamiento de un fallo generado en modo privilegiado por una llamada al sistema.

2. Un cambio de contexto involuntario en un sistema que usa el algoritmo de planificación planteado ocurre o bien cuando se desbloquea un proceso de mayor prioridad que el que está ejecutando en ese momento, ya sea debido a una interfrupción o a una llamada al sistema, o bien cuando se le termina el turno al proceso en ejecución. Por tanto, la detección de que se requiere un cambio de contexto involuntario se puede dar en cualquier nivel de anidamiento desde 1 hasta 4. Sin embargo, dado que se trata de un sistema operativo no expulsivo, en ese instante sólo se activa una interrupción software, lo que conlleva que el cambio de contexto se difiera hasta que termine el anidamientos actual y se llevará a cabo dentro de la rutina de tratamiento de la interrupción software. Por tanto, sea cual sea el motivo del cambio de contexto involuntario, éste se produce con un nivel de anidamiento 1, que corresponde con el tratamiento de la interrupción software.

3. Si el proceso que obtiene el procesador en un cambio de contexto, ya sea voluntario o involuntario, estaba previamente bloqueado antes de pasar al estado de listo para ejecutar, su nivel de anidamiento, como se comentó en el apartado 1, puede ser igual a 1 o a 2. El proceso estará parado en la operación de cambio de contexto asociada al bloqueo y, por tanto, el contador de programa apuntará a la instrucción siguiente al cambio de contexto. Este bloqueo puede haberse producido dentro de una llamada al sistema (nivel 1) o durante la ejecución de la rutina de tratamiento de un fallo de página (nivel 1 ó 2) si se requiere la lectura o escritura de una página.

   En el caso de que el proceso que obtiene el procesador haya estado previamente en ejecución, como se explicó en el apartado 2, se habrá detenido en una operación de cambio de contexto invocada desde la rutina de interrupción software, y continuará su ejecución con un nivel de anidamiento igual a 1. El contador de programa apuntará a la instrucción que sigue al cambio de contexto.

   Como última posibilidad, se presenta la situación de un proceso recién creado. En este caso, el sistema operativo prepara el contexto inicial de ejecución del proceso de manera que comience ejecutando una rutina interna del sistema que luego da paso al código inicial del programa. Por tanto, la ejecución del proceso se inicia con nivel de anidamiento igual a 1.

**b1)** En un determinado momento, el contenido actualizado de una página de la región puede estar en las siguientes ubicaciones:

- En memoria principal.
- En el soporte original, si la página no se ha modificado.

- En *swap*, en caso de que la página se haya modificado en algún momento.

Una llamada EXEC causará la liberación del mapa actual del proceso. Por tanto, se eliminará esta región liberando tanto los marcos de memoria como los bloques de *swap* que pueda ocupar. Recuérdese que al tener carácter privado no deben salvarse las modificaciones realizadas sobre la región.

**b2)** A continuación, se analizará la traza de ejecución de los procesos planteados:

- Antes de la llamada a FORK, el proceso Pr1 ha modificado las páginas 2 y 3. Si en algún momento fueran expulsadas estas páginas, serán escritas en *swap*.
- Como resultado del FORK, los procesos Pr1 y Pr2 comparten la región pero marcándose como *copy-on-write* (COW)
- Después del FORK, Pr1 modifica las páginas 1 y 2. Como están marcadas como COW, se hace un duplicado de esta páginas para Pr1.
- El proceso Pr2 modifica la página 2 pero esta operación está asociada sólo a Pr2.
- Antes del segundo FORK, como resultado de las operaciones ya analizadas, Pr1 tiene su propia copia de la página 1 y de la 2 y Pr2 de la página 2. Ambos procesos comparten la página 3 marcada como COW. Por lo que se refiere a la página 1 del proceso Pr2, todavía está vinculada al fichero original.
- Después del segundo FORK, Pr2 y Pr3 comparten las 3 páginas usando COW. A continuación, Pr2 escribe en las páginas 1 y 3 creándose una copia propia. Las lecturas finales de Pr3 no cambian el estado de ninguna página.

Si después de esta secuencia se ejecutan otros procesos que causan la expulsión de todas las páginas residentes de Pr1, Pr2 y Pr3, todas las páginas modificadas se escribirán al *swap* quedando la siguiente situación (en la que los números de bloque del *swap* se han asignando arbitrariamente):

- Pr1: página 1 en el bloque 1 del *swap*, página 2 en el bloque 2 y página 3 en el bloque 3.
- Pr2: página 1 en el bloque 4 del *swap*, página 2 en el bloque 5 y página 3 en el bloque 6.
- Pr3: página 1 en el bloque 1 del fichero (antes de crear este proceso, no se ha modificado esta página), página 2 en el bloque 5 (compartida con Pr2 usando COW) y página 3 en el bloque3 (compartida con Pr1 mediante COW).

Se dedican, por tanto, 6 bloques de *swap* a la región.

**b3)** En la mayoría de las ocasiones después de un FORK el proceso hijo invoca enseguida la llamada EXEC liberando su mapa de memoria. Por tanto, en ese corto intervalo que discurre entre su arranque inicial y la llamada a EXEC, normalmente modifica un número muy reducido de páginas de su mapa de memoria.
Si ejecuta primero el proceso padre, éste modificará numerosas páginas de su mapa produciéndose el duplicado de las mismas mediante la técnica COW. Cuando le toque ejecutar al hijo, prácticamente no usará su mapa cambiándolo cuando llegue al EXEC.
En caso de que ejecute en primer lugar el proceso hijo, antes de llamar a EXEC podría causar el duplicado de un número muy reducido de páginas para, a continuación, liberar su mapa como consecuencia del EXEC, Por tanto, las regiones del mapa del proceso padre ya no se comparten y cuando éste se ejecute no se requerirá duplicar las páginas. Este es el beneficio de esta estrategia.