PREGUNTAS DE AUTOEVALUACIÓN DE FUNDAMENTOS BÁSICOS DE SISTEMAS OPERATIVOS

Unidades y funciones matemáticas usadas en el texto

Las unidades básicas de almacenamiento u organización de la información son el *bit* (símbolo b) Y el *byte* (símbolo B). Un bit es un dígito binario, es decir, un 0 o un 1, que se utiliza para representar una pareja de posibles estados, por ejemplo: encendido y apagado, presente y ausente, etc. Un byte son ocho bits contiguos.

Prefijo Símbolo Valor Sistema de unidades

Prefijo	Símbolo	Valor	Sistema de unidades
kibi	Ki	2^10	IEEE 1541
kilo	K	10^3	SI
mebi	Mi	2^20	IEEE 1541
mega	M	10^6	SI
gibi	Gi	2^30	IEEE 1541
giga	G	10^9	SI
tebi	Ti	2^40	IEEE 1541
tera	T	10^12	SI

La función modulo devuelve el resto *i* de la división entera entre *k* y N. Esta función se suele denotar de la siguiente forma:
i = k mod N

Otra notación bastante utilizada es:
i = k % N

Realizando la división entera de 127 entre 100 se obtiene que el cociente es 1 y el resto es 27.

Luego *i* = 127 %100 = 27.

La función ceil o función techo aproxima el número *x* al número entero *y* mayor o igual más cercano.

Se denota de la siguiente forma:
y = ceil(x)

y = ceil(67,5) = 68.

La función floor o función suelo aproxima el número *x* al número entero *z* menor o igual más cercano.

Se denota de la siguiente forma:
z = floor(x)

z = floor(67,5) = 67.

Preguntas de autoevaluación tema 0

0.1. ¿Cuáles son los principales componentes hardware de un computador?

Independientemente del tipo de computador considerado, todos ellos están compuestos por uno o varios de los siguientes tipos de componentes físicos: procesador, memoria principal y controlador o adaptador de E/S.

0.2. Señalar cuál es la función del procesador y cuáles son sus principales componentes.

Se encarga de ejecutar secuencialmente las instrucciones de un programa y controlar el funcionamiento de los restantes componentes del computador.

Consta de los siguientes componentes:

- ***Unidad de control.*** Se encarga de buscar e interpretar las instrucciones y generar las señales de control necesarias a los restantes elementos del computador para su procesamiento.
- ***Unidad de procesamiento.*** Se encarga de la realización de operaciones aritméticas y lógicas sobre los datos.

Registros. Permiten al procesador almacenar de forma temporal información necesaria para su funcionamiento.

- ***Unidad de gestión de memoria*** (Memory Management Unit, MMU).

0.3. Señalar los registros existentes en un procesador.

(Respuesta en sección 0.2.1)

- ***Registro de instrucción.*** Contiene la instrucción actual que se tiene que ejecutar.
- ***Registro contador de programa***. Contiene la dirección de memoria donde se encuentra la próxima instrucción que hay que ejecutar.
- *Puntero de pila.* Dependiendo del tipo de procesador contiene la dirección de memoria de la próxima entrada libre (o de la última entrada ocupada) de la pila.
- ***Registro de estado del procesador.*** Contiene diferentes campos para almacenar información del estado del procesador en relación con el programa actualmente en ejecución, como por ejemplo: el modo de ejecución, el nivel de prioridad, el indicador de rebose, el indicador de arrastre, etc.
- ***Registros de propósito general.*** Para el almacenamiento de datos y resultados.

0.4. ¿Cuál es la función de la memoria principal de un computador?

Almacenar temporalmente los programas que se van a ejecutar.

¿Con qué tipos de memoria se implementa?

Memorias RAM. Su contenido se puede leer o escribir tantas veces como sea necesario. Su contenido es volátil, es decir, cuando se apaga el computador su contenido se borra.

0.5. ¿Qué es una palabra de memoria?

Es un conjunto ordenado de *n* bits contiguos que son manejados como un conjunto por la máquina.

0.6. Conocida la capacidad de la memoria principal expresada en palabras explicar cómo se determina el número de bits de que consta una dirección de memoria.

Numero de palabras x número de bits palabra.

0.7. ¿Qué es un controlador de E/S?

Módulo hardware que permite controlar a uno a varios dispositivos de E/S de un mismo tipo.

¿Puede un controlador de E/S controlar más de un periférico?

Si, si son del mismo tipo.

0.8. ¿Qué es un bus?

Es un medio de comunicación compartido entre dos o más dispositivos.

¿Por qué existen varios tipos de bus en los computadores modernos?

Para solucionar los problemas de rendimiento al ir apareciendo nuevos tipos de dispositivos de E/S con velocidades de transferencias más altas.

0.9. Señalar los diferentes tipos de memoria que conforman la jerarquía de memoria de un computador.

- ***Registros del procesador.*** Tiempo de acceso pequeño y capacidad pequeña.
- ***Memoria cache del procesador.*** Memoria RAM de acceso asociativo con tiempo de acceso más pequeño que la principal y capacidad también más pequeña.
- ***Memoria principal.*** Memoria RAM con mayor capacidad que la cache pero también mayor tiempo de acceso.

- ***Memoria secundaria.*** Formada por todos aquellos dispositivos de E/S conectados al computador dedicados al almacenamiento de programas y/o datos.

0.10. ¿Qué es el repertorio de instrucciones de un procesador?

Conjunto de instrucciones que es capaz de ejecutar

¿Puede un procesador ejecutar el repertorio de instrucciones de otro?

No, solo puede ejecutar instrucciones de su repertorio.

0.11. Enumerar los diferentes tipos de instrucciones presentes en el repertorio de instrucciones de un procesador.

- ***Instrucciones de transferencia de datos.*** Para copiar o mover datos de una posición de memoria o registro a otra posición de memoria o registro.
- ***Instrucciones de operaciones.*** Para realizar sobre operandos almacenados en memoria (o en registros) operaciones aritméticas, lógicas o de desplazamiento de bits.
- ***Instrucciones de transferencia de control.*** Para controlar la secuencia de la instrucciones. En esta categoría se encuentran las instrucciones de salto condicional e incondicional.
- ***Instrucciones privilegiadas de gobierno.*** Para modificar o comprobar el estado del procesador y los controladores de E/S. También para la gestión de la memoria.

0.12. Señalar la diferencia entre ejecución en modo supervisor y ejecución en modo usuario.

En *modo núcleo* el procesador puede ejecutar cualquier instrucción de su repertorio de instrucciones.
En *modo usuario* solo puede ejecutar un conjunto limitado de instrucciones del repertorio. Típicamente las instrucciones privilegiadas no se pueden ejecutar en modo usuario.

0.13. Explicar qué es el ciclo de ejecución de una instrucción y en qué fases se divide.

El *ciclo de instrucción* es la secuencia de operaciones realizadas por el procesador durante la ejecución de una instrucción.
Se divide en dos fases:

- ***Fase de búsqueda de la instrucción.*** El procesador busca en la memoria, de acuerdo con la dirección almacenada en el registro contador de programa, la instrucción que tiene que ejecutar. A menos que se indique lo contrario el procesador

incrementa el contador de programa después de cada fase de búsqueda para que quede apuntando a la próxima instrucción que hay que ejecutar. La instrucción leída en la memoria se almacena en el registro de instrucción.

- ***Fase de ejecución de la instrucción.*** El procesador decodifica el campo código de operación de la instrucción almacenada en el registro de instrucción, interpreta dicho campo de acuerdo con su repertorio de instrucciones y ejecuta las acciones adecuadas.

0.14. Explicar qué es un ensamblador y un compilador.

Un ensamblador es un programa que toma como entrada un fichero fuente escrito en lenguaje ensamblador y genera como salida un fichero objeto escrito en código máquina.

Un *compilador,* que es un programa que toma como entrada un fichero fuente escrito en un lenguaje de alto nivel y genera como salida un fichero objeto escrito en código máquina.

0.15. Describir las etapas de creación de un fichero ejecutable.

Con el código fuente mediante un compilador o ensamblador se genera un modulo objeto. Con ese módulo objeto, la librería y otros módulos objetos mediante el editor de enlace se crea el fichero ejecutable.

0.16. ¿Qué es una dirección relativa o lógica?

Son las direcciones asociadas por el compilador.

¿Y una dirección absoluta o física?

Son las direcciones de memoria principal.

0.17. Describir el funcionamiento de la E/S controlada por programa e indicar cuál es su principal inconveniente.

Cuando se ejecuta un programa de gestión de la E/S en el procesador que posee el control completo y directo de la operación de E/S. Este programa, antes de enviar o recibir un dato de un dispositivo de E/S, pregunta al controlador de E/S del dispositivo si éste se encuentra preparado. Dicha pregunta se implementa dentro de un bucle. El programa solo sale del bucle cuando el dispositivo se encuentra listo para enviar o recibir otro dato. A esta forma de implementar una espera por un determinado evento se le denomina *espera activa.*

El principal inconveniente es que desperdicia tiempo de uso del procesador debido a la espera activa del programa de gestión de E/S.

0.18. Explicar por qué la E/S controlada por interrupciones permite mejorar el rendimiento del computador.

El controlador de E/S avisará al procesador cuando el dispositivo de E/S estuviese preparado para enviar o recibir un dato. Mientras que el dispositivo de E/S no está listo, el procesador podría ejecutar otros programas, con lo que se mejoraría el rendimiento del computador.

0.19. Señalar las principales características de una interrupción hardware, una excepción y una trampa.

- ***Interrupciones hardware.*** Son producidas por módulos hardware externos al procesador, como por ejemplo los controladores de E/S. Estos dispositivos puede activar una interrupción para notificar al procesador que se ha completado una operación de E/ S, que se ha producido un cambio en el estado del periférico o que se ha producido algún error en el mismo.
- ***Excepciones.*** Son generadas por el hardware al intentar realizar una operación no permitida durante la ejecución de una instrucción de un programa, como por ejemplo, el acceso a una dirección de memoria ilegal, el rebose de una pila, la realización de una división por cero, etc.
- ***Trampas (traps).*** Se producen al ejecutar una determinada instrucción privilegiada del repertorio del procesador típicamente para solicitar los servicios del sistema operativo. Las trampas, a diferencia de las excepciones, son insertadas de manera voluntaria por el programador.

0.20. Describir el funcionamiento del DMA e indicar cuándo se emplea esta técnica de E/S. ¿Puede el DMA afectar al rendimiento del procesador?

El controlador de E/S puede transferir bloques de datos a la memoria principal (o viceversa) sin la intervención directa del procesador, excepto al inicio y al final de la transferencia. La implementación del DMA requiere de un módulo hardware adicional denominado *controlador de DMA* que se conecta al bus del sistema. Se emplea cuando la cantidad de datos a transferir entre la memoria principal y el dispositivo de E/S es grande.

Si puede afectar al rendimiento del procesador, recuérdese que solo un módulo puede usar el bus del sistema simultáneamente, por lo que si lo usa el controlador de DMA no lo podrá usar el procesador quedándose inactivo a la espera de usarlo.

0.21. ¿Qué es un canal o procesador de E/S?

Es un procesador auxiliar que se encarga de realizar todas las operaciones de E/S con un determinado conjunto de dispositivos de E/S.

0.22. Describir el proceso de arranque de un computador personal del tipo IBM PC compatible.

Cuando se enciende el computador el procesador comienza a ejecutar el BIOS, que en primer lugar comienza el *autotest de encendido* (Power On Self Test, POST).Si durante el POST se detecta algún problema en el hardware se envían una serie de sonidos por la bocina del computador para avisar al usuario.
Una vez realizado el POST, el BIOS busca el dispositivo de E/S desde el que se iniciará el sistema operativo. Para ello consulta en orden secuencial una lista de posibles dispositivos de arranque almacenados en la memoria CMOS de la placa base.
Una vez localizado el dispositivo de arranque, se accede a su primer sector (sector 0) que contiene el registro *de arranque maestro* (Master boot record, MBR), en el que se encuentra almacenado el programa denominado *cargador de arranque* (bootstrap loader) y la *tabla de particiones,* se carga en memoria la principal y se empieza a ejecutar. Accede al sector de arranque de la partición activa (primer sector de dicha partición) para cargar en la memoria principal un pequeño programa denominado *cargador de arranque secundario* que busca en el directorio raíz de dicha partición el código de inicio del sistema operativo lo carga en memoria y le transfiere el control para que se comience a ejecutar.
Cuando se inicia el sistema operativo éste obtiene del BIOS la configuración hardware del computador. A continuación comprueba que dispone de los drivers necesarios para cada dispositivo de E/S conectado, y los carga dentro del núcleo. A continuación inicializa las estructuras de datos del sistema operativo. Finalmente lanza un intérprete de comandos o una *interfaz de usuario gráfica* (Graphical User Interface, GUI) para que el usuario pueda interactuar con el sistema operativo.

Preguntas de autoevaluación tema 1

1.1. ¿Qué es un sistema operativo?

Un *sistema operativo* es una capa de software que gestiona de forma eficiente todos los dispositivos hardware de un computador

y además suministra a los usuarios una interfaz cómoda con el hardware.

¿Cuáles son sus principales objetivos?

- Gestionar o administrar eficientemente los dispositivos hardware de un computador.
- Ofrecer a los usuarios una interfaz cómoda con el hardware que les facilite el uso del computador.
-

1.2. **¿Qué función tiene el núcleo del sistema operativo?**

El *núcleo (kernel) del sistema operativo* que oculta el hardware a los programadores de los siguientes niveles, suministrando un conjunto de instrucciones denominadas *llamadas al sistema* con el que los programas ubicados en los niveles superiores pueden solicitar los servicios del núcleo.

1.3**. ¿Cuáles son los principales servicios de un sistema operativo?**

- ***Ejecución de programas.***
- ***Acceso a los dispositivos de E/.***
- ***Manipulación del sistema de archivos.***
- ***Comunicación y sincronización.***
- ***Detección y respuesta a errores.***
- ***Protección y seguridad.***
- ***Contabilidad***.

1.4. **¿En qué consiste el procesamiento por lotes o batch?**

Consiste en agrupar en lotes los trabajos con necesidades simi1ares, los cuales ahora se ejecutarían en el computador como un grupo.

1.5. ¿En qué consiste la técnica de multiprogramación?

Consiste en mantener en mantener en memoria principal varios trabajos simultáneamente. Si el trabajo que se está ejecutando actualmente en el procesador requiere la realización de una operación de E/S, mientras se completa dicha operación el procesador puede ejecutar otro trabajo. De esta forma se consigue aumentar el rendimiento del procesador.

1.6. **¿Cuáles son los principales criterios para clasificar los sistemas operativos?1.4**

- ***El número de usuarios que se pueden atender simultáneamente***. Pueden ser monousuario o multiusuario.

- ***El número de programas cargados en la memoria principal.*** Pueden ser monoprogramado y multiprogramado.
- ***Los requisitos temporales de los programas a ejecutar.***
- ***La finalidad del computador***.

1.7. ¿Qué es un sistema multiusuario?

Que puede atender simultáneamente a múltiples usuarios. Cada usuario trabaja con el sistema sin percibir que existen otros usuarios conectados.

1.8. ¿Qué es un sistema monoprogramado?

En la memoria principal del computador se almacena el sistema operativo y un único programa de usuario, que tiene a su disposición todos los recursos del computador. El procesador ejecuta dicho programa ininterrumpidamente desde su inicio hasta su finalización. Cuando finaliza dicho programa se carga otro programa en memoria que pasa a ser ejecutado.

1.9. ¿Qué define el grado de multiprogramación?

Al número de programas cargados en memoria principal.

1.10. ¿Cuáles son los requisitos de la multiprogramación?

El sistema operativo debe ser capaz de gestionar el uso de los recursos del computador entre todos los programas. Para ello debe disponer de algoritmos de planificación, mecanismos de sincronización y mecanismos de asignación y protección de la memoria principal y de la memoria secundaria. También es necesario que el hardware soporte mecanismos de protección de la memoria, E/S controlada por interrupciones y DMA.

1.11. ¿Cuáles son los beneficios de la multiprogramación?

Permite disminuir los tiempos de finalización de los programas y aumentar el uso de los recursos del sistema.

1.12. Enumerar y describir brevemente los tipos de sistemas operativos que se pueden distinguir en función de los requisitos temporales de los programas que se van a ejecutar.

- **Sistemas operativos de tiempo compartido o sistemas interactivos.** Cada usuario introduce desde su terminal una orden, bien mediante el uso del teclado o del ratón, y espera por la respuesta del sistema operativo. En todo momento, gracias a la multiprogramación, un usuario cree ser el único

que está interaccionando con el computador y tener a su disposición todos sus recursos.

- **Sistemas operativos de tiempo compartido o sistemas interactivos.** Cada usuario introduce desde su terminal una orden, bien mediante el uso del teclado o del ratón, y espera por la respuesta del sistema operativo.
- **Sistemas operativos de tiempo real.** Soportan *aplicaciones de tiempo real,* que son aquellas que reciben unas entradas procedentes de unos sensores externos, a través de unas tarjetas de adquisición de datos, y deben generar unas salidas en un tiempo de respuesta preestablecido.
- ***Sistemas operativos híbridos.*** Con capacidad para soportar tanto trabajos por lotes como aplicaciones interactivas o incluso aplicaciones suaves de tiempo real.

1.13. ¿Cómo se realiza la gestión de memoria principal y de archivos en los sistemas operativos por lote o batch?

La memoria se suele dividir en dos regiones: una ocupada permanentemente por el sistema operativo y la otra por programas transitorios.

La gestión de archivos se realiza de forma secuencial, lo que no requiere prácticamente de mecanismos de protección o de control de accesos concurrentes.

1.14. ¿Cuáles son los requisitos de un sistema operativo de tiempo compartido?

Debe disponer de un algoritmo de planificación que asegure un uso equitativo del procesador a todos los programas. También debe contar con mecanismos de protección de la memoria principal y de la memoria secundaria, así como con mecanismos de control de acceso concurrente a los archivos.

1.15. ¿Qué tipo de algoritmo de planificación emplean los sistemas operativos de tiempo real?

Utilizan un algoritmo de planificación basado en prioridades de tipo expropiativo, de tal forma que el proceso más prioritario siempre consigue el uso de los recursos que necesita.

1.16. Enumerar por orden las prioridades de ejecución de trabajos en un sistema operativo híbrido, y señalar algún ejemplo de sistema operativo de este tipo.

Normalmente se asigna a los trabajos por lotes una prioridad de ejecución más pequeña que a las aplicaciones interactivas, y a

éstas una prioridad de ejecución menor que a las aplicaciones suaves de tiempo real.
Ejemplos son UNIX y LINUX.

1.17. ¿Qué tipos de sistemas operativos se distinguen en función de la finalidad del computador?

- ***Sistemas operativos para macrocomputadores.***
- ***Sistemas operativos para servidores de red.***
- ***Sistemas operativos para computadores personales.***
- ***Sistemas operativos para computadores de mano.***
- ***Sistemas operativos integrados.***

1.18. Señalar las principales diferencias entre los sistemas operativos paralelo o multiprocesador, y los sistemas operativos distribuidos.

Los sistemas operativos paralelo o multiprocesador comparten reloj y en ocasiones la memoria, los sistemas operativos distribuidos no comparten ni reloj ni memoria.

Los sistemas operativos paralelo o multiprocesador utilizan una copia idéntica del sistema operativo, los sistemas operativos distribuidos en cada computador de la red ejecuta un sistema operativo.

1.19. ¿Qué son las llamadas al sistema?

Son las llamadas que hacen los programas en modo de ejecución usuario al sistema operativo para poder usar los recursos hardware.

1.20. ¿Cómo pueden invocar los programas de usuario a las llamadas al sistema?

- ***Mediante el uso de librerías de llamadas al sistema.***
- ***De forma directa***.

1.21. Enumerar la secuencia de eventos que se producen cuando un programa de usuario invoca a una llamada al sistema.

Se produce una *trampa* que provoca la conmutación hardware de modo usuario a modo supervisor y transfiere el control núcleo. El núcleo examina el identificador numérico de la llamada (cada llamada tiene asignado uno) y parámetros de la llamada para poder localizar y ejecutar a la rutina del núcleo asociada a la

llamada al sistema. Cuando dicha rutina finaliza, almacena el resultado en algún registro y provoca la conmutación hardware de modo supervisor a modo usuario para que el proceso de usuario que invocó la llamada continúe su ejecución.

1.22. ¿En qué categorías se pueden agrupar las llamadas al sistema atendiendo a su finalidad?

- ***Control de programas en ejecución (procesos).*** Se incluyen las llamadas al sistema para crear, ejecutar, suspender un tiempo prefijado, abortar y terminar procesos, para sincronizar la ejecución de un proceso con la aparición de un evento, las llamadas para obtener y establecer los atributos de un proceso, y las llamadas para asignar y liberar memoria.
- ***Administración de archivos o directorios***. Para crear, borrar, abrir, cerrar, leer, escribir archivos o directorios, para obtener y establecer los atributos de los archivos o directorios.
- ***Comunicaciones.*** Para crear o borrar una conexión de comunicación, para enviar o recibir mensajes o para conectarse a dispositivos remotos.
- ***Gestión de dispositivos.*** Para solicitar, liberar, leer o escribir un dispositivo, para obtener y establecer los atributos del dispositivo.
- ***Gestión de información del sistema.*** Para obtener y configurar datos del sistema, o para obtener o modificar la fecha y la hora.

1.23. ¿Cuáles son los subsistemas o componentes principales del núcleo de un sistema operativo?

- ***Subsistema de control de procesos.***
- ***Subsistema de administración de la memoria principal.***
- ***Subsistema de gestión de archivos.***
- ***Subsistema de E/S.***

1.24. ¿Qué factores definen la estructura del núcleo, y qué tipos se distinguen en función de éstos?

Queda definida por el número de módulos en que se descompone y cómo se interrelacionan.

Se distinguen principalmente los siguientes tipos de estructuras:

- ***Estructura monolítica o simple.***
- ***Estructura en módulos.***
- ***Estructura en capas.***

- ***Estructura extensible.***

1.25. ¿Cómo funciona un núcleo con estructura monolítica?

Todos los subsistemas y las estructuras de datos del núcleo están ubicados en un único módulo lógico. El software del núcleo está escrito como un conjunto de procedimientos. Hay un procedimiento principal, un conjunto de procedimientos de servicio y un conjunto de procedimientos auxiliares. Típicamente el procedimiento principal invoca al procedimiento de servicio requerido por una llamada al sistema. A su vez el procedimiento de servicio invoca a los procedimientos auxiliares que necesita. Un procedimiento es visible por todos los demás.

1.26. Comparar la estructura monolítica con la estructura en módulos.

En la monolítica existe un solo modulo en la de módulos existen varios.

En la monolítica no existe interface bien definida en la de módulos si tiene una interface bien definida en términos de entradas, salidas y funciones que realizan.

Un núcleo con estructura modular es mucho más fácil de mantener y modificar que un núcleo con estructura monolítica.

1.27. ¿Por qué se caracteriza la estructura en capas?

Se caracteriza porque el núcleo estar organizado en una jerarquía de capas, cada una de las cuales subyace sobre la anterior. Cada capa *i* se implementa como un objeto abstracto que encapsula una serie de estructuras de datos y la implementación de las operaciones que pueden manipularlas. Dichas operaciones pueden ser invocadas por las capas de mayor nivel *i* + 1, *i* + 2,… Asimismo la capa *i* puede invocar a las operaciones de las capas de los niveles inferiores *i* - 1, *i* - 2,...

¿Cuáles son sus principales ventajas e inconvenientes?

La principal ventaja de la estructura de capas, aparte de la sencillez de su diseño, es que simplifica el proceso de depuración y verificación del sistema operativo. Cuando se verifica y depura la capa *i,* se supone que el funcionamiento de la capa inferior *i* - 1 es correcto, ya que ha sido previamente depurada.

La mayor dificultad de la estructura en capas se encuentra en definir adecuadamente los servicios

De cada capa, ya que se debe tener en cuenta que una capa solo puede utilizar los servicios ubicados en capas inferiores. La estructura en capas suele ser menos eficiente que otras estructuras.

1.28. Para el caso de un núcleo con estructura extensible explicar qué es y de qué tareas se encarga:

a) El micronúcleo.

Se encarga de realizar únicamente los servicios absolutamente esenciales del sistema operativo, aquellos que dependen de la arquitectura de la máquina y que son independientes del tipo de sistema operativo, como por ejemplo, la gestión de memoria a bajo nivel, la comunicación entre procesos, gestión de la E/S y la gestión de las interrupciones.

b) Una extensión del núcleo.

Los servicios menos esenciales del sistema operativo, aquellos que son independientes de la arquitectura de la máquina y que dependen del tipo de sistema operativo, como por ejemplo la gestión de la memoria virtual, la administración de los sistemas de archivos o servicios de seguridad y protección, etc.

1.29. Señalar las principales ventajas y los inconvenientes de un núcleo con estructura extensible.

Ventajas:

- ***Manejabilidad.***
- ***Extensibilidad.***
- ***Fiabilidad.***
- ***Soporte simultáneo de múltiples sistemas operativos.***
- ***Portabilidad.***

La principal desventaja de un núcleo con estructura extensible es su menor rendimiento en comparación con otros tipos de estructuras. Entre las causas que contribuyen a este menor rendimiento destaca el uso del paso de mensajes como esquema de solicitud de servicios.

Preguntas de autoevaluación tema 2

2.1. ¿Qué es un proceso?

Es la entidad que se puede asignar y ejecutar en un procesador, es por tanto la unidad básica de trabajo de un sistema informático. En conclusión, un proceso es un programa en ejecución.

2.2. ¿Cuál es la principal diferencia entre un proceso ejecutándose en primer plano y otro ejecutándose en segundo plano?

Un proceso se ejecuta en *primer plano* (foreground) si los usuarios pueden interaccionar con el proceso durante su ejecución. Por el contrario se dice que un proceso se ejecuta en *segundo plano* (background) si un usuario no puede interaccionar con el proceso durante su ejecución.

2.3. ¿Qué regiones se diferencian en el espacio de direcciones de memoria lógica o virtual de un proceso?

Suelen distinguir al menos tres regiones: la región de código (también denominada región de texto), la región de datos y la región de pila. Adicionalmente pueden existir regiones de memoria compartida y regiones asociadas a librerías.

2.4. ¿Qué información contiene un marco de la región de pila del espacio de direcciones de memoria lógica de un proceso?

Los parámetros de la función, sus variables locales y las direcciones almacenadas en diferentes registros especiales de la máquina, como por ejemplo, el contador del programa y el puntero de la pila.

2.5. Describir brevemente los tres tipos de procesos que se pueden distinguir.

- ***Procesos de usuario.*** Son procesos asociados a la ejecución de programas invocados por los usuarios. Se ejecutan en modo usuario excepto cuando realizan llamadas al sistema que pasan a ser ejecutados en modo supervisor (realmente se ejecuta el sistema operativo pero en el nombre del proceso). Además se pueden ejecutar en primer plano o en segundo plano.
- ***Procesos demonio.*** Son procesos no asociados a ningún usuario que realizan tareas periódicas relacionadas con la administración del sistema, como por ejemplo, la administración y control de redes, y la administración de trabajos de impresión.
- ***Procesos del sistema operativo,* o *procesos del sistema.*** Son procesos que realizan tareas de administración del sistema operativo, como por ejemplo, el intercambio de procesos desde memoria principal a memoria secundaria. Se ejecutan normalmente en modo supervisor. Además se suelen ejecutar en segundo plano.

2.6. Describir brevemente cada uno de los principales estados en los que se puede encontrar un determinado proceso.

- ***Nuevo.*** El proceso acaba de ser creado pero todavía no se encuentra preparado para ser ejecutado, puesto que aunque se le han asignado algunas estructuras de datos aún no se encuentra cargado en la memoria principal.
- ***Preparado.*** El proceso está listo para ser ejecutado tan pronto como el planificador del sistema operativo lo considere oportuno.
- ***Ejecutándose.*** El proceso está siendo ejecutado en el procesador. En un computador con un único procesador solo puede existir en un determinado instante de tiempo un único proceso en este estado.
- ***Bloqueado.*** El proceso tiene que esperar hasta que se produzca un determinado evento, como por ejemplo, la finalización de una operación de E/ S.
- ***Terminado.*** El proceso ha finalizado su ejecución.

2.7. ¿Qué representa un diagrama de transición de estados?

Los posibles estados en que puede encontrarse un proceso en un determinado sistema operativo y las transiciones permitidas entre dichos estados. Los nodos del diagrama representa los estados y las líneas de conexión representan las posibles transiciones entre los estados.

2.8. Enumerar y describir las principales estructuras de control del sistema operativo.

- ***Tabla de procesos***. Es una estructura del sistema operativo que almacena información de control relevante sobre cada proceso existente.
- ***Tablas de memoria.*** Contienen información sobre el espacio asignado y el espacio libre de la memoria principal y del área de intercambio
- ***Tablas de E/S.*** Se utilizan para gestionar los dispositivos de E/S. Contienen información sobre un dispositivo está disponible o no y qué proceso lo está utilizando. También contienen información sobre el estado de una operación de E/S y el área de memoria principal usada como fuente o destino de una transferencia de E/S.
- ***Tablas de archivos.*** contienen información sobre los archivos abiertos por los procesos, su ubicación. en el almacenamiento secundario, su estado actual y otros atributos.

2.9. Enumerar las acciones que debe realizar el sistema operativo para crear un proceso.

- ***Comprobar si el proceso puede ser creado.*** El sistema operativo comprueba si existe suficiente espacio en memoria principal para crear otro proceso y si el usuario no ha excedido el número máximo de procesos que puede tener ejecutándose.
- ***Asignar una entrada de la tabla de procesos para el nuevo proceso.*** En este instante se le asigna un identificador numérico al proceso.
- ***Reservar espacio en memoria para el proceso.***
- ***Inicializar el bloque de control del proceso.***
- ***Establecer los enlaces apropiados.*** El sistema operativo debe configurar los punteros adecuados para enlazar la información del proceso en las diferentes listas, tablas y colas que mantiene.

2.10. Enumerar y describir las principales causas que originan la creación de un proceso.

- ***Arranque del sistema operativo.*** Al arrancar el sistema operativo se crean diversos procesos.
- ***Interacción del usuario con un intérprete de comandos o un entorno de ventanas.*** En sistemas operativos interactivos, cuando un usuario teclea una orden en un intérprete de comandos o hace clic con el ratón sobre un icono de una ventana se crea un nuevo proceso para atender la petición del usuario.
- ***Inicio de un trabajo por lotes.*** Cada vez que se inicia un trabajo por lotes se crea un proceso asociado a la ejecución de dicho trabajo.
- ***Un proceso en ejecución invoca a una llamada al sistema para crear otro proceso.*** Una determinada tarea puede plantearse para ser realizada en varias fases independientes, cada una de las cuales puede ser realizada por un proceso

2.11. ¿Cuáles son las principales causas por las que un proceso finaliza?

Un proceso termina cuando ha concluido de realizar la tarea para la que fue creado. También puede terminar un proceso debido a la aparición de una excepción durante su ejecución. Además, un proceso A puede terminar porque otro proceso B, con los permisos

adecuados, invoca una llamada al sistema para solicitar la terminación del proceso A.

2.12. ¿Qué es un cambio de contexto o proceso?

La interrupción de la ejecución de un proceso A para iniciar o continuar la ejecución de otro proceso B.

¿Cuáles son las principales causas que motivan un cambio de proceso?

- ***El proceso en ejecución pasa al estado bloqueado.*** Un proceso A pasa al estado bloqueado cuando debe esperar por la aparición de algún evento.
- **La terminación (voluntaria o forzada) del proceso en ejecución.**
- **El sistema operativo termina de atender una interrupción y existe un proceso B en estado preparado de mayor prioridad que el proceso actual.**
- **El proceso A en ejecución ha excedido el tiempo máximo de ejecución ininterrumpida.**

2.13. ¿Qué es el tiempo de conmutación y de qué factores depende?

El tiempo que se consume en el cambio de proceso.

Depende principalmente de factores asociados al hardware del computador como por ejemplo, la velocidad de lectura/escritura de la memoria principal, el número de registros del procesador cuyo contenido hay que salvar o cargar, la velocidad del procesador y la existencia de instrucciones especiales en el repertorio del procesador para salvar o cargar todos los registros.

2.14. ¿Qué se entiende por sobrecarga del sistema?

Tiempo que el procesador se encuentra ocupado ejecutando código del sistema operativo asociado a tareas y servicios de administración que no se pueden contabilizar a ningún proceso en particular.

2.15. Definir traza de ejecución.

Las instrucciones que ejecuta el proceso durante su tiempo de vida.

2.16. ¿Qué es un proceso multihilo?

Un proceso que está formado por dos o más hilos (una unidad elemental de asignación del procesador que sigue una determinada

traza de ejecución y que tiene asignado una pila de usuario, un espacio de almacenamiento y un bloque de control de hilo).

2.17. Enumerar las ventajas y los inconvenientes de los procesos multihilos.

Posibilita que puedan tener lugar, en el entorno del mismo proceso múltiples ejecuciones con un alto grado de independencia entre ellas.

- ***Aumento del rendimiento del sistema.***
- ***Ahorro de recursos.***
- ***Comunicación más eficiente***
- ***Mayor aprovechamiento de las arquitecturas multiprocesador.***
- ***Simplificación de la estructura de las aplicaciones.***

Inconveniente. La ejecución de múltiples hilos ejecutándose en paralelo en un proceso es análoga a la ejecución de múltiples procesos en paralelo.

2.18. Enumerar las ventajas y los inconvenientes de los hilos a nivel de usuario.

- ***Portabilidad.***
- ***Mejora del rendimiento del sistema.***
- ***Planificación independiente.***

La principal desventaja de los hilos de usuario se manifiesta en aquellos sistemas operativos que únicamente soportan un único hilo del núcleo. En estos sistemas cuando un hilo de usuario de un proceso entra en el estado bloqueado, entonces todo el proceso completo entra en el estado bloqueado.
Otra desventaja es que cuando un hilo se está ejecutando, no se puede planificar otro hilo de usuario del mismo proceso hasta que el primero no cede voluntariamente el uso del procesador.

2.19. Enumerar las ventajas y los inconvenientes de los hilos a nivel de núcleo.

Ventaja. Si uno se bloquea se puede planificar otro del mismo proceso o de otro proceso distinto. Además, en sistemas multiprocesador es posible ejecutar varios hilos del núcleo simultáneamente, cada uno de ellos en un procesador distinto.

Inconveniente. Su gestión contribuye a la sobrecarga del sistema. Para resolver este problema, muchos sistemas limitan el número de hilo del núcleo que se pueden crear, y además reciclan los hilos del núcleo ya existentes.

2.20. Describir las principales configuraciones en función del número y tipo de hilos soportados por un sistema operativo.

- ***Múltiples hilos de usuario sin soporte de hilos del núcleo.***
- ***Un hilo del núcleo por cada hilo de usuario.***
- ***Menor número de hilos del núcleo que hilos de usuarios.***

Preguntas de autoevaluación tema 3

3.1. Enumerar y describir brevemente los diferentes tipos de colas de procesos que mantiene un sistema operativo.

- ***Cola de procesos en el estado preparado.*** En ella se encuentran aquellos procesos que desean acceder al procesador para iniciar o continuar su ejecución. No tiene por qué ser única.
- ***Cola de procesos en el estado preparado en memoria secundaria.*** Contiene procesos que están a la espera de regresar a la memoria principal al estado preparado.
- ***Cola de procesos en el estado bloqueado.*** Los procesos que pertenecen a esta cola están a la espera de poder volver al estado preparado tan pronto ocurra el evento por el que han entrado en el estado bloqueado. No tiene por qué ser única.
- ***Cola de procesos en el estado bloqueado en memoria secundaria.*** Contiene procesos que están a la espera de pasar al estado preparado en memoria secundaria, si ocurre el evento por el que entraron en el estado bloqueado.
- **Cola de trabajos por lotes o cola de entrada.** Se almacena en memoria secundaria y contiene lo nuevos trabajos que llegan a un sistema operativo por lotes o a la parte por lotes de un sistema operativo híbrido. Cada trabajo se encuentra a la espera de que el sistema operativo cree un nuevo proceso para atenderlo.

3.2. ¿En qué consiste la actividad del sistema operativo conocida como planificación de procesos?

Un proceso durante su existencia puede pasar por varias colas. El sistema operativo debe determinar cuándo un proceso abandona una cola para acceder a un recurso o para ingresar en otra cola.

¿Qué niveles de planificación se distinguen?

- ***Planificación a corto plazo.*** Decide qué proceso en la cola de preparados será ejecutado a continuación en el procesador.

- ***Planificación a medio plazo***. Decide qué proceso en una cola de memoria principal es intercambiado a una cola de memoria secundaria, o viceversa.
- ***Planificación a largo plazo***. Decide qué trabajo de la cola de trabajo por lotes pasa a ser ejecutado en el sistema mediante la creación de un proceso.

3.3. ¿Qué función realiza el planificador a corto plazo?

La elección del proceso perteneciente a la cola de procesos en el estado preparado que pasará al estado ejecutándose.

¿Cuáles son sus componentes?

- **Encolador (enqueuer).** Cuando un proceso entra en el estado preparado, el encolador se encarga de incluir al proceso en la cola de procesos preparados mediante la configuración de los punteros necesarios en el bloque de control del proceso.
- **Conmutador de contexto (context switcher).** Se encarga de guardar el contexto del proceso que va a ser desalojado del procesador y cargar el contexto del proceso que ha sido planificado para ser ejecutado.
- **Distribuidor o despachador (dispatcher).** Se encarga de seleccionar un proceso de la cola de procesos preparados de acuerdo con un determinado algoritmo de planificación. También se encarga de cederle el control del procesador.

3.4. Enumerar algunas circunstancias que requieran la invocación del planificador a corto plazo.

- ***Cuando se ha terminado de crear un nuevo proceso hijo***. En este caso el planificador debe decidir si continúa ejecutando el proceso padre o comienza la ejecución del proceso hijo.
- ***Cuando un proceso entra en el estado bloqueado en espera de que se produzca algún evento***. En este caso dicho proceso ya no requiere utilizar el procesador por lo que el planificador debe decidir qué proceso ejecutar.
- ***Cuando se termina de atender una interrupción.*** Por ejemplo, si llega una interrupción para avisar que se ha completado una operación de E/S con el disco por la que estaba esperando en el estado bloqueado un cierto proceso.
- ***Cuando un proceso finaliza***. Si un proceso ha entrado en el estado terminado obviamente ya no utilizará el procesador por lo que el planificador debe decidir qué proceso será ejecutado a continuación.

3.5. ¿Qué tareas realiza el planificador a medio plazo?

Se encarga de decidir qué procesos en memoria principal de los pertenecientes a la cola de procesos en el estado preparado o la cola de procesos en el estado bloqueado serán intercambiados a memoria secundaria, pasando a formar parte de la cola de procesos preparados memoria secundaria o de la cola de procesos bloqueados en memoria secundaria, respectivamente. También se encarga de realizar la decisión contraria, es decir, qué procesos en las colas de memoria secundaria pasarán a las colas de memoria principal.

3.6. ¿Qué tareas realiza el planificador a largo plazo?

Se encarga de decidir qué trabajo de los existentes en la cola de trabajos por lotes será admitido en el sistema para ser ejecutado. Además, debe decidir cuándo se pueden admitir nuevos trabajos. La admisión de un trabajo consiste en la creación de un proceso para el procesamiento de dicho trabajo, el cual pasa a la cola de procesos en el estado preparado o la cola de procesos en el estado preparado en memoria secundaria.

3.7. ¿Qué es un proceso limitado por CPU?

Un proceso que emplea la mayor parte de su tiempo haciendo cálculos en el procesador.

¿Y un proceso limitado por E/S?

Un proceso que emplea la mayor parte de su tiempo esperando por la realización de operaciones de E/S.

3.8. ¿Cuales son los objetivos principales que debe cumplir la función de planificación de procesos?

- ***Equidad.*** En ausencia de un esquema de prioridades preestablecido, los procesos con requisitos similares deben obtener un uso de los recursos similares. Es decir, ningún proceso debe sufrir inanición.
- ***Previsibilidad.*** Trabajos con características similares deben ejecutarse en tiempos similares usando recursos similares.
- ***Equilibrado de los recursos.*** Se debe intentar mantener ocupados todos los recursos del computador. Para cumplir con este objetivo se debe procurar que los programas cargados en memoria sean una mezcla de programas limitados por CPU y programas limitados por E/ S.
- ***Proporcionalidad.*** Las peticiones que son percibidas por el usuario como sencillas deben emplear menos tiempo de

respuesta que las complejas ya que de lo contrario el usuario se irrita o considera que el sistema se ha colgado.

3.9. Definir los siguientes criterios de planificación del procesador: utilización del procesador o eficacia, productividad, tiempo de entrega o tiempo de estancia, tiempo de espera, tiempo de respuesta y plazo de finalización.

- ***Utilización del procesador o eficacia.*** Fijado un cierto tiempo de observación, hace referencia al tanto por ciento de dicho tiempo que el procesador ha estado activo.
- ***Productividad (throughput) o rendimiento***. Es el número de trabajos completados por unidad de tiempo (típicamente una hora). Esta medida depende obviamente de la longitud de los procesos.
- **Tiempo de entrega (turnaround time), también denominado tiempo de retorno o tiempo de estancia.** Es el tiempo medio que transcurre desde que se lanza un proceso hasta que finaliza. Se calcula como la suma del tiempo de ejecución y el tiempo de espera por los recursos, incluyendo el procesador.
- ***Tiempo de espera.*** Es la suma de los tiempos que un proceso pasa esperando en las diferentes colas del sistema por la obtención de recursos.
- ***Tiempo de respuesta* (response time).** Es el tiempo transcurrido desde que un usuario manda una orden desde su terminal hasta que comienza a recibir la respuesta. Obviamente se encuentra limitado por la velocidad de los dispositivos de E/S.
- ***Plazo de finalización* (deadline).** Hace referencia al tiempo máximo preestablecido que un proceso tiene para ser completado en un sistema de tiempo real.

3.10. Señalar los criterios principales considerados en la planificación del procesador dependiendo del tipo de sistema operativo.

- ***El planificador de un sistema por lotes debe intentar maximizar la utilización de procesador y la productividad. Además debe intentar minimizar el tiempo de entrega.***
- ***El planificador de un sistema de tiempo compartido debe intentar minimizar el tiempo de respuesta.***
- ***El planificador de un sistema de tiempo real debe intentar maximizar el número de plazos de finalización cumplidos.***

3.11. Definir planificación expropiativa y no expropiativa. Señalar sus principales diferencias y ventajas.

La planificación del procesador es *no expropiativa* (nonpreemptive) si permite que un proceso pueda estar ejecutándose en el procesador ininterrumpidamente hasta que termine o se bloquee en espera de un evento.

La planificación del procesador es *expropiativa* (preemptive) si el proceso que se está ejecutando en el procesador puede ser interrumpido en cualquier momento y pasado al estado preparado para proceder a ejecutar otro proceso distinto.

Una planificación expropiativa produce una mayor sobrecarga al sistema que una no expropiativa, que se ejecuta con mayor frecuencia el planificador y se realizan más cambios de proceso; pero proporciona un mejor servicio a la población de procesos ya que previene que un proceso monopolice del procesador.

3.12. ¿Qué es un cuanto de ejecución?

El tiempo máximo durante el cual un proceso se ejecuta ininterrumpidamente.

3.13. ¿Qué tipo de planificación, expropiativa o no expropiativa, es más recomendable para un sistema operativo por lotes?

La no expropiativa, o la expropiativa pero con un cuanto largo.

¿Y para un sistema de tiempo compartido?

La expropiativa con objeto de atender las peticiones de todos los usuarios.

3.14. Describir el algoritmo de planificación del primero en llegar primero en ser servido, (FCFS) (First-Come First-Served).

Este algoritmo cede el uso del procesador al primer proceso que lo solicite, éste puede utilizarlo ininterrumpidamente hasta que termine o se bloquee en espera de un evento.

3.15. Describir el algoritmo de planificación de primero el proceso más corto, SJF (Shortest Jop First).

Es un algoritmo de tipo no expropiativo que selecciona para ser ejecutado al proceso con tiempo de procesamiento más corto. Si dos o más procesos tienen el mismo tiempo de procesamiento entonces se les aplica el algoritmo FCFS.

3.16. Describir el algoritmo de planificación del proceso con menor tiempo restante, SRT (Shortest Remaining Time Next).

Siempre selecciona para ser ejecutado al proceso con menor tiempo restante de ejecución. Si mientras se está ejecutando en el

procesador un proceso A llega a la cola de preparados un proceso B con menor tiempo restante de ejecución, el planificador expropia el procesador al proceso A y se lo cede al proceso B.

3.17. Describir el algoritmo de planificación de turno rotatorio *(round robin)*.

Asigna el uso ininterrumpido del procesador a o durante una cantidad fija de tiempo denominada *cuanto* (quantum) o *rodaja de tiempo* (time slice). Finalizado el cuanto, se genera una interrupción de reloj que produce que la ejecución del proceso sea interrumpida y se pase a ejecutar el primer proceso de la cola de preparados, como en la planificación. El proceso interrumpido se coloca al final de dicha cola y tendrá que esperar turno volver a utilizar el procesador.

Si un proceso termina de usar el procesador antes de consumir su cuanto, entonces se planifica el primer proceso de la cola preparados sin esperar a que finalice el cuanto. Por otra parte, si finaliza un cuanto y no existen procesos en la cola de preparados, entonces el proceso que se estaba ejecutando puede seguir usando el procesador.

3.18. Describir el algoritmo de planificación basada en prioridades.

Asigna a cada proceso en el momento de su creación una determinada *prioridad de ejecución* en la forma de un número entero positivo. Dependiendo del sistema, la máxima prioridad puede corresponder a un número pequeño, típicamente 0, o a un número grande. Siempre se planifica para ejecución al proceso de mayor prioridad. En caso de que varios procesos posean la misma prioridad éstos se deben planificar mediante otro algoritmo distinto, como por ejemplo, FCFS o SJF.

3.19. Describir el algoritmo de planificación basada en múltiples colas de prioridad.

Implementa varias colas de procesos preparados o *colas de prioridad*, cada una de las cuales solo pueda contener procesos con una determinada prioridad o dentro rango de prioridades.

Siempre se ejecuta un proceso perteneciente a la cola de preparados con mayor prioridad. Sólo cuando dicha cola está vacía se puede planificar un proceso perteneciente a una cola de preparados de menor prioridad. Un proceso a lo largo de su vida solo puede pertenecer a una determinada cola de preparados.

Además para planificar cada cola se puede utilizar un algoritmo distinto: FCFS, SJF, rotatorio, etc.

Puede producir la inanición de los procesos pertenecientes a las colas de menor prioridad si las colas de mayor prioridad nunca se quedan vacías. Para evitar este problema se puede asigna procesos de cada cola un determinado porcentaje de tiempo de uso del procesador.

3.20. Describir el algoritmo de planificación basada en múltiples colas de prioridad y realimentación.

Un proceso solo puede pertenecer a una determinada cola de preparados o cola de prioridad durante su tiempo de vida. En consecuencia, posee una prioridad estática. Esta restricción simplifica la implementación del algoritmo pero también lo hace poco flexible. Si se levanta esta restricción y se permite que un proceso pueda ir cambiando de cola de prioridad durante su existencia, es decir, que su prioridad pueda ser modificada dinámicamente, se dice que el algoritmo posee *realimentación.*

3.21. Describir el algoritmo de planificación por tiempo límite.

En los sistemas de tiempo real estrictos el tiempo de servicio de algunos procesos deben completarse en un determinado *plazo* (deadline) o *tiempo límite.*

Para lograr este objetivo el planificador solo admite un proceso en la cola de procesos preparados si es capaz de garantizarle que su tiempo de servicio se atenderá dentro de su tiempo límite, teniendo en cuenta los tiempos de servicio y plazos de los procesos ya existentes en dicha cola. En consecuencia el planificador de estos sistemas debe conocer por adelantado para cada proceso su tiempo de servicio y su tiempo límite de ejecución.

3.22. Describir la planificación de hilos en función del tipo de hilo soportado por el sistema operativo.

Si ente se soportan hilos a nivel de usuario, el sistema operativo no es consciente de la existencia de estos hilos y por ello realiza una planificación global a nivel de proceso de acuerdo con un determinado algoritmo de planificación.

Una vez que un proceso está siendo ejecutado en modo usuario es el hilo de planificación dentro de dicho proceso el que decide qué hilo va a ser ejecutado. Por lo tanto cada proceso puede planificar sus hilos con el algoritmo de planificación que considere más oportuno.

Puesto que en modo usuario no se pueden tratar las interrupciones, no existen interrupciones de reloj disponibles para establecer el cuanto de un hilo. Éste se ejecuta hasta que voluntariamente decide ceder el uso del procesador o expira el cuanto del proceso. En dicho caso el sistema operativo debe seleccionar otro proceso para ser ejecutado.

Si se soportan hilos del núcleo, el sistema operativo planifica estos hilos usando algún determinado algoritmo de planificación. El planificador puede tener en cuenta consideraciones relativas al rendimiento del sistema a la hora de planificar hilos del núcleo de igual prioridad.

En sistemas con una configuración del tipo menor número de hilos del núcleo que hilos de usuarios, aparte de las dos planificaciones comentadas, es necesario implementar con la biblioteca de planificación local para establecer qué hilo de usuario puede usar un hilo del núcleo.

Preguntas de autoevaluación tema 4

4.1. Explicar cuándo se produce y en qué consiste el problema de la condición de carrera.

Se producen cuando varios procesos acceden a un recurso compartido, con lo cual en vez de cooperar compiten por el ya que el resultado final de la ejecución depende del orden en que se hayan planificado los procesos, es en que se hayan ejecutado las instrucciones .de lectura y escritura de cada proceso sobre el recurso.

Surge cuando múltiples procesos independientes se ejecutan concurrentemente y acceden para leer o escribir en un recurso compartido.

4.2. ¿Qué es una sección crítica?

Una instrucción o conjunto de instrucciones secuenciales del código de un proceso que requieren manipular un recurso compartido (variable, dato, fichero, …) con otros procesos. Dentro del código de un proceso pueden existir varias secciones críticas.

4.3. ¿En qué consiste la exclusión mutua?

Que el uso de un recurso por parte de un proceso excluya su uso para los restantes.

¿Qué problema permite resolver?

Las condiciones de carrera.

4.4. Explicar brevemente los problemas que puede producir el uso de la exclusión mutua si no se implementa correctamente.

Puede producir interbloqueo y la inanición de procesos.

El interbloqueo o *bloqueo mutuo* de dos o más procesos, se produce por ejemplo cuando dos procesos A y B necesitan dos recursos RI y R2 para realizar una cierta función. Si A posee RI y B posee R2 o puede progresar, ambos se quedan esperando por el recurso que necesitan, y no pueden liberar el recurso que poseen hasta obtener el otro recurso y realizar la función correspondiente.

La *inanición de un proceso* se produce cuando un proceso no puede progresar al necesitar un recurso al que nunca llega a acceder porque el sistema operativo siempre le relega dando prioridad a otros procesos.

4.5. ¿Qué requisitos debe cumplir el mecanismo o técnica que se utilice para implementar la exclusión mutua?

- ***Garantizar que solo un proceso puede estar a la vez dentro de una sección crítica asociada a un recurso R.*** Sólo se puede ejecutar a la vez una única sección crítica asociada a un cierto recurso compartido R.
- ***No puede realizar suposiciones sobre la velocidad relativa y prioridad de ejecución de los procesos concurrentes.***
- ***Garantizar que un proceso que no está ejecutando una sección crítica asociada a un recurso R no impide o bloquea el acceso a R a otros procesos.***
- ***Asegurar que ningún proceso se quede esperando indefinidamente a poder acceder a una sección crítica asociada a un recurso R.*** Es decir, debe evitar la inanición de procesos. Obviamente esto requiere que un proceso no debe permanecer dentro de su sección crítica por un tiempo ilimitado.
- ***Si ningún proceso está dentro de una sección crítica asociada a un recurso R, entonces si un proceso desea entrar en una sección crítica se le debe conceder rápidamente.***

4.6. ¿Por qué se caracterizan las dos soluciones software a las exclusiones mutuas descritas en el texto?

Porque no presuponen el apoyo de sistema operativo o del hardware del computador para su implementación.

¿Cuál es su principal inconveniente?

Que cualquier proceso que desea entrar en una sección crítica, si está siendo ejecutada por otro proceso, debe esperar a

que éste termine mediante la ejecución de un bucle que comprueba el valor de una condición basada en el valor de una o varias variables. Esta variable hace las veces de *cerrojo* (lock).

4.7. ¿Cómo se puede lograr la exclusión mutua usando un cerrojo y alternancia estricta?

En esta solución (código pag.125) se considera la existencia de una variable global de acceso *turno*, que se usa como cerrojo y que puede tomar como valor el identificador numérico de un proceso. En este caso el proceso A tiene el identificador 0 y el proceso B el identificador 1. Nótese que el valor al que se inicializa el cerrojo determina qué proceso puede entrar en primer lugar en su sección crítica, en este caso es el proceso A.
Cuando un proceso (A o B) quiere entrar en su sección crítica comprueba el valor de *turno*. Si coincide con su identificador numérico entonces puede ejecutar su sección crítica, en caso contrario deberá esperar. La espera que realiza el proceso es una espera activa ya que entra en un bucle para comprobar el valor de *turno*, del que no sale hasta que *turno* toma como valor el identificador numérico del proceso que está esperando.
Una vez que turno toma el valor del identificador numérico del proceso que desea entrar en su sección crítica, éste podrá ejecutar dicha sección. Cuando finalice debe configurar el valor de *turno* con el identificador numérico del otro proceso concurrente. En consecuencia, el acceso a la sección crítica se va alternando entre los procesos A y B. Hasta que un proceso no finaliza la ejecución de su sección crítica, el otro proceso no puede ejecutar la suya.

¿Qué problemas presenta esta solución software?

La velocidad de ejecución de los procesos A y B viene limitada por la velocidad de ejecución del proceso más lento.

Si un proceso falla (por ejemplo A) antes de poder cambiar el valor de turno al identificador del otro proceso (B), entonces éste (B) permanecerá bloqueado indefinidamente.

4.8. ¿Qué acciones se realizan en la función *acceso_sc* del algoritmo de Peterson?

En primer lugar se calcula el valor de *otro_pid*, es decir, el identificador asociado al otro proceso concurrente que se está ejecutando, en este caso el proceso B. Para ello realiza la operación *otro_pid== l-pid*, luego en este caso *otro_pid== 1*
A continuación el proceso A expresa su deseo de acceder al recurso compartido poniendo el valor TRUE en el elemento 0 del

vector de dos elementos enteros *petición_rec*. Luego asigna a la variable *turno* un valor igual al identificador numérico del otro proceso, en este caso 1. Finalmente entra en un bucle while en espera de poder usar el recurso compartido. Dicho bucle se ejecutará mientras se cumplan las siguientes dos condiciones:
1. *turno==otro pid*. La variable *turno* coincida con el identificador del otro proceso que se está ejecutando concurrentemente. En este caso el proceso es B luego turno vale 1.
2. *petición_rec [otro_pid] ==TRUE.* El otro proceso (en este caso B) haya solicitado previamente usar el recurso compartido.

¿Y en la función *salida_sc*?

Lo único que hace es desactivar la petición de uso del recurso por parte del proceso invocador; para lograrlo configura con el valor FALSE el elemento asociado al proceso invocador en el vector *petición _ rec*.

4.9. ¿Por qué el algoritmo de Peterson evita la posibilidad de interbloqueo de procesos?

Por que el proceso que primero ejecuta la instrucción *peticion_rec [id_proceso] =TRUE* se garantiza el uso del recurso. El otro proceso tendrá que esperar.

Nunca puede suceder que los dos procesos se queden bloqueados en espera de entrar en su sección crítica, ya que si A está bloqueado a la fuerza turno=l y en consecuencia el proceso B podrá entrar en su sección crítica.

4.10. Cuando se utiliza el algoritmo de Peterson ¿puede un proceso monopolizar el uso de un recurso?

No puede monopolizar el uso de un recurso ejecutando repetidas veces sección crítica ya que siempre concede una oportunidad de entrar al otro proceso al ejecutar la instrucción *turno=otro_pid.*

4.11. ¿Cómo se garantiza la exclusión mutua mediante el uso de instrucciones máquina especiales?

Las instrucciones especiales garantizan que si un procesador PI está accediendo a una posición de memoria ningún otro procesador acceder a dicha posición hasta que PI haya terminado de usar dicha posición de memoria.
Estas instrucciones máquina especiales realizan acciones tales como leer y comprobar, o leer y escribir una posición de memoria determinada. Se caracterizan por ser *atómicas,* es decir, se ejecutan en un ciclo de instrucción que no puede ser interrumpido,

por lo que su ejecución siempre se completa. De esta forma se garantiza el acceso exclusivo a la posición de memoria accedida.

¿Cuáles son sus principales inconvenientes?

- ***La espera que realiza un proceso para entrar en su sección crítica activa***. La espera activa supone un problema si se dilata en exceso en el tiempo.
- ***Puede producir inanición de procesos.***
- ***Puede presentar esta solución es la aparición de interbloqueos.***

4.12. ¿Cómo se garantiza la exclusión mutua mediante el uso del bloqueo de las interrupciones?

Al permitir que un proceso A, antes de entrar región crítica, ejecute una instrucción especial para bloquear el mecanismo de interrupciones del sistema, y cuando salga de la región crítica ejecute otra instrucción especial que lo vuelva a habilitar.

¿Cuáles son sus principales inconvenientes? (Respuesta en sección 4.2.6)

El rendimiento del sistema se puede degradar bastante al no permitirle atender las interrupciones más prioritarias en el momento en que llegan. Además se deja en manos del proceso la decisión de ceder el procesador y si éste nunca lo devuelve el sistema se quedará' colgado.

Esta solución no sirve en el caso de sistemas multiprocesadores, ya que un proceso B, que se estuviera ejecutando en un procesador P2 distinto al procesador P1 en el que ejecuta al proceso A podría acceder a una sección critica aunque estén deshabilitadas las interrupciones, ya que esto solo garantiza que el proceso A no puede ser expulsado del procesador P1, pero no que se ejecuten otros procesos en otros procesadores.

4.13. ¿Qué son los semáforos?

Son un mecanismo de sincronización de procesos concurrentes gestionado por los sistemas operativos.

4.14. Describir el funcionamiento de las operaciones *init_sem, wait _sem, signal_sem* cuando se aplican sobre un semáforo general y cuando se aplican sobre un semáforo binario.

Semáforo general.

- ***init_sem.*** Inicializa el semáforo S con el estado N.
- ***wait_sem.***

  Disminuye en una unidad el valor del semáforo. Es decir, S = S -1. Si el valor resultante es un número negativo entonces el

proceso que ha invocado esta operación es añadido a la cola de procesos bloqueados asociada al semáforo y se bloquea.

- ***signal_sem*.**
  Incrementa en una unidad el valor del semáforo, es decir, S = S + 1. Si el valor resultante es menor o igual a O entonces se elimina de la cola asociada al semáforo uno de los procesos bloqueados, y se le despierta, lo que hace que pase al estado preparado para ejecución

Semáforo binario.

- ***init_sem.*** Inicializa el semáforo S con el estado N.
- ***wait_sem (S).*** Esta operación comprueba el valor del semáforo S. Si S=0, entonces el proceso es colocado en la cola de procesos bloqueados asociada al semáforo y se bloquea. Si S= 1, entonces pone el semáforo a O y el proceso puede continuar su ejecución.
- ***signal_ sem (S).*** Esta operación comprueba si la cola de procesos bloqueados asociada al semáforo S está vacía. En caso afirmativo, pone el semáforo a 1, y continúa su ejecución. En caso negativo, es decir, hay procesos bloqueados en el semáforo, entonces el sistema operativo elimina de la cola asociada al semáforo a uno de los procesos bloqueados, y le despierta lo que hace que pase al estado preparado para ejecución.

4.15. ¿Cómo implementa el sistema operativo las operaciones sobre un semáforo?

El sistema operativo se encarga de asignar las estructuras de dato necesarias para los semáforos y de realizar las operaciones sobre ellos.
El núcleo implementa las operaciones wait_sem y signal_sem como primitivas o funciones elementales de su código que se ejecutan de forma atómica, es decir, se ejecutan en un único ciclo de instrucción que no puede ser interrumpido, por lo que su ejecución siempre se completa.
El núcleo tiene que garantizar que dos procesos no pueden realizar simultáneamente dos operaciones sobre un mismo semáforo.
Luego en la implementación de las primitivas del núcleo de las operaciones wait_sem y signal_sem se ha de resolver un problema de exclusión mutua.
En sistemas con un único procesador la exclusión mutua se garantiza bloqueando las interrupciones cuando se está ejecutando una operación sobre el semáforo.

4.16. Describir cómo se usan los semáforos para obtener el acceso con exclusión mutua a un recurso.

Hay utilizar tantos semáforos como recursos distintos compartan, a través de regiones críticas, los procesos recurrentes. Supóngase, por simplificar, que únicamente hay un recurso cuyo acceso se va a regular con un semáforo S. El semáforo se inicializa al, init_sem (S,1). Cuando un proceso desea entrar en una región crítica asociada a dicho recurso realiza una operación wait_sem (S). Si el valor de S es 1, entonces S se hace 0 y el proceso puede acceder inmediatamente a la sección crítica. A partir de momento cualquier proceso que desee entrar en su sección crítica e invoque una operación wait_sem (S) producirá que se decremente en una unidad el semáforo y se quede bloqueado en la cola de dicho semáforo.
Cuando el proceso que estaba dentro de su sección finalice entonces ejecutará signa1_sem (S) que incrementará en una unidad el valor del semáforo y desbloqueará a uno de los procesos bloqueado en la cola del semáforo.

4.17. Describir cómo se usan los semáforos para sincronizar procesos.

En el caso de dos procesos A y B basta con usar un semáforo S que se inicializa a 0, init_sem (S,0). Si el proceso A invoca una operación wait_sem (S), S toma el valor -1 (o 0 si era un semáforo binario) y se bloquea en la cola del semáforo en espera de que el proceso B complete alguna operación. Cuando el proceso B finaliza dicha operación ejecuta una operación signa1_sem (S) para notificárselo proceso A. Como resultado de dicha operación el semáforo S toma el valor 0, el proceso A se desbloquea y pasa al estado preparado para ejecución.

4.18. ¿Qué problemas potenciales se pueden producir debido a un mal uso de los semáforos?

La colocación inadecuada de las operaciones wait_sem y signal_ sem pueden producir errores de temporización y de funcionamiento (condición de carrera, inanición o interbloqueo) que pueden ser difíciles de detectar, ya que solo se producen bajo determinadas secuencias de ejecución, las cuales no se dan siempre.

4.19. ¿Qué es un monitor?

Es un módulo software, que consta de un conjunto de procedimientos, variables y estructuras de datos, definido en algunos lenguajes de alto nivel que posee la propiedad especial de que solo permite a un único proceso simultáneamente ejecutar alguno de sus procedimientos. Además, las variables contenidas dentro del monitor solo son accesibles por los procedimientos del monitor y no por procedimientos externos.

4.20. ¿Qué utilidad tienen las variables de condición de un monitor?

Cada variable de condición tiene asociada una cola de procesos bloqueados en espera de que se cumpla dicha condición.

¿Tienen asociada algún tipo cola?

Si, una cola de procesos asociada a la condición X.

4.21. Describir el funcionamiento de las operaciones *wait_mon y signal_mon* de un monitor.

wait_mon(X). El proceso dentro del monitor que realiza esta operación queda suspendido en una cola de procesos asociada al cumplimiento de la condición X. En consecuencia otro procedimiento puede entrar en el monitor.

signal_mon (X). Comprueba si la cola de procesos asociada a la condición X contiene algún proceso bloqueado. En caso afirmativo se desbloquea a un proceso. Si la cola estaba vacía esta operación no tiene ningún efecto. Si no existe ningún proceso en la cola de condición la señal de aviso se pierde.

4.22. Explicar qué sucede con un proceso que invoca una operación *signal_mon* según la solución de:

a) Hoare.

El proceso invocador A de la operación signa1_mon se bloquea y se permite ejecutar en monitor al proceso B desbloqueado por dicha operación. Cuando B finalice un procedimiento o se bloquee en una condición entonces el proceso A se desbloqueará.

b) Hansen.

El proceso invocador de la operación signa1_mon sale del monitor inmediatamente. Luego esta operación aparecería como la sentencia final del procedimiento de un monitor.

c) Lampson y Redell.

El proceso A invocador de la operación signa1_mon continua su ejecución hasta finalizar el procedimiento del monitor o

bloquearse en una condición. Luego se retomará la ejecución del proceso B desbloqueado por signa1_mon.

4.23. Describir cómo se obtiene en un monitor el acceso con exclusión mutua a un recurso.

Un monitor garantiza implícitamente la exclusión mutua sobre recursos compartidos por varios procesos concurrentes. Simplemente hay que definir dentro del monitor los recursos compartidos.

4.24. Describir cómo se obtiene en un monitor la sincronización de procesos.

Mediante el uso de las variables de condición y de las operaciones wait_mon y signa1_mon. Por lo tanto si un monitor está bien definido, la sincronización de los procesos está garantizada.

4.25. ¿Qué es el paso de mensajes?

Es un mecanismo de sincronización y comunicación entre procesos soportado por los sistemas operativos tanto de sistemas centralizados como distribuidos.

4.26. ¿Qué es un mensaje?

Es un conjunto de información que puede ser intercambiada entre un proceso emisor y un proceso receptor.

¿Cuáles son las dos operaciones básicas que se pueden realizar sobre un mensaje?

send(destino, mensaje)
receive(fuente, mensaje).

4.27. ¿Cuáles son los aspectos básicos en el diseño del mecanismo de paso de mensajes?

La especificación de la fuente y el destino del mensaje.
El esquema de sincronización o, el formato del mensaje.
El medio de almacenamiento de los mensajes.

4.28. Explicar cómo se realiza la comunicación directa mediante paso de mensajes.

El proceso emisor del mensaje especifica explícitamente en la operación send el proceso al que va dirigido el mensaje, send(P_receptor, mensaje). Por su parte, el proceso receptor del mensaje especifica explícitamente en la operación receive el proceso emisor del mensaje, receive(P_emisor, mensaje).

Otra posibilidad, es que el proceso receptor del mensaje especifique implícitamente en la operación receive el proceso emisor del mensaje, receive(ident, mensaje). Donde ident toma el valor del identificador del proceso con el que se ha realizado la comunicación, el cual no se sabe a priori, sino una vez que se completa la operación receive.

4.29. Explicar cómo se realiza la comunicación indirecta mediante paso de mensajes.

El proceso emisor envía el mensaje a una estructura de datos compartida denominada *buzón* que se implementa como una *cola de mensajes.* Por su parte, el proceso receptor solicita recibir el mensaje de dicho buzón. Además previamente a las operaciones de emisión o recepción un proceso (emisor o receptor) tiene que solicitar mediante una llamada al sistema la creación del buzón.

¿Qué esquemas de difusión de mensajes posibilita? (Respuesta en sección 4.6.2)

- **Un emisor - un receptor**. Un proceso emisor envía un mensaje a un buzón para que lo recoja determinado proceso receptor.
- **Un emisor - varios receptores**. Un emisor envía un mensaje a un buzón para que lo puedan leer varios receptores.
- **Varios emisores - un receptor.** Varios procesos emisores envían mensajes a un buzón, que en este caso se suele denominar *puerto,* para que los lea un único proceso receptor.
- **Varios emisores - varios receptores.** Varios procesos emisores envían mensajes a un buzón para que sean leídos por varios procesos receptores.

4.30. Describir el funcionamiento de un buzón que sea propiedad de:

a) Un proceso.

Entonces se implementa en el espacio de direcciones de dicho proceso. En este caso solo el proceso propietario puede recibir mensajes de dicho buzón. Cuando el proceso propietario finaliza entonces el buzón desaparece. Si algún proceso posteriormente envía mensajes a este buzón debe ser notificado de la no existencia del mismo.

b) El sistema operativo.

Entonces se implementa en el espacio de direcciones del núcleo, por lo que es independiente de cualquier proceso. En este

caso el sistema operativo debe proporcionar llamadas al sistema para crear y borrar el buzón. El proceso que crea un buzón es considerado su propietario. Inicialmente solo el propietario puede leer mensajes del buzón pero dicho privilegio se puede compartir con otros procesos mediante las llamadas al sistema adecuadas. En consecuencia pueden existir varios receptores.

4.31. Enumerar los cuatro esquemas de sincronización posibles en el mecanismo de paso de mensajes.

Operación send con bloqueo. El proceso emisor pasa al estado bloqueado hasta que el mensaje sea recibido por el proceso receptor o por el buzón.
Operación send sin bloqueo. El proceso emisor envía el mensaje a un proceso receptor o a un buzón y continúa su ejecución. Un proceso, por error o intencionadamente, puede estar enviando continuamente mensajes, lo que repercute en el rendimiento del sistema ya que se están consumiendo tiempo de procesador y espacio de almacenamiento.
Operación receive con bloqueo. El proceso receptor pasa al estado bloqueado en espera de recibir un mensaje de un proceso emisor o de que exista algún mensaje en el buzón. El receptor puede quedarse indefinidamente bloqueado si el proceso emisor finaliza antes de enviar el mensaje o si el mensaje se pierde en el caso de un sistema distribuido
Operación receive sin bloqueo. El proceso receptor obtiene de un proceso emisor o de un buzón un mensaje válido o uno nulo y continua su ejecución. Su principal desventaja es que si se ejecuta antes que llegue el mensaje, éste puede perderse.

4.32. Describir las partes de un mensaje.

La *cabecera del mensaje* contiene información de control del mensaje como por ejemplo: el tipo y la prioridad del mensaje, identificadores del destino y origen del mensaje, la longitud o tamaño del mensaje, punteros a otros mensajes de la cola de mensajes, etc.
El *cuerpo del mensaje* contiene el contenido propiamente dicho del mensaje.

4.33. Describir las principales ventajas e inconvenientes relativas a que la longitud de un mensaje sea fija o variable.

Los *mensajes de longitud fija* son más fáciles de implementar para un sistema operativo, ya que se almacenan en bufferes de tamaño fijo, lo que hace más sencilla y eficaz su asignación. Su

principal desventaja es que complican a los programadores la creación de los procesos emisores ya que los mensajes largos deben ser divididos y enviados en varios mensajes del tamaño de mensaje soportado, lo que puede producir problemas de secuenciamiento con los procesos receptores.

Los *mensajes de longitud variable* facilitan al programador la programación de los procesos emisores. Sin embargo, su implementación a nivel del sistema operativo es más compleja, ya que ahora tiene que asignar trozos de memoria de tamaño variable, según el tamaño de cada mensaje, lo cual aumenta la sobrecarga y puede producir problemas de fragmentación de memoria.

4.34. Describir los mecanismos de mensajes posibles en función de la capacidad de las colas de mensajes.

- ***Mecanismo de mensajes sin buffer.*** Las colas de mensajes son de capacidad nula, por lo que lo mensajes nunca pueden esperar. El proceso emisor se bloquea hasta que el receptor recibe el mensaje.
- ***Mecanismo de mensajes con buffer.*** Las colas de mensajes permiten almacenar temporalmente un número finito de mensajes. Cada vez que llega un mensaje se añade a la cola y cada vez que un proceso lee un mensaje se borra (salvo que se especifique lo contrario). Si la cola está llena, el proceso emisor se bloquea hasta que exista espacio disponible en la cola.

4.35. Supuesto que se dispone de la operación *send* sin bloqueo, de la operación *receive* con bloqueo y que la comunicación es indirecta a través de un buzón. Describir cómo se puede obtener mediante paso de mensajes:

a) La sincronización de procesos.

El mensaje que se va enviar no requiere tener ningún contenido.

Para conseguir la sincronización el proceso A debe ejecutar una operación receive cuando debe esperar por una acción que debe realizar el proceso B. Por su parte, el proceso B debe ejecutar una operación send cuando termine de ejecutar dicha acción.
Si se ejecuta antes la operación receive del proceso A que la operación send del proceso B, entones el proceso A se quedará bloqueado a la espera de la recepción del mensaje. En caso contrario el proceso A continuará su ejecución, eso significa que el proceso B ya ha realizado la acción por la que iba esperar el proceso A, y en consecuencia no tiene necesidad de esperar.

b) El acceso con exclusión mutua a un recurso.

La cola de mensajes se inicializa fuera del código de los procesos con un mensaje que no requiere tener ningún contenido y que se denotará como nulo.

Un proceso que desea ejecutar su sección crítica primero debe ejecutar una operación receive para recibir un mensaje. Si el buzón está vacío entonces se bloquea hasta que llegue un mensaje a la cola. Cuando dicho evento ocurra entonces el proceso obtiene el mensaje (que se supone que se borra de cola) y puede sea entrar en su sección crítica. Cuando sale ejecuta una operación send para enviar el mensaje de nuevo a la cola, y permitir que otro proceso que estuviera bloqueado en la cola de mensajes pueda desbloquearse y acceder a su región crítica.

Preguntas de autoevaluación tema 5

5.1. ¿Qué es un interbloqueo?

Aquella situación en la cual un conjunto de procesos está bloqueado en espera de la liberación de uno o varios recursos que se encuentran asignados a otro proceso del mismo conjunto. Como todos los procesos del conjunto están bloqueados, ninguno de ellos liberará los recursos que posee y que otro proceso necesita, y en consecuencia ninguno puede continuar su ejecución.

5.2. Enumerar y explicar las cuatro condiciones necesarias y suficientes para la existencia del interbloqueo.

- ***Exclusión mutua.*** Cada instancia de un recurso solo puede ser asignada a un proceso como máximo.
- ***Retención y espera.*** Cada proceso retiene los recursos que le han sido asignado mientras espera por la adquisición de los otros recursos que necesita.
- ***No existencia de expropiación.*** Si un proceso posee un recurso, éste no se le puede expropiar.
- ***Espera circular.*** Existe una cadena circular de dos o más procesos, de tal forma que cada proceso de la cadena se encuentra esperando por un recurso retenido por el siguiente proceso de la cadena.

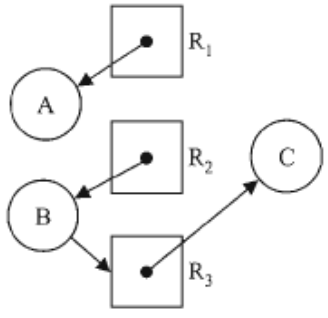
5.3. Enumerar las principales estrategias que puede implementar un sistema operativo para el tratamiento de los interbloqueos.

- ***Prevención de interbloqueos.*** Consiste en impedir que se produzca alguna de las cuatro condiciones necesarias para que se produzca el interbloqueo.
- ***Evitación de interbloqueos.*** Consiste en examinar las posibles consecuencias de asignar los recursos solicitados y evitar el interbloqueo mediante la no asignación de recursos en situaciones potencialmente peligrosas.
- ***Detección y recuperación de interbloqueos.*** Consiste en comprobar cada cierto tiempo el estado del sistema para detectar la existencia de interbloqueos y en el caso de que existan tomar las medidas oportunas para eliminarlos.

5.4. ¿Cómo se construye un grafo de asignación de recursos?

Se representa a cada proceso con un círculo y a cada recurso con un cuadrado. Cada instancia de un mismo recurso, se representa con un punto dentro del cuadrado asignado al recurso. Si un proceso ha solicitado una instancia de recurso y se encuentra bloqueado en espera de que le sea asignada, entonces se representa una flecha que sale del proceso hacia el recurso. Por otra parte, si un proceso tiene asignado una instancia de un recurso, entonces se representa una flecha que va desde la instancia del recurso al proceso.

Considérese un computador con una unidad de DVD (recurso R_1), una impresora (recurso R_2) y un disco duro (recurso R_3). Supóngase que se están ejecutando tres procesos A, B y C. En la Figura 5.1 se muestra el grafo de asignación de recursos en un determinado de tiempo T_1. El proceso A tiene asignado R_1, el proceso B tiene asignado R_2 y se encuentra bloqueado en espera de que le asignen R_3, que está asignado al proceso C. Se observa que en el grafo no existe ningún ciclo, luego no existe interbloqueo.



5.5. ¿Cómo se pueden detectar los interbloqueos en un grafo de asignación de recursos?

Si el grafo no contiene ningún camino que sea un ciclo entonces no existe interbloqueo. Por otra parte la existencia de un ciclo no es condición suficiente para que exista interbloqueo,

dependerá también del número de instancias de cada recurso implicado en el ciclo y del tipo de caminos de los que formen parte dichas instancias. Si los recursos que forman parte de un ciclo únicamente tienen una instancia entonces existe interbloqueo. Por otra parte, si en el ciclo existen recursos que tienen más de una instancia, entonces para que exista interbloqueo todas las instancias de dichos recursos deben formar parte de caminos que sean ciclos.

5.6. ¿En qué consiste la estrategia de prevención de interbloqueos?

Consiste en eliminar la aparición de alguna de las cuatro condiciones necesarias y suficientes para que se produzca un interbloqueo: exclusión mutua, retención y espera, no existencia de expropiación y espera circular.

5.7. ¿Es aconsejable eliminar la condición de exclusión mutua para prevenir el interbloqueo?

No si se quiere garantizar la integridad y la no corrupción de los mismos. Por ejemplo, si dos procesos quieren escribir en una impresora no deberían de hacerlo al mismo tiempo.

5.8. Señalar los inconvenientes de la eliminación de la condición de retención y espera para prevenir el interbloqueo.

Se puede eliminar forzando a que un proceso solicite a la vez todos los recursos que va a necesitar en su ejecución. Hasta que no pueda obtener todos los recursos permanecerá bloqueado.

Uno de los principales inconvenientes que tiene la eliminación de esta condición es que requiere conocer por anticipado todos los recursos que va a necesitar un proceso durante su ejecución, lo cual en muchas ocasiones (salvo quizás en trabajos por lotes) es difícil de saber.

Otro inconveniente es que produce problemas de rendimiento del sistema. Un proceso puede permanecer bloqueado un tiempo excesivo o incluso permanentemente (inanición) hasta poder obtener a la vez todos los recursos que necesita.

5.9. Explicar cómo se consigue la eliminación de la condición de no existencia de expropiación en la estrategia de prevención de interbloqueo.

Se consigue permitiendo que el SO pueda expropiar a un proceso los recursos que retiene. Solo es posible en recursos para los cuales el SO puede salvar el estado en el que se encontraban, para restaurarlo cuando el proceso expropiado consiga otra vez su uso. Este es el caso del procesador (cambio de contexto de un proceso) y la memoria principal (intercambio en la memoria secundaria). En consecuencia se debe valorar si la desaparición de los interbloqueos mediante la eliminación de la existencia de expropiación compensa la disminución del rendimiento del sistema debido a la sobrecarga por las tareas de guardar y restaurar el estado de los recursos.

5.10. Explicar una posible forma de eliminar la condición de espera con objeto de prevenir el interbloqueo.

Se puede eliminar asignando a cada recurso del sistema un número y forzando a que un proceso solo pueda solicitar los recursos en orden ascendente. La principal desventaja es que los recursos deben ser solicitados en el orden establecido, en lugar de pedirlos cuando se necesitan realmente. De esta forma, un proceso retiene un recurso que no va a utilizar hasta haber utilizado otros. Impidiendo así que pueda ser asignado a otro proceso.

5.11. Explicar en qué consiste la estrategia de evitación de interbloqueos.

También conocida como predicción de interbloqueos consiste en conceder a un proceso solamente aquellas peticiones de recursos que tengan garantizado que no conducirán a un estado de interbloqueo.

En esta estrategia un proceso tiene que especificar por adelantado todos los recursos que va a necesitar para su ejecución. Obviamente si los recursos solicitados por un proceso superan el número máximo de recursos (asignados o no) del sistema, entonces, dicho proceso no es admitido para ser ejecutado. Si están disponibles y su concesión no conduce a un estado de interbloqueo entonces se le concede su uso. En caso contrario, el proceso tiene que esperar y se bloquea.

El SO debe contabilizar el número de recursos disponibles, el número de recursos asignados a cada proceso y el número de

recursos que restan por asignar a cada proceso. Además, se encarga de comprobar si una asignación es segura o, por el contrario, puede conducir a un estado de interbloqueo.

5.12. ¿Qué es un estado seguro?

El estado del sistema con respecto a la asignación de sus recursos en un determinado instante de tiempo queda determinado por los vectores **Re, Rd** y las matrices **N** y **A**. Un estado es seguro si posibilita al menos una secuencia de asignación de recursos a los procesos que garantiza que éstos se pueden ejecutar hasta su finalización sin que se produzca interbloqueo, incluso aunque los procesos soliciten simultáneamente todos los recursos que van a necesitar.

¿Y un estado inseguro?

. Un estado inseguro es aquél que no garantiza que no se pueda producir un interbloqueo.

5.13. Describir la técnica de denegación de asignación de recursos.

La técnica más usada de evitación de interbloqueos es la denegación de asignación de recursos que consiste en no conocer a un proceso una petición adicional de un recurso si dicha petición puede conducir a un estado inseguro.

5.14. ¿Para qué se utiliza el algoritmo del banquero?

La implementación más conocida de la técnica de degeneración de asignación de recursos. Simula el comportamiento de un banquero que realiza préstamos y recibe pagos sin caer nunca en la posibilidad de no poder satisfacer todas las necesidades de sus clientes. La analogía es la siguiente: el banquero representa al SO, los clientes son los procesos y el dinero prestado los recursos.

El algoritmo asegura que el número de recursos asignados a todos los procesos nunca puede exceder del número de recursos del sistema. Además, nunca se puede hacer una asignación peligrosa, es decir, asignar recursos de modo que no queden suficientes para satisfacer las necesidades de todos los procesos.

¿Cuáles son sus pasos?

Los pasos de que consta este algoritmo son los siguientes:

1. Se parte del un estado inicial de asignación seguro S_k con k = 0.

2. Cuando un proceso realiza una petición de asignación de recursos, se simula que se concede la petición y se actualiza el estado del sistema. A este estado ficticio se le denota como S':

3. Se comprueba si S' es seguro.

. En caso afirmativo se concede la petición al proceso y se actualiza el estado del sistema:

S_{k+1} =S'

. En caso negativo la petición se deniega y el proceso se bloquea hasta que se le puedan conceder los recursos que solicita.

1. Volver al paso 2.

¿Qué inconvenientes presenta su uso?

En la práctica no se utiliza, ya que es difícil conocer por adelantado los recursos que van a necesitar los procesos durante su ejecución. Además dicho algoritmo parte de la suposición de que el número de procesos y de recursos existentes es fijo, lo cual no tiene por qué ser cierto. En general, la población de procesos varía dinámicamente en el tiempo. Por otra parte, los recursos que inicialmente están disponibles pueden que con el tiempo no lo estén debido a posibles fallos, por ejemplo, que se quede atascado papel en una impresora o que se dañe una unidad de cinta.

5.15. Describir la técnica de denegación de la iniciación de un proceso.

Consiste en no iniciar un proceso si el número máximo de recursos que va a necesitar durante su ejecución pueden conducir a un interbloqueo.

El proceso p+1 puede iniciar su ejecución si el número máximo de instancias de cada recurso j que necesita sumadas al número máximo de instancias de dichos recursos necesitadas por los procesos ya existentes no supera el número de instancias existentes. En consecuencia, el proceso puede iniciar su ejecución si el sistema está en un estado seguro.

No es óptima ya que supone que todos los procesos van a solicitar simultáneamente todas las instancias de recursos que van a necesitar, es decir, trabaja con el peor de los casos posibles

5.16. ¿En qué consiste la estrategia de detección y recuperación de interbloqueos?

En no limitar las asignaciones de recursos a los procesos y comprobar periódicamente, mediante la utilización de algún algoritmo, si se ha producido algún interbloqueo. En caso afirmativo se emplea alguna técnica para intentar recuperar al sistema del interbloqueo.

5.17. Enumerar los pasos del algoritmo de Coffman para la detección de interbloqueos.

Este algoritmo utiliza los siguientes vectores y matrices:

- **Re:** vector de recursos existentes

- **Rd:** vector de recursos disponibles

- **A:** Matriz de recursos asignados a cada proceso

- **M:** Matriz de recursos necesitados por cada proceso adicionalmente de los que ya posee.

1. Se marca cada proceso que tenga una fila i (i = 1,2,…,p) de la matriz de asignación **A** igual a cero ya que dicho proceso no retiene ningún recurso y en consecuencia no puede producir interbloqueo.

2. Se realiza la asignación **X = Rd** donde **X** es un vector auxiliar.

3. Para cada “i” no marcado se comprueba la condición Mi es mayor o igual a X donde Mi es la fila i de la matriz M.

4. Si no existe ningún i que cumpla la condición, entonces el algoritmo finaliza. Si existe algún i que cumpla la condición, entonces se marca el proceso, se realiza la suma vectorial X = X + Ai y se vuelve al paso 3.

5.18. Describir algunas técnicas de recuperación del interbloqueo.

- ***Recuperación mediante expropiación de recursos.*** Consiste en ir expropiando recursos a algunos procesos y concedérselos a otros hasta conseguir salir del interbloqueo. Esta técnica requiere establecer algún criterio para seleccionar los procesos a los que se les va a expropiar los recursos. En ocasiones como por ejemplo en los sistemas por lotes la expropiación se debe hacer de manera manual por el administrador del sistema.

- ***Recuperación mediante el retroceso de los procesos interbloqueados a algún punto de control previo.*** Consiste en ir realizando puntos de control (checkpoints) periódicamente durante el tiempo de vida de un proceso. En cada punto de control se guarda el contexto del proceso en un fichero distinto. Cuando se detecta que el proceso está interbloqueado se retrocede a su ejecución a un punto de control anterior con la esperanza de que se evite el interbloqueo.

- ***Recuperación mediante el aborto selectivo de procesos.*** Consiste en ir abortando selectivamente algún proceso para que libere los recursos que posee y puedan ser asignados a algún proceso interbloqueado de forma que pueda salir de dicho estado y continuar la ejecución. Esta técnica requiere establecer algún criterio de selección de los procesos que van a ser abortados, ya sea alguno de los interbloqueados o cualquier otro proceso.

- ***Recuperación mediante el aborto de todos los procesos interbloqueados.*** Se trata de la solución más drástica y curiosamente la más utilizada.

5.19. Señalar las ventajas y los inconvenientes de las técnicas de detección y recuperación de interbloqueos.

Es una estrategia de tratamiento de interbloqueos mucho menos conservativa que la prevención y la evitación de interbloqueos, ya que al no limitar el número de asignaciones permite un mayor grado de concurrencia de procesos.

La detección de interbloqueos introduce un grado de sobrecarga al sistema que puede ser tolerable mediante el uso de algoritmos de detección eficientes y mediante el ajuste de la frecuencia de invocación de dicho algoritmo.

Por otra parte la recuperación de interbloqueos puede producir una sobrecarga elevada al sistema. Además puede producir un desaprovechamiento de los recursos para aquellos procesos que son reiniciados o retornados a un punto de control anterior.

En general puede afirmarse que la recuperación de interbloqueos es bastante útil en sistemas con baja probabilidad de interbloqueos.

5.20. Explicar en qué consiste una estrategia mixta de tratamiento de interbloqueos.

Las estrategias de tratamiento de interbloqueos descritas pueden utilizarse conjuntamente con objeto de aprovechar sus ventajas y reducir sus inconvenientes. La forma de proceder es la siguiente:

1. Agrupar los recursos existentes del sistema en diferentes clases.
2. Ordenar las clases para evitar la espera circular
3. Utilizar para cada clase la estrategia de tratamiento de los interbloqueos que se considere más oportuna.

5.21. ¿En qué hipótesis se basa la estrategia de ignorar los interbloqueos?

Tratar los interbloqueos supone algún coste al sistema. Por este motivo muchos SO como UNIX o Windows, no utilizan ninguna estrategia de tratamiento de interbloqueos. Consideran que los usuarios prefieren sufrir algún interbloqueo de vez en cuando que disponer de un sistema que les limite el uso de recursos o con alta sobrecarga. Se basan en la suposición de que la probabilidad de que se produzca un interbloqueo es pequeña y confían en el azar para que no se produzca. En el caso de que se produzca será el

administrador del sistema el que tenga que encargar de detectar el interbloqueo y recuperarlo.

Preguntas de autoevaluación tema 6

6.1. Definir los siguientes conceptos: a) Espacio del núcleo. b) Espacio de usuario.

a).Un SO para ser ejecutado debe estar cargado en la memoria principal. El SO se carga cuando arranca el computador mediante un programa cargador. Al espacio ocupado por el código, las estructuras de datos y la pila (o pilas) del núcleo del SO se le denomina espacio del núcleo.

b).Aparte del SO, en la memoria principal pueden estar cargados (total o parcialmente) uno o varios procesos. Al espacio de memoria principal ocupado por la imagen de un proceso, es decir, por su espacio de direcciones de memoria lógica, se le denomina espacio de usuario.

6.2. ¿Qué es el área de intercambio en memoria secundaria?

Cuando se desea ejecutar un programa en el SO tiene que crear un proceso asociado a un programa en la memoria secundaria y cargarlo en la principal. Para ello debe buscar el archivo ejecutable, crear una copia de la imagen del proceso en la memoria secundaria y cargar dicha imagen en memoria principal. Algunos SO directamente crean y cargan la imagen del proceso en la memoria principal sin crear una copia en la memoria secundaria.

El SO reserva espacio en la memoria secundaria, típicamente un HD, para almacenar las copias de las imágenes de los procesos. A dicho espacio se le denomina área de intercambio. El tamaño de esa zona suele estar predefinido, aunque algunos sistemas permiten que su tamaño máximo sea configurado por el administrador.

6.3. ¿En qué consiste la operación de intercambio de procesos?

También llamado swapping es la operación de cargar la imagen de un proceso desde el área de intercambio a la memoria principal o viceversa.

6.4. ¿En qué casos se suele realizar la operación de intercambio fuera de memoria principal?

Swapping out o intercambio fuera de memoria principal se suele realizar en los siguientes casos:

- ***Cuando se requiere espacio en la memoria principal para cargar otros procesos.***
- ***Cuando un proceso necesita más espacio en memoria principal debido al crecimiento de su región de datos o de su región de pila.***
- ***Cuando es necesario regular el grado de multiprogramación del sistema, es decir, el número de procesos cargados en memoria principal.***

¿Y la operación de intercambio dentro de memoria principal?

Swapping in o intercambio dentro de memoria principal se suele realizar en los siguientes casos:

- ***Se desea aumentar el grado de multiprogramación del sistema para mejorar su rendimiento.***
- ***Existe el espacio en memoria principal para cargar más procesos.***

Existe un proceso de mayor prioridad en el área de intercambio.

6.5. ¿Cuáles son las principales tareas del intercambiador?

- ***Gestionar y asignar el espacio de intercambio.*** Para ello el SO mantiene la estructura de datos con información del espacio libre en el área de intercambio.
- ***Seleccionar procesos para ser intercambiados fuera de memoria principal.*** El intercambiador escoge en primer lugar para ser intercambiados aquellos procesos que se encuentran en estado bloqueado de acuerdo a los siguientes criterios: menor prioridad del proceso y mayor tiempo de residencia en

memoria. En segundo lugar escoge a procesos en el estado preparado, con los mismos criterios que el caso anterior.

- ***Seleccionar procesos para ser intercambiados dentro de memoria principal.*** Se suelen escoger aquellos procesos preparados para ejecución que llevan más tiempo en el área de intercambio o de mayor prioridad.

6.6. ¿Qué dos razones justifican la existencia del área de intercambio?

- ***Una operación de lectura y escritura en el área de intercambio es más rápida que en el sistema de archivos ya que no hay que traducir una ruta de acceso al archivo ni comprobar permisos de accesos.***

- ***La imagen de un proceso es una entidad dinámica cuyo contenido suele diferir del contenido del archivo ejecutable conforme se va ejecutando el proceso.*** La imagen del proceso debe ser guardada para poder continuar con la ejecución del proceso sin tener que reiniciarla desde el principio.

6.7. Describir cómo se realiza la asignación de memoria en sistemas monoprogramados.

Se divide en dos particiones, una reservada para contener permanentemente a aquellas partes del SO que deban estar siempre en memoria principal. Al conjunto de estas partes se le denomina monitor del SO.

La memoria principal no ocupada por el SO conforma otra partición que está disponible para almacenar de forma temporal a un proceso de usuario o a uno de sistema asociado a alguna parte no residente en el SO. Puesto que solo puede estar cargado un proceso en memoria principal, los procesos se ejecutan una a continuación de otro. Cuando un proceso finaliza, el SO puede cargar otro proceso en la única partición disponible para tal efecto.

En función de si la partición de memoria asociada al SO se encuentra en la parte inferior o superior es posible distinguir las siguientes configuraciones:

- ***Exclusivamente con una RAM***. La partición del SO ocupa la parte inferior de la memoria principal.

- ***Con una RAM y una ROM donde el SO ocupa la ROM desde Dir_{f1} hasta Dir_{max}.*** Es una configuración muy poco flexible ya que el código del SO al residir en una ROM no puede ser modificado y actualizado.

- ***Con una RAM y una ROM donde la partición del SO ocupa la parte inferior de la RAM y la memoria ROM contiene la tabla de vectores de interrupciones del hardware y las rutinas de servicio de las interrupciones, a esta memoria se le conoce también por BIOS***.

6.8. ¿En qué consiste la técnica de particionamiento fijo?

En sistemas multiprogramados se cargan múltiples procesos en el estado preparado para ejecución en la memoria principal. En un sistema con un único procesador, solo uno de los procesos podrá encontrarse en el estado ejecutándose. Si el proceso en ejecución se bloquea en espera de la aparición de un evento, como por ejemplo la finalización de una operación de E/S, otro proceso en el estado preparado para ejecución puede ser planificado para ser ejecutado. La multiprogramación permite mejorar el rendimiento del sistema, evitando que sus recursos se encuentren inactivos.

La forma más sencilla de gestionar la memoria en sistemas multiprogramados consiste en dividir la memoria principal en un número fijo N de particiones que pueden ser de igual o de diferente tamaño. Esta división puede realizarse, por ejemplo, manualmente en tiempo de arranque del sistema. A esta técnica se le conoce como particionamiento fijo.

6.9. ¿Qué información contiene la tabla de descripción de particiones?

El SO mantiene una estructura de datos, denominada tabla de descripción de particiones, que contiene la siguiente información por cada partición: dirección física de comienzo (también llamada dirección base), tamaño y estado (si está libre u ocupada). El SO se encarga de gestionar la entrada de cada partición, ya que la dirección base y el tamaño son fijados en tiempo de arranque.

Cuando se asigna una partición a un proceso, el SO guarda el número de partición asignada en el bloque de control de proceso. Además marca como ocupada la entrada de la tabla de descripción de particiones asociada a dicha partición.

Cuando un proceso finaliza su ejecución o es intercambiado al área de intercambio en memoria secundaria su partición de memoria queda libre para ubicar a otro nuevo proceso o a un proceso intercambiado procedente del área de intercambio.

Cuando hay que asignar una partición a un proceso se busca una partición libre en la tabla de descripción de particiones.

6.10. ¿Qué desventajas presenta el particionamiento fijo con particiones de igual tamaño?

A pesar de tener la ventaja de que no importa qué partición libre se asigne a un proceso al ser todas iguales presenta las siguientes desventajas:

- ***Limitación del tamaño máximo de los procesos que se pueden cargar en memoria principal.*** Un proceso cuyo espacio de direcciones lógicas sea de mayor tamaño que el tamaño fijado para las particiones no podrá ser cargado en memoria y en consecuencia no podrá ser ejecutado. Existe una excepción y es si el programa ha sido diseñado con superposiciones (overlay). Una superposición es un conjunto de instrucciones y datos que se pueden y ejecutar y acceder de forma autónoma. Un programa se compone de varias superposiciones que se almacenan en memoria secundaria. En un determinado instante de tiempo solo es necesario, que para que pueda progresar en su ejecución, que esté cargada en memoria principal una única superposición del proceso. Cuando una superposición de un programa termina de ejecutarse invoca a otra superposición que es cargada desde memoria secundaria en el espacio de memoria principal que ocupaba la anterior. Esta técnica permite ahorrar espacio de memoria. Sin embargo, el diseño y la programación de un programa con superposiciones bastante tediosa, sobre todo en el caso de programas largos.

- ***Fragmentación interna.*** Si el tamaño del espacio de direcciones lógicas de un proceso es inferior al tamaño de la partición que se la ha asignado entonces existe un espacio no utilizado dentro de dicha partición. Nótese que este espacio no puede ser utilizado por ningún otro proceso, es decir, es un espacio desperdiciado. Al desaprovechamiento de la memoria dentro de una partición se le denomina fragmentación interna.

6.11. ¿Qué ventajas e inconvenientes presenta el uso de una cola por partición?

La ventaja es que intenta minimizar la fragmentación interna dentro de cada partición. Sin embargo, puede provocar que las particiones de mayor tamaño se queden vacías mientras que las colas asociadas a particiones de menor tamaño están llenas de procesos esperando. Es decir, está infrautilizando la memoria.

Una posible solución consiste en mantener una única cola de procesos para todas las particiones.

6.12. ¿Qué criterios de búsqueda se emplean con las particiones de distinto tamaño y una única cola de procesos?

- ***Criterio del primer ajuste:*** Consiste en buscar el proceso más cercano a la cabecera de la cola cuyo tamaño sea igual o menor al de la partición libre. Este criterio de búsqueda produce búsquedas rápidas, sin embargo no minimiza la fragmentación interna.

- ***Criterio del mejor ajuste*****:** Busca en toda la cola al proceso de mayor tamaño que entre dentro de la partición. Minimiza la fragmentación interna pero produce búsquedas más lentas. Además discrimina a los procesos de menor tamaño, como suelen ser los asociados a tareas interactivas. Obviamente estos procesos deben ejecutarse rápidamente y no ser relegados continuamente por otros procesos de mayor tamaño. Una posible solución sería imponer un valor máximo de veces que un proceso puede ser discriminado.

6.13. ¿Qué desventajas presenta el particionarniento fijo con particiones de distinto tamaño?

Los mismos que con el uso de particiones de igual tamaño: la existencia de fragmentación interna y la limitación del tamaño

máximo del espacio de direcciones lógicas del proceso que puede ser cargado en memoria. Sin embargo ambos problemas son de una magnitud menor que en el caso de particiones de tamaño fijo.

6.14. ¿Cómo se realiza la traducción de direcciones lógicas a físicas y la protección en la técnica de gestión de memoria mediante particionarniento fijo?

Requiere el uso de dos registros denominados registro base y registro límite.

Cuando un proceso es planificado para ser ejecutado, el SO carga en el registro base la dirección física base de la partición de memoria principal donde se encuentra cargado el proceso. Además carga el registro límite con la dirección lógica más alta del espacio de direcciones lógicas del proceso, es decir, con el tamaño de su espacio lógico. El SO obtiene la información que carga en estos registros del bloque de control del proceso.

Cuando el procesador hace una referencia a una dirección lógica se compara con el valor de registro límite. Si la dirección es mayor que dicho valor, entonces el hardware genera una excepción por error de direccionamiento que al ser tratada por el SO típicamente suele producir la terminación del proceso y la notificación de dicha circunstancia al usuario propietario del proceso. Si la dirección lógica es menor que el valor del registro límite entonces se le suma el contenido del registro base para generar la dirección física correspondiente.

Nótese que el registro base impide el acceso de un proceso a direcciones físicas inferiores a las asignadas a su espacio lógico. Por otra parte el registro límite impide el acceso a direcciones físicas superiores a las asignadas a su espacio lógico. De esta forma se protege el espacio de direcciones lógicas del SO o el espacio de otro proceso contra el acceso involuntario o intencionado del proceso en ejecución.

6.15. ¿Cuáles son las principales ventajas e inconvenientes de la técnica de gestión de memoria mediante particionarniento fijo?

La principal ventaja es que es sencilla de implementar para el SO, únicamente requiere una tabla de descripción de particiones y

de una o varias colas para contener a los procesos que deben ser cargados en memoria. La gestión de estas estructuras produce muy poca sobrecarga.

Los principales inconvenientes de esta técnica son que provoca fragmentación interna y que el tamaño máximo de proceso que puede ser cargado en memoria viene limitado por la partición de mayor tamaño que se defina.

Además el número máximo de procesos cargados en memoria principal o grado de multiprogramación está limitado por el número de particiones que se definan en el momento de arrancar el sistema. Ello repercute en una disminución del rendimiento máximo del sistema, es decir, del grado de utilización del procesador y de los canales de E/S.

6.16. ¿En qué consiste la técnica de gestión de memoria denominada particionamiento dinámico?

Se caracteriza en que el número de particiones no es fijo sino que varía a lo largo del tiempo. Así, en la memoria principal se distinguen dos tipos de elementos: particiones y huecos.

Inicialmente al arrancar hay una partición y un gran hueco que se va rellenando con particiones.

6.17. ¿Qué es la fragmentación externa?

Con el paso del tiempo, al crearse nuevos procesos e intercambiarse otros, la memoria se va fragmentando entremezclando particiones asignadas con huecos. En la memoria principal irán apareciendo huecos cada vez más pequeños, algunos de los cuales no pueden ser utilizados por sí solos para crear otras particiones debido a su pequeño tamaño. A este desaprovechamiento de memoria entre particiones debido a la dispersión del espacio libre en una serie de huecos de pequeño tamaño se le denomina fragmentación externa.

¿Cómo se puede solucionar?

La fragmentación interna supone un problema cuando aunque exista tamaño suficiente en memoria este no es contiguo. Una forma de solucionarlo es la compactación de memoria, que consiste en agrupar todos los huecos pequeños en un único hueco. Una

posible implementación de esta técnica consiste en mover todas las particiones hacia un extremo de la memoria de forma que estén contiguas. Así se consigue que los huecos se desplacen hacia el otro extremo y se agrupen para formar un único hueco. El principal inconveniente de la compactación es que consume bastante tiempo de uso de procesador, produciendo una alta sobrecarga.

6.18. ¿Qué estructuras de datos debe mantener el sistema operativo para implementar el particionamiento dinámico?

Debe de contener una o varias estructuras de datos que contengan la información sobre las particiones y los huecos existentes en la memoria principal.

Una posibilidad es mantener una única estructura de datos que mantenga simultáneamente las particiones y los huecos existentes en memoria principal. Puede implementarse con un mapa de bits o una lista enlazada (simple o doble). En el caso de usar una lista ésta puede ser organizada por direcciones físicas o por tamaños.

Otra posibilidad consiste en mantener dos listas independientes: una de particiones y otras de huecos. Así se consigue acelerar el proceso de búsqueda de huecos, sobre todo si dicha lista ésta organizada por tamaños. Sin embargo la operación de desasignación de memoria se ralentiza ya que cuando un proceso finaliza, o es intercambiado al área de intercambio, el SO tiene que actualizar dos estructuras de datos.

6.19. Enumerar y describir brevemente algunos de los algoritmos de búsqueda más empleados en la asignación de memoria en el particionarniento dinámico.

- ***Primer ajuste (first fit*).** Se comienza a buscar desde el principio de la estructura de datos y se asigna el primer hueco con un tamaño igual o mayor a S. Este algoritmo es sencillo de implementar. Además es rápido ya que minimiza el tiempo de búsqueda. Con el paso del tiempo este algoritmo produce pequeños huecos al comienzo de la estructura de datos.

- ***Algoritmo del siguiente ajuste (next fit).*** La búsqueda en la estructura de datos de un hueco de tamaño igual o mayor a S

se inicia desde donde finalizó la búsqueda anterior. Aunque este algoritmo se ideó con la intención de mejorar el aprovechamiento de memoria que producía el algoritmo del primer ajuste, diversas simulaciones han demostrado que con el paso del tiempo produce resultados ligeramente peores, tanto en tiempo de búsqueda como en aprovechamiento de memoria.

- ***Algoritmo del mejor ajuste (best fit).*** Se busca en toda la estructura de datos un hueco cuyo tamaño sea igual o exceda lo mínimo posible al valor S. Este algoritmo, al buscar en toda la estructura de datos, es más lento que el algoritmo del primer ajuste o del siguiente ajuste. Además produce huecos de memoria muy pequeños que no suelen ser utilizables para posteriores asignaciones, lo que obliga a realizar compactación de memoria más frecuentemente.
- ***Algoritmo del peor ajuste (worst fit).*** Se busca en toda la estructura de datos un hueco cuyo tamaño sea igual o exceda lo máximo posible el valor S, es decir, se busca el hueco más grande posible. Con ello se pretende que los huecos que se generen sean aprovechables para posteriores asignaciones y en consecuencia haya que realizar compactación menos frecuentemente. Diferentes simulaciones han demostrado que con el paso del tiempo este algoritmo no es muy efectivo reduciendo la fragmentación externa y la frecuencia de realización de compactación.

De estos cuatro el más rápido es el algoritmo del primer ajuste, lo cual es lógico ya que en principio, no necesita buscar en toda la estructura de datos. Por otra parte en cuanto a al aprovechamiento de memoria el mejor es el primer ajuste y el peor el peor ajuste.

6.20.. ¿Cómo se realiza la traducción de direcciones lógicas a físicas y la protección en la técnica de gestión de memoria mediante particionamiento dinámico?

Se implementa de igual manera que en la técnica del particionamiento fijo, es decir, se utilizan dos registros dedicados denominados registro base y registro límite. El SO, cuando se va a

ejecutar un proceso, carga en el registro base la dirección física de inicio de la partición de memoria física asociada al proceso, y carga el registro límite el tamaño de la partición.

6.21. Señala las principales ventajas e inconvenientes de la técnica de particionamiento dinámico frente a la técnica de particionarniento fijo.

Las ventajas son:

- ***Minimiza la fragmentación interna*.** Como se ha comentado la fragmentación interna de una partición es prácticamente despreciable, ya que como máximo será cercana a una unidad de asignación.
- ***Permite ejecutar procesos de mayor tamaño.*** En un determinado instante de tiempo el tamaño máximo del proceso que puede cargarse en memoria viene limitado por la existencia de un hueco de tamaño igual o mayor que lo pueda contener. Inicialmente en la memoria solo está cargado el SO, por lo que el resto de la memoria formaría un hueco de tamaño S que podría albergar un proceso de tamaño máximo menor o igual a S.
- ***Permite asignar fácilmente más espacio a procesos cuya región de datos o pila aumenta por encima del tamaño inicialmente asignado.*** Para ello el SO puede proceder de dos formas. Una opción es crear una partición de mayor tamaño y reubicar al proceso en dicha partición. Otra posibilidad es añadir a la partición del proceso el espacio de un hueco adyacente a la partición.

Las desventajas son las siguientes:

- ***Las estructuras de datos que requiere para su implementación y los algoritmos para su gestión son más complejos.*** En consecuencia se produce una mayor sobrecarga del sistema.
- ***Produce fragmentación externa.*** Lo que reduce el aprovechamiento de la memoria.

6.22. ¿En qué consiste la paginación?

Consiste en dividir la memoria principal en bloques del mismo tamaño denominados marcos de página o páginas físicas.

6.23. Cuando se usa paginación ¿en qué campos se descompone una dirección física y una dirección lógica?

El espacio de direcciones de un proceso también se divide en bloques del mismo tamaño Sp denominados páginas o páginas lógicas. Una página de un proceso se carga en un marco de página libre de memoria principal. Además las páginas de un mismo proceso no tienen porque ocupar marcos contiguos.

¿Cómo se puede determinar el tamaño de dichos campos?

Sea Cmp la capacidad de memoria principal, el número Nmp de marcos de página de tamaño Sp en que se divide la memoria principal se obtiene de la siguiente forma:

$$Nmp = floor\left(\frac{Cmp}{Sp}\right)$$

Cada marco de página tiene asignado un identificador numérico entero positivo j que lo identifica de forma unívoca. Al número j se le denomina número de marco de página. Así se habla del marco de página j o marco j de memoria principal.

Por otra parte, si Cx es el tamaño del espacio de direcciones lógicas de un proceso, el número Np de páginas de tamaño Sp en que se descompone el espacio de proceso se obtiene de la siguiente forma:

$$Np = ceil\left(\frac{Cx}{Sp}\right)$$

6.24. Enumerar y describir brevemente las estructuras de datos que utiliza el sistema operativo para implementar la paginación.

- ***Tablas de páginas.*** Cada proceso tiene asignada una tabla de páginas. Cada entrada i de la tabla de páginas de un proceso X contiene, entre otras informaciones, el marco j donde se encuentra almacena la página i del proceso X y los permisos de acceso a la página. Además puede contener información sobre la ubicación de la copia de la página en

memoria secundaria, aunque a veces esta información se implementa en una tabla diferente.

- ***Tabla de marcos de página.*** Esta tabla tiene tantas entradas como marcos de página tiene la memoria principal. Cada entrada j de la tabla de marcos de página contiene, entre otras, las siguientes informaciones relativas al marco: su estado, es decir, si está libre u ocupado, punteros para crear una lista de marcos libres, y la ubicación en memoria secundaria de la página i contenida en el marco.

- ***Lista de marcos libres.*** Es consultada cuando hay que asignar espacio a los procesos que deben ser cargados en memoria principal. Cuando hay que asignar un marco se selecciona al primero de la lista. Si un marco queda libre es colocado al final de la lista.

6.25. ¿Qué es la paginación simple?

La técnica de paginación de memoria puede implementarse de diferentes formas. En su implementación más sencilla, denominada paginación simple, para que un proceso X pueda ser ejecutado es necesario que todas sus páginas se encuentren cargadas en memoria principal, aunque no es necesaria que estén cargadas en marcos contiguos.

En la paginación simple cuando un proceso X de tamaño Np páginas tiene que ser cargado en memoria principal, en SO comprueba la existencia de Np marcos libres en la lista de marcos libres. Si no hay marcos suficientes la carga del proceso tendrá que ser pospuesta hasta existan Np marcos libres. Si hay marcos libres suficientes, entonces se les asigna al proceso, los elimina de la lista de marcos libres, los marca como ocupados en la tabla de marcos de página y carga en ellos las páginas del proceso. A continuación, se crea o se actualiza (si había sido intercambiado) la tabla de páginas del proceso indicando el número de marco donde está cargada cada página. Además, si la tabla de páginas no estaba ya creada, almacena en el bloque de control del proceso un puntero a la dirección física de comienzo de la tabla de páginas.

Si un proceso fuese intercambiado fuera de memoria, todas sus páginas desaparecerían de memoria, cuando volviese a ser cargado en memoria sus páginas se cargarán en otros marcos.

6.26. Describir la traducción de direcciones en paginación con un registro base.

Cuando un proceso es planificado para ser ejecutado, el SO carga en dicho registro la dirección física de comienzo de la tabla de páginas del proceso.

Cuando durante un ciclo de instrucción se referencia a una dirección lógica Dir_L, se suma el campo número de página i de la dirección lógica a Dir_{F0} para obtener la dirección física Dir_{F1} de la entrada i de la tabla de páginas. A continuación se realiza una operación de lectura a memoria principal para acceder a Dir_{F1} y obtener el marco de página j donde se encuentra almacenada la página i.

Una vez obtenido el número de marco j donde se aloja la página i ya se tiene toda la información para construir la dirección física DirF a la que equivale la dirección lógica Dir_L, ya que el campo de desplazamiento es común en ambas direcciones.

El mecanismo descrito de traducción de direcciones tiene como principal ventaja que únicamente requiere que el SO cargue el registro base en cada cambio de proceso, lo cual repercute en unos tiempos de cambio de proceso más pequeños.

Su principal inconveniente es que la traducción de una dirección lógica requiere de un acceso a la memoria principal para localizar en la tabla de páginas el marco j que contiene la página i. Este acceso introduce un retardo en la ejecución de la instrucción en curso. Por este motivo este mecanismo de traducción de direcciones se utiliza en raras ocasiones. Sí se suelen utilizar variaciones del mismo.

6.27. Describir la traducción de direcciones en paginación con un banco de registros.

En un banco de registros el SO puede cargar en él, cuando se produce un cambio de proceso, una copia de la tabla de páginas del proceso que se va a ejecutar.

De esta forma cuando se referencia a una dirección lógica Dir_L, se utiliza el campo número e página i de Dir_L para acceder a la entrada i de la copia de la tabla de páginas del proceso almacenada en el banco de registros y obtener el marco de página j donde se encuentra almacenada la página i.

Una vez obtenido el número de marco j donde se aloja la página i ya se puede construir la dirección física Dir_F a la que equivale la dirección lógica, ya que el campo desplazamiento es común en ambas direcciones.

Tiene como principal ventaja que la traducción de una dirección lógica no requiere de ningún acceso a memoria principal. Su principal inconveniente, sin embargo, es que solo se puede utilizar si se asegura que el tamaño de las tablas de páginas de los procesos nunca será mayor que el tamaño del banco de registros. Debe tenerse en cuenta que el banco de registros suele tener entre 64 y 256 registros.

Otro inconveniente de este mecanismo de traducción es que el SO, cuando se produce un cambio de proceso, tiene que copiar en el banco de registros la tabla de páginas del nuevo proceso planificado lo que aumenta el tiempo de cambio de proceso y por tanto la sobrecarga.

6.28. Describir la traducción de direcciones en paginación con un TLB.

Un TLB es una memoria caché especial de alta velocidad. Su funcionamiento es similar a la memoria caché del procesador. Cada entrada del TLB contiene una copia de una entrada de la tabla de páginas del proceso actualmente en ejecución. Conviene matizar que no es necesario copiar en una entrada del TLB todos los campos de una entrada de la tabla de páginas del proceso en ejecución sino únicamente aquellos que son necesarios para poder realizar la traducción de direcciones (número de página i y de marco j), la protección, o determinar si la página ha sido modificada. Un TLB puede tener entre 8 y 4096 entradas.

Cuando el procesador hace una referencia a una dirección lógica Dir_L, el TLB examina en paralelo el campo número de página

i de cada una de sus entradas para ver si alguno coincide con el número de página i al que hace referencia Dir_L. Si el campo nº de página i de alguna entrada del TLB, por ejemplo la entrada h, coincide con el nº de página i de la dirección lógica, se dice que se ha producido un acierto en el TLB. Si nunca coincide se dice que se ha producido un fallo en el TLB.

Si se produce un acierto se lee el campo nº de marco j de dicha entrada h para formar la dirección física Dir_F junto con el campo de desplazamiento Dir_L.

Si se produce un fallo la gestión del mismo requiere de las siguientes acciones.

- ***Una operación de lectura a memoria principal para acceder a la entrada i de la tabla de páginas del proceso y obtener el marco de página j donde se encuentra almacenada la página i***. Con dicha información ya se puede formar Dir_F.
- ***Copiar en una entrada vacía del TLB, el contenido de dicha entrada i.*** Si todas las entradas del TLB están ocupadas entonces alguna debe ser elegida para ser reemplazada de acuerdo con algún algoritmo de reemplazamiento. Por lo tanto, si en un futuro próximo se vuelve a referenciar a una dirección lógica asociada a la página i del proceso en ejecución se producirá un acierto en el TLB ya que su entrada asociada en la tabla de páginas estará cargada en una entrada del TLB.

  La gestión de un fallo en le TLB la puede realizar el hardware, el SO o ambos en colaboración. Si la gestión del fallo la realiza el hardware únicamente o en colaboración con el SO, entonces también se dispone de un registro base donde se almacena la dirección física de comienzo de la tabla de página del proceso en ejecución.

  Cuando se produce un cambio de proceso, el SO tiene que cargar el registro base. Además tiene que borrar todas las entradas del TLB, ya que la información que contiene no sirve

para el próximo proceso que se va a ejecutar. Por lo tanto en los primeros instantes de la ejecución de un proceso se producen fallos en el TLB.

6.29. ¿Qué es una tabla de páginas paginada?

La tabla de páginas de un proceso se ubica en memoria principal ocupando un rango de direcciones físicas contiguas, para facilitar su localización y acceso. Si el proceso posee un espacio de direcciones lógicas grande, entonces su tabla de páginas ocupará una cantidad no despreciable de memoria física contigua.

Una forma de tratar las tablas de páginas grandes es descomponerla también en páginas, se tiene por tanto una tabla de páginas paginada. De esta forma la tabla de páginas se fragmenta y puede ubicarse en memoria principal de forma no contigua. Así se pueden distinguir dos tipos de páginas:

- ***Páginas ordinarias.*** Contienen instrucciones y datos del espacio lógico de un proceso.
- ***Páginas de tablas de páginas*****.** Contienen entradas de una tabla de páginas.

¿Cuántos accesos a memoria requiere?

Para localizar en memoria principal a las páginas de tabla de páginas se utiliza otra tabla de páginas denominada tabla de páginas de primer nivel o primaria. A las entradas de la tabla de páginas original contenidas en una página se les denomina tabla de páginas de segundo nivel o secundaria. Por lo tanto para cada proceso existe una única tabla de primer nivel y varias de segundo nivel, tantas como páginas ocupen la tabla de páginas original del proceso.

6.30. ¿Qué es una tabla de páginas invertida?

Tiene asociada una entrada por cada marco j de memoria principal, en vez de una entrada por cada página i de un proceso. De ahí el nombre de tabla invertida. En consecuencia el número de entradas de la tabla de páginas invertida es fijo y es igual al número de marcos. No depende, por tanto, del número de procesos que haya en el SO o del número de páginas en que se divida el espacio virtual de estos procesos.

Cada entrada j de la tabla de páginas invertida contiene típicamente los siguientes campos: número de página i alojada en el marco j, identificador numérico del proceso al que pertenece dicha página y bits de control (protección, referenciada, etc).

6.31. ¿Cómo se realiza la protección en la técnica de paginación?

En la técnica de gestión de memoria mediante paginación la protección del espacio de direcciones de un proceso se implementa a nivel de página. En cada entrada i de la tabla de páginas de un proceso existe un campo protección con un tamaño de uno o varios bits que en función de su valor permiten establecer el tipo de acceso que se permiten sobre dicha página i: solo lectura, lectura y escritura, ejecución, …

Cada vez que el procesador referencia a una dirección lógica se comprueba, antes de traducir la dirección, que el tipo de acceso que se pretende realizar sobre la página i no viola los permisos establecidos por el campo de protección. Si se detecta un intento de acceso incorrecto entonces el hardware genera una excepción denominada fallo de protección. El tratamiento de dicho fallo por parte del SO, suele ocasionar la finalización del proceso y el envío de un mensaje de aviso al usuario del proceso.

Adicionalmente a este mecanismo de protección a nivel de página, suele existir también un registro límite donde el SO almacena en cada cambio de proceso el número de páginas de que consta el proceso que va a ser ejecutado. Cuando se produce una referencia a una dirección lógica, antes de examinar el campo de protección, se compara el campo número de página de la dirección lógica con el valor almacenado en el registro límite. Si el número de página es mayor que dicho valor, entonces el hardware provoca una excepción que debe ser atendida por el SO. El tratamiento de esa excepción suele producir las mismas acciones comentadas para el caso de un fallo de excepción.

6.32. ¿Cómo se implementa la compartición de páginas?

Cada vez que se ejecuta un programa el SO crea un proceso asociado a su ejecución. El espacio lógico de dicho proceso se

divide típicamente en tres regiones: código, datos y pila. Para poder ser ejecutado en un esquema de paginación simple todas las páginas en que se descompone el espacio lógico del proceso deben estar cargadas en memoria.

En entornos de tiempo compartido varios usuarios pueden estar ejecutando simultáneamente el mismo programa. Supóngase que se ejecutan concurrentemente tres instancias del mismo programa, cada instancia tendrá asociada un proceso. Esto significa que existirán en memoria tres copias idénticas de las páginas del código del programa. Si el programa posee código reentrante, es decir, que no se puede modificar, se ahorraría espacio en memoria si solo se mantuviese cargada en memoria principal una única copia de las páginas de código de dicho programa. Dichas páginas son compartidas en modo solo lectura por todos los procesos asociados a un mismo programa.

Es el SO el que detecta si el código de un programa es reentrante. En caso afirmativo solo carga en la memoria principal una copia de las páginas del código de un programa, independientemente del número de instancias de dicho programa que se estén ejecutando.

En cada entrada de la tabla de marcos existe un contador de referencias que indica el número de entradas en las tablas de páginas que están referenciando a una página física compartida. De esta forma una página solo puede ser eliminada de la memoria si el contador de referencias es cero.

6.33. Enumerar las ventajas y los inconvenientes de la paginación simple.

Ventajas:

- ***No produce fragmentación externa.***
- ***Produce una sobrecarga pequeña***
- ***No necesita intervención humana.***
- ***Permite compartir entre varios procesos un código común.***

La principal desventaja es que produce fragmentación interna. Suele suceder que se desaprovecha parte del espacio de la última página asignada al proceso.

Puede considerarse como ventaja o inconveniente que la paginación es invisible al programador.

6.34. ¿En qué consiste la segmentación?

Cada segmento tiene asignado un nombre (código principal, subrutinas, datos, pila, etc) y una longitud. El compilador compila cada segmento comenzado por la dirección lógica 0. De esta forma cada segmento tiene su propio espacio de direcciones lógicas.

Por otra parte, cuando el SO crea un proceso asociado a un determinado programa, asigna a cada segmento un identificador numérico entero positivo h. Al número h se le denomina número del segmento. Así se habla del segmento h del proceso X.

6.35. ¿Cuáles son los campos de que consta la dirección lógica de un segmento?

Una dirección lógica consta de dos campos: El número de segmento de s bits y el desplazamiento dentro del segmento de d bits. El tamaño s del campo número de segmento se obtiene a partir del número total de segmentos Ns de que consta un proceso X.

¿Cómo se puede determinar su tamaño?

Por su parte el tamaño mínimo d del campo desplazamiento dentro de un segmento se obtiene a partir del tamaño Ss del segmento expresado en unidades direccionables (usualmente palabras).

Para un proceso que conste de Ns segmentos, el campo número de segmento será del mismo tamaño para todas las direcciones lógicas. Sin embargo, el tamaño mínimo del campo desplazamiento dependerá de la longitud de segmento al que esté asociado dicha dirección lógica. Por otra parte, en la segmentación simple una dirección lógica nunca puede ser de mayor tamaño que una dirección física.

6.36. ¿Qué es la segmentación simple?

La segmentación simple es una técnica de gestión de la memoria que soporta los elementos en los que se descompone el código de un programa. Básicamente el programa es dividido por el compilador en segmentos. Cada segmento es una entidad lógica conocida por el programador asociada a una determinada estructura de datos o a un módulo, pero nunca a una mezcla de ambos.

6.137. ¿Qué información contiene una tabla de segmentos?

Esta tabla está indexada por el número de segmento, es decir, tiene una entrada por cada segmento del proceso. Cada entrada contiene la dirección física base de cada segmento y la longitud del segmento (en unidades de asignación). También contiene información relativa a los permisos de acceso al segmento (lectura, escritura, ejecución). Además puede contener información sobre ubicación de la copia del segmento en memoria secundaria, aunque a veces esta información se implementa en una tabla diferente.

6.38. Describir la traducción de direcciones en segmentación con un registro base.

En la técnica de gestión de memoria mediante segmentación, la traducción de direcciones lógicas a direcciones físicas requiere que el SO cargue en el hardware cierta información sobre la tabla de segmentos del proceso actualmente en ejecución. La información que caga depende de los componentes hardware disponibles para realizar la traducción de direcciones. Si se dispone de un registro base, entonces el SO, en cada cambio de proceso, carga en este registro la dirección física Dirf0 de comienzo de la tabla de segmentos. El SO obtiene esta dirección del bloque de control del proceso.

Cuando se hace una referencia a una dirección lógica Dirl se realizan los siguientes pasos para traducirla a una dirección física:

1. Se suman el campo número de segmento h de la dirección lógica y Dirf0 para obtener la dirección física Dirf1 de la entrada h de la tabla de segmentos.

2. Se accede a Dirf1 para leer la dirección física base y la longitud del segmento h.

3. Se compara el valor del campo desplazamiento de la dirección lógica con la longitud del segmento h. Si el desplazamiento es mayor entonces el hardware genera una excepción por violación del tamaño del segmento.

4. Si el desplazamiento es menor que la longitud del segmento h se suma a la dirección física base del segmento h el desplazamiento (Desp.) indicado en la dirección lógica para obtener la dirección física Dirf2 a la que equivale Dirl.

   Tiene como principal ventaja que únicamente requiere que el SO cargue en cada cambio de contexto un solo registro, lo cual disminuye el tiempo de cambio de proceso. Su principal inconveniente es que requiere un acceso a la memoria principal para leer en la tabla de segmentos la dirección física base y la longitud del segmento. Este acceso introduce un retardo en la ejecución de instrucciones en curso.

Para acelerar la traducción de direcciones, al igual que sucedía con la técnica de paginación, se puede utilizar un banco de registros o un TLB.

6.39. ¿Cómo se implementa la protección en la técnica de segmentación?

Se implementa a nivel de segmento. En cada entrada h de la tabla de segmentos de un proceso existe un campo protección con un tamaño de uno o varios bits que en función de su valor permiten establecer el tipo de acceso que se permiten sobre dicho segmento.

6.40. ¿Cómo se implementa la compartición de segmentos en la técnica de segmentación?

El SO mantiene una tabla con los segmentos cargados en memoria principal. Cada entrada de esta tabla contiene, entre otras informaciones relativas a un segmento cargado en memoria, un contador de referencias que mantiene el número de procesos que referencian a dicho segmento a través de alguna entrada de sus tablas de segmentos, y en consecuencia lo comparten. Dicho segmento no podrá ser eliminado de memoria principal hasta que el contador de referencia valga 0.

6.41. Enumerar las ventajas y los inconvenientes de la segmentación simple.

Ventajas:

- ***Produce una fragmentación interna despreciable.*** Cuando se asigna espacio dentro de un hueco de memoria para un segmento de un proceso, dicho espacio se asigna como un múltiplo de una determinada unidad de asignación.

  Por ello la fragmentación interna por segmento será como máximo del orden de una unidad de asignación.

- ***Soporta la visión modular que el programador posee de su programa.***

- ***Permite manejar con facilidad estructuras de datos que crecen.***

- ***Facilita la protección y compartición de las diferentes pares de un programa.***

Inconvenientes:

- ***Produce una fragmentación externa.***

- ***Aumenta la sobrecarga del sistema.***

6.42. ¿En qué consiste la técnica de segmentación con paginación?

Consiste en combinar ambas técnicas. Básicamente consiste en dividir cada segmento de un proceso en un determinado número de páginas.

Una dirección lógica se descompone en tres campos: número de segmento de s bits, número de página de p bits y desplazamiento dentro de la página de k bits. Nótese que la suma de la longitud del campo número de página y desplazamiento dentro de la página, es igual a la longitud del campo desplazamiento dentro de un segmento de una dirección lógica en segmentación.

Cuando se carga un proceso en memoria principal el SO crea la tabla de segmentos del proceso y varias tablas de páginas (una

por cada segmento). Ahora cada entrada de la tabla de segmentos no contiene la dirección física base del segmento sino la dirección física base de la tabla de páginas asociada al segmento. Además el campo longitud del segmento viene expresado en número de páginas.

Si se dispone de un registro base, entonces el SO, en cada cambio de proceso, carga en dicho registro base la dirección física Dirf0 de comienzo de la tabla de segmentos. Cuando el procesador hace una referencia a una dirección lógica Dirl, se realizan los siguientes pasos para traducirla a una dirección física:

1. Se suma el campo número de segmento h de la dirección lógica y Dirf0 para obtener la dirección física DirF1 de la entrada h de la tabla de segmentos.
2. Se accede a Dirf1 para leer la dirección física base de comienzo de la tabla de páginas (Base) y la longitud del segmento h (Longitud).
3. Se compara el valor del campo desplazamiento de Dirl con la longitud del segmento h. Si el desplazamiento es mayor entonces el hardware genera una excepción por violación del tamaño del segmento.
4. Si el desplazamiento es menor que la longitud del segmento h, entonces se suma el campo número de página de Dirl y la dirección física base de la tabla de páginas del segmento para generar la dirección física Dirf2 de la entrada i de la tabla de páginas.
5. Se accede a Dirf2 para leer el número de marco j que contiene la página i referenciada.
6. Se construye la dirección física Dirf con el número de marco j leído y con el campo desplazamiento de Dirl.

   El mecanismo descrito de traducción de direcciones realiza dos accesos a memoria principal: el primero para leer en la entrada h de la tabla de segmentos la dirección física base de la tabla de páginas y la longitud del segmento, el segundo

para leer en la entrada i de la tabla de páginas el marco de página j donde se aloja la página i. Para acelerar la traducción de direcciones se puede utilizar un banco de registros o un TLB.

Preguntas de autoevaluación tema 7

7.1. ¿Qué ventajas e inconvenientes presenta el uso de memoria virtual?

Permite aumentar el grado de multiprogramación del sistema, ya que en memoria principal caben más procesos si solo hay algunas partes de cada proceso que si sus espacios de direcciones virtuales estuviesen cargados por completo. Además permite ejecutar procesos con un espacio de direcciones virtuales de tamaño superior al tamaño de la memoria física disponible para procesos.
Otra ventaja es que reduce el número de operaciones de E/S que son necesarias para intercambiar un proceso bloqueado desde memoria principal a secundaria o viceversa.

Se debe diseñar con cuidado ya que de lo contrario el rendimiento del sistema puede verse seriamente afectado. Si un proceso está causando continuamente excepciones porque requiere cargar en memoria partes del proceso que no están cargadas, su ejecución se ralentiza enormemente y además estará provocando una fuerte sobrecarga al sistema que tendrá que ser atendido por continuas excepciones.

7.2. ¿Qué requisitos debe reunir el hardware de un computador para poder implementar la memoria virtual?

Que soporte el reinicio de las instrucciones de su repertorio. Además cuando se implementa la mv también se suele utilizar un componente hardware llamado unidad de gestión de memoria (MMU) que se encarga de realizar la traducción de una dirección virtual en una dirección física. Una MMU puede implementarse en un circuito integrado independiente o dentro del circuito integrado del procesador. Consta de registros, sumadores y comparadores. También puede tener un banco de registro o un TLB.

7.3. Describir brevemente el funcionamiento de la técnica de paginación por demanda.

Al igual que en la técnica de paginación simple divide el espacio de la memoria principal en bloques de igual tamaño

denominados marcos de página. Además divide el espacio de un proceso en bloques del mismo tamaño denominado páginas. En un determinado instante de tiempo un marco de página j puede estar libre u ocupado por una página i de un proceso X.

En la paginación por demanda, a diferencia de la simple, para que un proceso pueda ejecutar no es necesario que todas las páginas del mismo estén cargadas en memoria principal, únicamente se cargan las páginas que se van referenciando durante la ejecución de un proceso. Cuando se referencia a una página que no está cargada en la memoria principal el hardware produce una excepción denominada fallo de página. El SO se encarga de atender a los fallos de página. Básicamente el tratamiento de un fallo de página requiere que se busque un marco libre o se elija uno ocupado para ser reemplazado, y se copie en el mismo desde memoria secundaria la página que produjo el fallo de página.

7.4. ¿Qué tareas debe realizar el sistema operativo cuando usa la técnica de paginación por demanda?

- ***Asignación de marcos de memoria principal.*** En un sistema multiprogramado se suelen cargar simultáneamente en memoria dos o más procesos. El SO tiene que decidir cuántos marcos asigna inicialmente a cada uno.
- ***Control de carga.*** El SO debe controlar el grado de multiprogramación del sistema.
- ***Copia en la memoria secundaria de páginas modificadas.*** El SO debe decidir en qué momento y de qué forma copiará las páginas que han sido modificadas en la memoria principal y secundaria.

7.5. Enumerar y describir brevemente los campos que suele contener una entrada de la tabla de páginas de un proceso cuando se utiliza paginación por demanda.

Como mínimo, una entrada de una tabla de páginas asociada a la página i de un proceso, aparte del marco de página j donde se aloja la página, debe contener un bit denominado presente/ausente o validez/invalidez para indicar si la página está cargada en la memoria principal.

Cuando se crea un proceso y éste entra en el estado preparado para ejecución, el SO crea una tabla de páginas para dicho proceso que también reside en memoria principal. El tamaño de la tabla de páginas coincide con el tamaño del espacio virtual de

un proceso. La entrada i de la tabla se descompone en un número fijo de campos que contienen información sobre la página i del espacio de direcciones virtuales de un proceso. El número, la longitud y el nombre de estos campos depende de cada SO, pero generalmente se suelen encontrar al menos los siguientes campos.

- ***Número de marco de página.*** Contiene el número de marco j de memoria principal donde se encuentra almacenada la página i. El tamaño de este campo depende del número de marcos en que se descomponga la memoria principal.
- ***Validez o presencia.*** Consta de un único bit v que se activa (v =1) si la página i está cargada en el marco j, es decir, la entrada de la tabla se considera que es válida y puede ser utilizada. Si la página no pertenece actualmente al espacio de direcciones virtuales del proceso o la página no está cargada en memoria principal este bit está desactivado (v=0). En cada referencia a memoria hay que comprobar este bit. Si está desactivado entonces el hardware provoca un fallo de página.
- ***Protección.*** Este campo en su forma más simple consta de un único bit que en función de su valor indica si la página es de solo lectura o lectura-escritura.
  Usualmente suele contar de 3 bits (p2p1p0) lectura, escritura y ejecución respectivamente.
- ***Referenciada.*** Consta de un único bit que se activa cuando la página i es referenciada por una dirección virtual.
- ***Modificada.*** Consta de un único bit que se activa cuando una página es modificada en una operación de escritura.

  Los campos nº de marco de página, validez y protección son configurados por el SO y consultados por la MMU cada vez que hay que traducir una dirección virtual en una dirección física.

  Por su parte, los campos referenciada y modificada son activados por el hardware. Inicialmente cuando una página i de un proceso es cargada en memoria estos campos de la entrada i de la tabla de páginas del proceso son configurados a 0 por el SO. Además cada cierto tiempo durante la ejecución de un proceso el SO pone a 0 el campo referenciada de todas las paginas cargadas en memoria.

7.6. ¿Qué elementos hardware son necesarios para lograr el reinicio de instrucciones en la técnica de paginación por demanda?

Las arquitecturas que soportan memoria virtual por demanda de página disponen de un registro especial donde copian el valor del registro contador del programa antes de iniciar la fase de búsqueda de una instrucción.

7.7. En la técnica de paginación por demanda ¿qué eventos provocan que el sistema operativo tenga que acceder a la memoria secundaria para leer o escribir páginas de procesos?

Cuando la referencia a una página i produce un fallo de página, el SO debe leer dicha página en memoria secundaria y copiarla a la memoria principal. También, cuando se produce un fallo de página y no existen marcos libres en memoria principal el SO debe seleccionar una página para ser reemplazada. Si la página ha sido modificada o no existe una copia de la misma en el área de intercambio dicha página debe ser escrita allí.

7.8. ¿En qué localizaciones de memoria secundaria puede buscar el sistema operativa una página de un proceso?

Bloque de disco de un archivo ejecutable. Parte de las páginas en que se divide el espacio de direcciones virtuales de un proceso, en concreto las asociadas a las regiones de código y datos, son creadas por el SO a partir del archivo ejecutable del cual es instancia dicho proceso. Conviene aclarar que los bloques de disco que ocupa el archivo no contienen a las páginas ya formadas propiamente dichas sino que poseen el contenido que permite su creación.

Bloque de disco del área de intercambio. El área de intercambio se utiliza para el almacenamiento de las páginas de un proceso presentes en la memoria principal cuando éste es intercambiado fuera para hacer sitio a otros procesos. También se utiliza para el almacenamiento de las páginas que han sido seleccionadas para ser reemplazadas en memoria principal por otras durante el tratamiento de fallos de página.

7.9. Señala las ventajas e inconvenientes de que el sistema operativo copie por adelantado en el área de intercambio todas las páginas del espacio de direcciones virtuales de un proceso.

El objetivo de esta estrategia es evitar tener que acceder al sistema de archivos cuando se produzcan fallos de página y acelerar así su tratamiento.

Si la copia de la imagen del proceso ocupa un rango de direcciones contiguas del disco y además cada bloque del área de

intercambio contiene una página, entonces el SO únicamente necesita conocer la dirección de disco donde comienza la copia de la imagen del proceso en el área de intercambio. Dicha dirección se almacena en el bloque de control de proceso.

El problema es que el espacio de direcciones virtuales de un proceso puede aumentar su tamaño debido al crecimiento de la región de pila o la región de datos. Si esto sucede hay que buscar un hueco lo suficientemente grande en el área de intercambio y copiar allí de nuevo todo el espacio del proceso.

Otro problema es que consume bastante espacio del área de intercambio, ya que todas las páginas de un proceso son copiadas por adelantado en dicha área, independientemente de que se vaya a utilizar o no.

7.10. ¿Qué significa que un marco esté bloqueado o pinchado? Señalar una situación donde se realice el bloque de marcos.

Si su contenido no puede ser remplazado. El SO bloquea un marco de página activando un bit de bloqueo en la entrada asociada a dicho marco en la tabla de marcos de páginas. Los Nmp marcos en que se divide la memoria principal en un conjunto V marcos están reservados para alojar el espacio del núcleo. También se reservan marcos para alojar buffers de E/S. Los marcos reservados están bloqueados.

7.11. ¿Qué acciones realiza el sistema operativo para el tratamiento de un fallo de página?

Si el bit de validez está desactivado entonces el hardware produce una excepción denominada fallo de página y guarda en la pila del núcleo el valor del contador de programa y de los registros especiales. A continuación se cede el control al SO para que atienda el fallo de página. Cómo se realiza el tratamiento de un fallo de página depende de cada SO. Pero de forma general se puede decir que en primer lugar el SO guarda el contexto del proceso A cuya ejecución produjo la excepción. En segundo lugar comprueba la causa de la excepción, ya que existen diferentes motivos que pueden causar una excepción y la identifica como fallo de página.

A continuación intenta averiguar cuál fue la dirección virtual que produjo el fallo de página, consultando típicamente los registros ocultos del procesador. Si el procesador no dispone de dichos registros el SO tendrá que simular la ejecución de la instrucción que produjo el fallo hasta descubrirla.

Después de determinar la dirección virtual que produjo el fallo de página, el SO comprueba que se trata de una dirección legal, es decir, que pertenece al espacio de direcciones virtuales del proceso y que el tipo de acceso que se pretende no viola el tipo de acceso permitido según el campo de protección de su entrada asociada en la tabla de páginas. Si no es una dirección legal entonces enviará al proceso una señal de aviso o finalizará el proceso.

Si la dirección es legal, el SO consulta la lista de marcos de páginas libres. Si no hay ningún marco libre debe invocar al algoritmo de reemplazamiento de página para seleccionar un marco de página j, que no esté bloqueado, donde cargar la página i que produjo el fallo.

Si la página k que estaba contenida en el marco j tenía su bit de modificada activado, entonces la página k es planificada para ser copiada al disco duro. El SO pasa al proceso A cuya ejecución produjo fallo de página al estado bloqueado mientras se completa la copia de la página k al disco duro. También bloquea el marco j en la tabla de marcos de página para impedir que pueda ser utilizado para almacenar una página distinta de la página i. Además, se realiza un cambio de proceso, para permitir que otro proceso B se pueda ejecutar mientras tanto.

Una vez que el marco j está limpio, es decir, la página k se ha copiado en el disco, el SO localiza la ubicación de la página i en memoria secundaria y planifica su copia a memoria principal.

Cuando la página i ha sido copiada en memoria principal se produce una interrupción de disco para avisar de tal evento al SO. Éste actualiza las tablas de páginas y desbloquea el marco j en la tabla de marcos de página. A continuación el SO modifica el contador de programa y otros registros guardados en el contexto de proceso A para que cuando sea planificado de nuevo pueda reiniciarse justo desde el comienzo de la instrucción que produjo el fallo de página.

El SO despierta el proceso A, que pasa al estado preparado para ejecución. Cuando el proceso A sea planificado para ejecución se cargará su contexto y se reiniciará su ejecución desde el principio de la instrucción que produjo el fallo de página.

En el caso de que el SO hubiese encontrado un marco libre en la lista de marcos libres, entonces en la gestión del fallo se habrían ahorrado los pasos de invocar al algoritmo de reemplazamiento de página y la operación E/S para copiar la página k desde memoria principal al disco.

7.12. ¿Qué afirma el principio de localidad de referencias?

También denominado ***principio de proximidad*** afirma que durante un determinado intervalo de tiempo de ejecución de un proceso éste referencia a direcciones virtuales próximas entre sí, es decir, a páginas cercanas. Por tanto una vez que dichas páginas se han traído a memoria, el proceso se ejecutará durante dicho intervalo de tiempo sin necesidad de provocar fallos de página. Dicho conjunto de páginas forma una zona de localidad del proceso. Transcurrido dicho intervalo el proceso pasará a referenciar durante otro intervalo de tiempo a otro conjunto de páginas cercanas, es decir, a otra zona de localidad, y así sucesivamente.

¿Qué es una zona de localidad?

Conjunto de páginas cercanas.

7.13. ¿Qué es el conjunto de trabajo de un proceso?

Si el proceso se está ejecutando dentro de una zona de localidad, por el principio de proximidad, las referencias futuras serán parecidas a las referencias pasadas. Esta idea es la que subyace en el concepto de *conjunto de trabajo de un proceso,* que se puede definir para el instante Ti de tiempo virtual como el conjunto de páginas distintas que el proceso en ejecución ha referenciado en sus últimas referencias a memoria principal.

7.14. ¿Qué conclusiones se derivan del estudio del conjunto de trabajo de un proceso?

- ***Una página que es miembro del conjunto de trabajo de un proceso no debe ser seleccionada como página víctima para ser reemplazada, ya que ello provocaría rápidamente un nuevo fallo de página.***
- ***Un proceso debe tener asignados el suficiente número de marcos para poder tener cargado en memoria su conjunto de trabajo.*** En caso contrario estará constantemente provocando fallos de página, es decir, se produce el fenómeno conocido como sobrepaginación o trasiego.

7.15. ¿Qué es la sobrepaginación?

Un proceso debe tener asignados el suficiente número de marcos para poder tener cargado en memoria su conjunto de trabajo. En caso contrario estará constantemente provocando fallos de página, es decir, se produce el fenómeno conocido como sobrepaginación o trasiego.

7.16. Definir los siguientes términos: a) Página víctima. b) Conjunto de marcos candidatos.

De entre el conjunto de páginas candidatas, el algoritmo de reemplazamiento debe elegir una para ser reemplazada. A la página elegida se le denomina *Página víctima.*

En relación con el conjunto de páginas candidatas a ser reemplazadas, se define el término *conjunto de marcos candidatos,* que hace referencia a los marcos de memoria principal donde se encuentran almacenadas las páginas candidatas.

7.17. Señalar las diferencias entre las estrategias de reemplazamiento de páginas local y global.

El conjunto de páginas candidatas a ser reemplazadas puede estar formado únicamente por las páginas cargadas en memoria principal en marcos no bloqueados pertenecientes al proceso que ha producido fallo de página. En dicho caso se dice que el SO utiliza una estrategia de reemplazamiento de páginas local.

Otra posibilidad es que el conjunto de páginas candidatas a ser reemplazadas esté formado por todas las páginas cargadas en memoria principal ubicadas en marcos no bloqueados. En este caso se dice que el SO utiliza una estrategia de reemplazamiento de páginas global.

7.18. ¿Qué criterios se tienen en cuenta para establecer el rendimiento de un algoritmo de reemplazamiento?

El número de fallos de página que permite evitar y al tiempo de procesador que requiere su ejecución. Lo deseable es que el algoritmo de reemplazamiento que utilice el SO minimice el número de fallos de página y produzca una sobrecarga mínima. El problema es que estos dos requisitos son opuestos, por lo que hay que tratar de llegar a un compromiso entre ambos.

También es deseable que el algoritmo de reemplazamiento de páginas que se utilice disminuya el número de fallos de página a

medida que aumenta el número de marcos de página de la memoria principal. Es decir, cumpla el principio de que a más recursos mejor comportamiento.

7.19. ¿Qué es la anomalía de Belady?

Aquellos algoritmos de reemplazamiento que pueden aumentar el número de fallos de página si aumenta el número de marcos en memoria principal.

7.20. Describir el algoritmo de reemplazamiento óptimo, sus ventajas e inconvenientes.

Selecciona para ser reemplazada aquella página del conjunto de páginas candidatas a ser reemplazadas que tardará más en volver a ser referenciada por una dirección virtual. Minimiza el número de fallos de página pero no se puede implementar en la práctica ya que requiere conocer la cadena de direcciones virtuales futuras a las que hará referencia el procesador.

7.21. Describir el algoritmo de reemplazamiento LRU, sus ventajas e inconvenientes.

El algoritmo de reemplazamiento de la página usada menos recientemente o algoritmo de remplazamiento LRU (Least Recently Used) selecciona para ser reemplazada aquella página del conjunto de páginas candidatas a ser reemplazadas que lleva más tiempo sin ser referenciada, es decir, sin ser utilizada. La idea de este algoritmo es la de intentar predecir el futuro próximo teniendo en cuenta el pasado reciente.

Produce un número de fallos de página bastante cercano al algoritmo óptimo: Sin embargo, su principal inconveniente es que su implementación es compleja y produce bastante sobrecarga al sistema.

7.22. Describir dos posibles implementaciones del algoritmo de reemplazamiento LRU.

Una posible implementación es mediante el uso de lista enlazada. Cada entrada de la lista contiene el número de página de una página del conjunto de páginas candidatas a ser reemplazadas. Cada vez que una página es referenciada su número es colocado al principio de la lista. Es decir, la lista funciona como una pila. Cada vez que el procesador referencia a una dirección virtual, la lista debe ser ordenada, lo cual consume un tiempo importante.

Otras implementaciones requieren el apoyo de hardware. Por ejemplo, el algoritmo LRU se puede implementar utilizando un contador hardware que es incrementado cada vez que se referencia una dirección virtual. Además en cada entrada de una tabla de páginas hay que añadir un campo adicional para copiar el valor del contador, cada vez que una página es referenciada. Cuando se produce un fallo de página el SO examina el campo contador de cada entrada de la tabla de páginas y selecciona para ser reemplazada la página cuyo campo contador contiene el menor valor, ya que será la página que lleve más tiempo sin ser referenciada. Esta solución produce bastante sobrecarga, ya que en cada referencia a una dirección virtual hay que realizar un acceso a memoria principal para copiar el valor del contador en el campo de entrada correspondiente de la tabla de páginas. Además hay que realizar una búsqueda en todas las entradas de la tabla de páginas del campo contador de menor valor. Cuanto mayor sea el número de entradas más tiempo empleará la búsqueda.

7.23. Describir el algoritmo de reemplazamiento mediante envejecimiento. ¿Qué ventaja presenta este algoritmo frente al LRU?

Es una aproximación al algoritmo LRU que introduce una sobrecarga mucho más pequeña ya que se puede implementar de forma bastante eficiente.

Básicamente se asigna un registro de desplazamiento software de n bits a cada página cargada en memoria principal. El registro se inicializa a 0 cuando la página es cargada en memoria. Cada cierto tiempo T preestablecido, el SO desplaza un bit a la derecha el contenido del registro asignado a cada página cargando el bit referenciada en el bit más significativo de cada registro. Además pone a 0 el bit referenciada de cada página.

7.24. Describir el algoritmo de reemplazamiento PIFO, sus ventajas e inconvenientes.

Primero en entrar primero en salir selecciona para ser reemplazada aquella página del conjunto de páginas candidatas a ser reemplazadas que lleva más tiempo cargada en la memoria principal.

Para implementar este algoritmo el SO debe mantener una lista enlazada o cola FIFO. Cada entrada de la lista contiene el número de página de una página del conjunto de páginas

candidatas a ser reemplazadas. Cuando una página se carga en memoria su número de página se coloca al final de la lista. Así el número de página que se encuentra al principio de la lista indica la página que lleva más tiempo cargada en memoria y será la elegida para ser reemplazada. Cuando una página k que ocupaba el marco j es reemplazada por una página i entonces su número se elimina del principio de la lista y se coloca al final el número de la página que ha sido cargada en el marco j reemplazando a la página k.

Solo es necesario actualizar esta estructura de datos cuando se carga una página en un marco de memoria, por lo que su gestión requiere muy poco tiempo, además es sencillo de implementar y programar por lo que produce poca sobrecarga. Sin embargo, su rendimiento (número de fallos) no suele ser muy bueno. Además adolece de la anomalía de Belady.

7.25. Describir el algoritmo de reemplazamiento de la segunda oportunidad.

Es una variante del algoritmo FIFO que busca la página que lleva más tiempo cargada en memoria y no ha sido referenciada recientemente.

Básicamente este algoritmo consulta el bit referenciado (r) de la página que se encuentra al principio de la cola FIFO. Si r=0, entonces la página es seleccionada para ser reemplazada y el algoritmo finaliza. Si r=1 el algoritmo pone el bit a 0 y coloca el número de la página al final de la cola.

Tiene un mejor rendimiento en cuanto a la disminución del número de fallos de página que el algoritmo FIFO. Sin embargo produce mayor sobrecarga al estar moviendo páginas en la cola. Una forma de reducir esta sobrecarga es con una cola circular donde si r=1 este pasa a 0 y el puntero apunta a la siguiente número de página de la lista.

7.26. Describir el algoritmo de reemplazamiento del reloj mejorado.

El rendimiento del algoritmo del reloj puede mejorarse si además del bit referenciada (r) tiene en cuenta también el de modificada (m).

- R = 0 y m = 0. Son las candidatas ideales para ser reemplazadas.
- R = 0 y m = 1. Si se elige una página de este tipo para ser reemplazada, será necesario realizar una operación de

escritura en memoria secundaria para actualizar la copia de la página en el área de intercambio.

- R = 1 y m = 0. No son buenas candidatas para ser reemplazadas ya que puede que vuelvan a ser referenciadas próximamente.
- R = 1 y m = 1. No son buenas candidatas porque aparte de poder ser referenciadas próximamente deben ser salvadas en el área de intercambio.

Este algoritmo primero hace una ronda buscando R=0 y m=0, si no encuentra una segunda con 01, Si no encuentra una tercera con 00 y si no encuentra una última con 01.

7.27. Describir el algoritmo de reemplazamiento WSClock.

Para implementar WSClock es necesario que en cada entrada i de la tabla de página de un proceso se mantenga un campo denominado tiempo de último uso que contiene el tiempo virtual de último uso de página i. Si r= 1 entonces se escribe en el campo tiempo de último uso el tiempo virtual del proceso, es decir el tiempo que hace que la página no fue referenciada.

Si la edad de la página es menor o igual al tiempo de proceso entonces la página pertenece al conjunto de trabajo del proceso. Si no es una buena candidata a ser reemplazada.

En el caso 01 se planifica para ser escrita en el área de intercambio y el puntero pasa a la siguiente página.

El algoritmo WSClock busca páginas con r = 0, m = 0 y que no pertenezcan al conjunto de trabajo del proceso. Si durante una vuelta completa se da la situación de que todas las paginas pertenecen al conjunto de trabajo del proceso, entonces seleccionará la primera página que encuentre con r= 0 y m=0 aunque pertenezca al grupo de trabajo.

7.28. ¿Qué condiciones debería cumplir un buen algoritmo de reemplazamiento?

El mejor algoritmo realizable de reemplazamiento de páginas es aquél que produce una tasa de fallos cercana al producido por el algoritmo de reemplazo óptimo y además su implementación produce poca sobrecarga.

¿Cuáles son los algoritmos de reemplazamiento más utilizados en los sistemas operativos modernos?

Los más utilizados son el algoritmo de envejecimiento y WSClock ya sean en la forma descrita o con modificaciones.

WSClock solo se puede usar en local ya que se basa en el conjunto de trabajo de un determinado proceso.

7.29. ¿En qué consiste la tarea de asignación de memoria principal que realiza el sistema operativo para implementar la paginación por demanda?

Consiste en decidir cuántos marcos de página de la memoria principal reserva para un proceso que se tiene que ejecutar.

7.30. ¿Cuál es el número mínimo de marcos que es necesario asignar a un proceso?

Queda definido por el repertorio de instrucciones del computador. Se deben reservar como mínimo los marcos suficientes para contener todas las páginas a las que puede referenciar cualquier instrucción en el repertorio de la máquina.

7.31. Describir las dos posibles estrategias de asignación de memoria principal en paginación por demanda.

- ***Asignación fija***. Consiste en asignar a cada proceso un número prefijado de marcos. Su principal ventaja es que produce poca sobrecarga ya que el SO solo tiene que decidir cuántos marcos asigna cuando se va a cargar el proceso. Su principal inconveniente es que si el conjunto de trabajo del proceso crece por encima del número de marcos reservados, entonces se produciría sobrepaginación, aunque existieran marcos libres en memoria. Por otro lado, si el conjunto de trabajo disminuye por debajo del número de marcos reservados se estaría desperdiciando memoria. Si el SO utiliza una estrategia de asignación fija, entonces la estrategia de reemplazo de páginas tiene que ser local. Si se reemplaza una página, la página víctima se elige de entre el conjunto de marcos asignados al proceso.
- ***Asignación variable***. Consiste en ir variando a lo largo del tiempo de vida del proceso el número de marcos que tiene asignados. Su principal ventaja es su flexibilidad, lo que permite disminuir la sobrepaginación y la memoria desperdiciada. Además, permite que la política de reemplazo pueda ser local o global. El principal inconveniente de la asignación variable es que introduce una mayor sobrecarga

ya que el SO debe decidir continuamente cuantos marcos reserva a cada proceso.

7.32. Enumerar y describir brevemente los algoritmos de asignación de marcos más conocidos que se pueden emplear) en la asignación variable.

- *Asignación equitativa.* Cada cierto tiempo comprueba el número de procesos en ejecución y asigna a cada proceso el mismo número de marcos. Se trata de un algoritmo justo ya que todos los procesos son tratados por igual Sin embargo desperdicia memoria.
- ***Asignación proporcional***. Asigna a cada proceso un número de marcos en proporción a su tamaño o prioridad, o una combinación de ambos.
- ***Por frecuencia de fallos de página (PPF)***. Se basa en el principio que cuanto mayor sea el número de marcos asignados a un proceso menor será su tasa de fallos. Este principio lo cumplen todos aquellos algoritmos de reemplazamiento que no sufren la anomalía de Belady. El algoritmo mide la frecuencia de fallos de página en un proceso. Si la frecuencia de fallos de página se encuentra por encima de un valor límite prefijado, significa que la tasa de fallos es alta y asigna más marcos al proceso. Por otra parte, si la frecuencia está por debajo de un límite inferior prefijado se asignarán lo marcos de este proceso a otros que lo necesiten. En el caso de que varios procesos pasen los límites y no existan marcos libres se intercambiarán en el área de intercambio.

7.33. Representar gráficamente la influencia del grado de multiprogramación sobre el porcentaje de uso del procesador.

7.34. ¿Qué es la política de limpieza?

El SO debe decidir en qué momento y de qué forma copiará las páginas que han sido modificadas de la memoria principal a la memoria secundaria. A la estrategia o política seguida por el SO para realizar esta tarea se le conoce comúnmente como política de limpieza.

7.35. Señalar las principales ventajas e inconvenientes de las políticas de limpieza por demanda y por adelantado.

- ***Por demanda***. La página modificada se escribe en memoria cuando es seleccionada como página víctima para ser

reemplazada por otra página que ha producido un fallo de página. Con esta política se aplaza la escritura hasta que es estrictamente necesaria, con lo que se minimiza el número de operaciones de escritura. Sin embargo, cuando se produce un fallo de página hay que realizar dos operaciones de E/S, una de escritura de la página que se va a reemplazar y la otra de lectura de la página que ha producido el fallo de página, con lo que aumenta el tiempo que transcurre hasta que puede reiniciarse la ejecución del proceso que ha producido el fallo de página. El fallo de página se atendería más rápidamente si la página víctima ya hubiera sido copiada en memoria secundaria con anterioridad a que se produjese el fallo de página.

- ***Por adelantado.*** Las páginas que han sido modificadas se agrupan en lotes y se planifica cada cierto tiempo su escritura en memoria secundaria antes de que sean elegidas por el algoritmo de reemplazamiento. La principal ventaja es que permite atender los fallos de página más rápidamente al ahorrar, normalmente, la operación de escritura de la página víctima. Su principal inconveniente es que desperdicia tiempo de E/S, ay que una página que ha sido modificada una vez y que todavía no ha sido elegida como página víctima, puede volver a ser modificada en un futuro próximo por lo que tendrá que ser copiada varias veces. Una posible solución a las escrituras innecesarias consiste en seleccionar para ser copiadas solamente aquellas páginas modificadas que son candidatas a ser reemplazadas.

7.36. ¿Qué factores hay que tener en cuenta a la hora de seleccionar el tamaño de página?

- ***Fragmentación interna.*** En promedio la fragmentación interna es de media página por proceso. En consecuencia, cuanto mayor sea el tamaño de la página mayor será la fragmentación interna y mayor será el desperdicio de espacio de la memoria principal.
- ***Tamaño de una tabla de páginas.*** Cada proceso tiene una tabla de páginas cargada en memoria principal que posee un número de entradas igual al número de páginas en que se descompone el espacio de direcciones virtuales del proceso. Cuanto menor sea el tamaño de una página, en más páginas se descompondrá el espacio virtual de proceso, y las tablas de páginas serán de mayor tamaño. Por lo tanto, consumirán más espacio de memoria. Además, si el tamaño de la tabla de

páginas aumenta, el tiempo de cambio de proceso será mayor ya que el SO tendrá que cargar o vaciar más registros si se tiene una MMU con banco de registro o con TLB.

- ***Número de fallos de página.*** Si el tamaño de página es pequeño el espacio de direcciones virtual constará de más páginas por lo que la probabilidad de que se produzca un fallo de página será mayor.
- ***Tiempo de uso de E/S***. El tiempo de una lectura o escritura en un disco se descompone en tiempo de búsqueda, tiempo de latencia rotacional y tiempo de transferencia. De estos tres tiempos el tiempo de transferencia es mucho más pequeño que los tiempos de búsqueda y de latencia. Por ello se tarda menos tiempo en transferir una página de tamaño grande que en transferir varias páginas de tamaño más pequeño. Por otra parte las páginas se escriben o se leen en memoria secundara una a una. Si el tamaño de página es pequeño, el espacio de direcciones virtuales del proceso se descompondrá en un mayor número de páginas, por lo que la probabilidad de realizar operaciones de E/S será mayor, tiempo de ES que no puede ser utilizado por la ejecución de los procesos.

7.37. ¿Cuáles son las ventajas e inconvenientes de tener un tamaño de página grande?

Disminuye el tamaño de las tablas de páginas, el número de fallos de página y el tiempo de uso de la E/S. Sin embargo aumenta la fragmentación interna.

7.38. ¿En qué consiste la estrategia conocida como paginación por adelantado? ¿Cuándo se considera que esta estrategia es efectiva?

En un SO con memoria virtual mediante demanda de página cuando comienza la ejecución de un proceso o se continúa con la ejecución de un proceso que estaba bloqueado o intercambiado, se producen un conjunto de fallos de página iniciales hasta que el proceso tiene cargadas las páginas de su conjunto de trabajo.

Para evitar este conjunto de fallos iniciales se puede usar la estrategia conocida como paginación por adelantado (prepaging) que consiste en cargar un cierto número de páginas asociadas a un proceso antes de iniciar o continuar con su ejecución. También se puede utilizar la paginación por adelantado para cada vez que se

produce un fallo de página, con el objetivo de evitar futuros fallos de página.

El punto clave de esta estrategia radica en cargar las páginas adecuadas para evitar los fallos de página, ya que si se cargan varias páginas que no van a ser referenciadas se estaría perdiendo tiempo en cargar dichas páginas y no se evitarían fallos.

En general si se cargan Npa páginas por adelantado, se referenciarán Npr páginas y no se referenciarán Npa-Npr páginas. Cada página referenciada cargada por adelantado evita un fallo de página. La paginación por adelantado se considera que es efectiva si el tiempo ahorrado en tratar Npr fallos de páginas es superior al tiempo empleado en cargar Npa – Npr que no son referenciadas. En caso contrario no compensa utilizar paginación por adelantado.

7.39. ¿Qué opciones se emplean en relación a la selección de páginas que van a ser cargadas por adelantado?

- ***Cargar el conjunto de trabajo del proceso***, que debe ser mantenido por el SO. Esta opción se puede emplear cuando se va a continuar con la ejecución un proceso que fue intercambiado o bloqueado.
- ***Cargar un conjunto de páginas ubicadas de forma contigua en el área de intercambio,*** ya que el tiempo de una operación de E/S para traer a memoria desde el disco Npa páginas contiguas es más pequeño que el tiempo empleado en realizar Npa operaciones de E/S para traerlas una a una cuando sean referenciadas.

7.40. ¿En qué consiste la reserva de marcos libres y qué ventajas ofrece?

El tiempo de tratamiento de un fallo de página se reduce si se dispone de un marco libre de memoria principal donde cargar la página referenciada por la dirección virtual que ha producido el fallo de página. Por este motivo el SO puede mantener una reserva de marcos libres implementada en la forma de una lista de marcos libres. En cada entrada de la lista de marcos libres se mantiene el número de marco j y la pagina i que contiene actualmente. Nótese que un marco j calificado como libre puede o bien estar vacío o contener una página i que es reemplazable.

Si la reserva de marcos libre cae por debajo de un cierto límite se utiliza el algoritmo de reemplazamiento para seleccionar páginas reemplazables, cuyos marcos se agregan a la lista hasta alcanzar cierto límite máximo.

Cuando se necesita un marco libre se coge el situado en la cabecera de la lista de marcos libres. Por otro lado, si se selecciona alguna página i como reemplazable su marco j es colocado al final de la lista si la página no ha sido modificada. Además el campo válido de la entrada i de la tabla de páginas correspondiente se desactiva. Si la página ha sido modificada, entonces ingresa en una lista de páginas modificadas que serán escritas por lotes en la memoria secundaria. Una vez escritas pasan a ser colocadas al final de la lista de marcos libres.

Mientras la página i siga cargada en el marco j puede ser utilizada. Por este motivo cuando se produce un fallo de página se comprueba la lista de marcos libres por si alguno de ellos contuviera la página que ha producido el fallo, y de este modo ahorrarse la lectura de la página en memoria secundaria. Así, la lista de marcos libres se utiliza como una memoria caché software de páginas.

Preguntas de autoevaluación tema 8

8.1. Explicar qué es un dispositivo modo bloque y un dispositivo modo carácter. Señalar algunos ejemplos de cada tipo.

Modo bloque almacena información en bloques de tamaño fijo, normalmente 512 bytes o una potencia de dos múltiplos de la anterior, cada uno de los cuales tiene asignado un determinado número identificador o dirección. Ejemplos de dispositivos modo bloque son los discos y unidades de cinta.

Modo carácter envía o recibe información como una secuencia o flujo lineal de bytes, en ellos la información no se organiza con una estructura concreta por lo que no es direccionable, y en consecuencia no permite la realización de operaciones de búsqueda.

8.2. ¿Qué información debe pasar un proceso a la llamada al sistema que invoque para solicitar una petición de E/S?

La dirección lógica del dispositivo de E/S sobre la que desea operar (usualmente especificada mediante una ruta simbólica que puede ser traducida por el subsistema de E/S), la dirección lógica del espacio del proceso donde se almacenarán los datos que se lean en el dispositivo (o donde se encuentran los datos que se escribirán en el dispositivo) y el número de bytes que se van a transferir.

8.3. Señalar dos ejemplos donde el propio sistema operativo realiza operaciones de E/S durante la realización de alguna de sus tareas administrativas.

Dar formato a una cadena de caracteres para que sea mostrada en la pantalla de una determinada forma, o interpretar una cadena de caracteres introducida por teclado para asociarla a las variables del proceso adecuadas.

8.4. ¿Cuáles son las capas de software de E/S del núcleo de un sistema operativo?

Subsistema de E/S, drivers de los dispositivos de E/S y manejadores de las interrupciones.

8.5. ¿Qué función tiene el subsistema de E/S?

Es el componente del SO que se encarga de efectuar todas aquellas tareas necesarias para la realización de las operaciones de E/S que son comunes a todos los dispositivos independientes de los mismos. Es decir, gestiona la parte independiente del dispositivo de todas las operaciones de E/S.

8.6. Enumerar las tareas del subsistema de E/S.

- ***Asignación y liberación de dispositivos dedicados*.** Algunos dispositivos E/S, como por ejemplo un disco, pueden ser utilizados simultáneamente por varios procesos. No existe ningún inconveniente en que varios procesos tengan abiertos diferentes archivos de un disco. Sin embargo, otros dispositivos, como una unidad de cinta magnética o una impresora, solo puede ser utilizados simultáneamente por un único proceso. A estos últimos dispositivos se les conoce como dispositivos dedicados. Es responsabilidad del subsistema de E/S determinar si una petición de E/S de un proceso A sobre un dispositivo dedicado puede ser aceptada o debe ser rechazada. Si la petición es aceptada, pero el dispositivo se encuentra ocupado atendiendo la petición de

otro proceso B, el subsistema de E/S puede bloquear al proceso A y colocarle en una cola de procesos bloqueados en espera del uso del dispositivo. Cuando el dispositivo se encuentre disponible el primer proceso de la cola será desbloqueado y se le asignará el uso del dispositivo. De esta forma, el proceso A obtendrá cuando le llegue su turno el uso del dispositivo.

- ***Bloqueo de procesos que solicitan una operación de E/S.*** Cuando al subsistema de E/S le llega, a través de la interfaz de llamadas al sistema, una petición de un proceso para la realización de una operación de E/S, éste puede pasar o no al estado bloqueado al proceso que solicitó la operación de E/S. La decisión de bloquear un proceso dependerá de si el proceso invocó una llamada al sistema de tipo bloqueante o del tipo y estado del dispositivo involucrado en la operación de E/S solicitada.
- ***Planificación de la E/S.***
- ***Invocación del driver de dispositivo apropiado.***
- ***Almacenamiento temporal de datos de E/S o buffering.***
- ***Proporcionar un tamaño de bloque uniforme a los niveles superiores del software.***
- ***Gestión de los errores producidos en una operación de E/S.***

8.7. ¿Qué es un driver de dispositivo?

Contiene el código que permite a un SO controlar un determinado tipo de dispositivo de E/S. Un driver se diseña teniendo en cuenta las especificaciones de la interfaz de drivers del subsistema de E/S de cada SO y las características del dispositivo.

8.8. ¿Qué significa y qué ventajas presenta el hecho de que la interfaz para los drivers de los dispositivos de un subsistema de E/S sea uniforme?

Que las funciones que debe aportar el driver y las funciones del núcleo que puede invocar son las mismas para todos los drivers. Si la interfaz no fuera uniforme habría que modificar el código del SO para adaptar el subsistema de E/S a cada nuevo driver. Además el subsistema de E/S tendría la necesidad de conocer qué funciones puede ejecutar cada driver, ya que todos no pueden ejecutar las mismas.

8.9. ¿Con qué elementos interactúa el driver de un dispositivo?

Un driver de dispositivo interactúa con el subsistema de E/S y con el controlador de E/S que controla el dispositivo. Un driver suministra al subsistema de E/S el conjunto de funciones que se pueden realizar sobre el dispositivo, tales como lectura o escritura. Además un driver puede invocar a ciertas rutinas o procedimientos del núcleo. Los procedimientos a los que tiene acceso un driver quedan definidos por la interfaz de drivers del subsistema de E/S.

8.10. ¿Qué acciones realiza un driver?

- ***Comprobar que los parámetros de la función invocada por el subsistema de E/S son correctos y que la operación de E/S solicitada se puede realizar.*** En caso contrario devuelve un error.
- ***Traducir los parámetros de dicha función en parámetros específicos del dispositivo.***
- ***Comprobar si el dispositivo de E/S está ocupado atendiendo alguna petición anterior de E/S. Si el dispositivo está ocupado entonces coloca la petición en una cola***. Si el dispositivo no está atendiendo ninguna petición, comprueba si se encuentra preparado, ay que quizás el driver deba activar e inicializar el dispositivo.
- ***Generar un conjunto de órdenes para el controlador del dispositivo dependiendo de la petición de E/S solicitada por el subsistema de E/S.*** Dichas órdenes son cargadas en los registros del controlador. El driver puede comprobar que el controlador acepta una orden y que se encuentra listo para aceptar la siguiente.
- ***Una vez transmitidas todas las órdenes al controlador, el driver debe esperar a que el controlador las ejecute.*** Si el tiempo de espera estimado es importante, entonces el driver se bloquea usando algún mecanismo de sincronización hasta que el controlador finalice. Cuando la operación de E/S se complete el controlador activará una interrupción. El manejador o rutina de servicio de dicha interrupción despertará al driver, si éste se bloqueó.
- ***Comprobar que no se han producido errores en la operación de E/S***. En dicho caso quizás transfiera al subsistema de E/S el resultado de la operación de E/S, por ejemplo, un bloque leído en el disco, o puede que simplemente le informe de que la operación se ha completado. Si se ha producido algún error y el driver sabe cómo resolverlo realiza la acción correspondiente. Si no sabe cómo resolverlo o la solución que plantea no surte efecto,

entonces informa del error al subsistema de E/S para que tome las medidas correspondientes oportunas.

- ***Examinar la cola de peticiones de E/S pendientes, si existe alguna procede a atenderla***. Si la cola está vacía el driver se bloqueará en espera de la llegada de nuevas peticiones.

8.11. Explicar las características y el funcionamiento de los manejadores de interrupciones.

Cuando finaliza una operación de E/S en un dispositivo y éste se encuentra preparado para procesar otra, el controlador de E/S que lo supervisa genera una interrupción. En un computador que implemente un sistema de interrupciones vectorizadas cada interrupción suministra un número denominado número del vector de interrupción que se utiliza como índice en una tabla, denominada tabla de vectores de interrupción. Esta tabla usualmente se encuentra almacenada en las posiciones más bajas de memoria. Cada entrada de esta tabla se denomina vector de interrupción, que entre otras informaciones contiene la dirección de comienzo del manejador (handler) de la interrupción.

Los manejadores de interrupciones forman parte del núcleo del SO y son extremadamente dependientes del hardware. Por este motivo, a la hora de portar un SO a una nueva arquitectura, esta parte del núcleo siempre tiene que ser reescrita.

Con objeto de que el rendimiento del computador sea óptimo, los manipuladores de interrupciones tienen una alta prioridad de ejecución. Además su código suele ser pequeño y rápido de ejecutar. Las acciones específicas que realiza un manejador de interrupciones dependen de cada tipo de interrupción.

Si el driver del dispositivo se bloqueó en espera de que el controlador de E/S estuviera preparado para procesar otra petición de E/S, entonces una acción que debe realizar un manejador de interrupción es el desbloqueo del driver del dispositivo mediante el uso del mismo mecanismo de sincronización (semáforo, mensajes, …) que utilizó el driver para bloquearse.

En el caso en que no se realice DMA un manejador de una interrupción también se puede encargar de transferir datos desde un registro del controlador del dispositivo a un buffer en el espacio del núcleo, o viceversa, en función de si se trata de una operación

de lectura o de escritura, respectivamente. Esta función, en ocasiones, la puede realizar el propio driver en lugar del manejador.

8.12. ¿Qué es el buffering? ¿Qué problemas resuelve?

En la transferencia directa una transferencia se almacenaría el bloque directamente en una zona contigua de la región de datos del espacio de direcciones virtuales del proceso. En un sistema con paginación dicha zona formará parte de una página i del proceso.

Esta forma de realizar la trasferencia de datos obliga al SO a mantener cargada en un marco j de la memoria principal la página i mientras no se complete la operación de E/S, ya que en caso contrario se perderían los datos. Las operaciones E/S son lentas en comparación con los accesos a memoria principal, por lo que el marco quedará bloqueado un tiempo importante. Cuanto mayor sea el número de marcos bloqueados, menor será el número de marcos candidatos a ser reemplazados al atender un fallo de página lo que produce sobrepaginación.

Otro problema es la imposibilidad de poder intercambiar al proceso hasta que la operación E/S no se haya terminado.

Para evitar los problemas que plantea la transferencia directa de datos desde el dispositivo al espacio de usuario del proceso los SO implementan la técnica conocida como buffering, que consiste en almacenar temporalmente en buffers los datos que se transfieren entre un dispositivo y un proceso, o viceversa. Un buffer es un área de memoria principal a la que únicamente tiene acceso el SO, es decir, pertenece al espacio del núcleo.
El subsistema E/S es el encargado de asignar buffers para las operaciones de E/S. Por su parte, las rutinas de servicio de las interrupciones o los drivers, se encargan de transferir los datos entre los buffers y los dispositivos, o viceversa, dependiendo de si se trata de una operación de entrada o salida.

8.13. Enumerar y explicar brevemente las diferentes estrategias para la realización del buffering.

- ***Buffering con buffer único.*** Los datos son transferidos primero desde el dispositivo al buffer ubicado en el espacio del núcleo. Posteriormente los datos se transfieren desde el buffer al espacio del proceso. En una operación de escritura, los datos se transfieren primero desde el espacio del proceso al buffer en el espacio del núcleo.

- ***Buffering con dos buffers o buffering doble.*** En una operación de E/S se asignan dos buffers. Cuando el primer buffer se llena, se pasa a llenar el segundo mientras se vacía el primero. Si cuando el segundo está lleno se puede comenzar a llenar el primero de nuevo.
- ***Buffering circular.*** Consiste en utilizar más de dos buffers para una operación E/S. Primero se llena el primero, luego el segundo, el tercero, etc… Cuando se termina de llenar el primer buffer se pueden comenzar a vaciar. Si se termina de llenar el último buffer del conjunto y el primero ya ha sido vaciado se puede comenzar otro ciclo.

8.14. ¿En qué consiste una caché de buffers de bloques de disco?

Se trata de un área de memoria principal del espacio del núcleo reservada para almacenar buffers. Dichos buffers contienen los bloques de disco que han sido recientemente transferidos.

El funcionamiento es similar al de una caché hardware. Cuando un proceso solicita la realización de una operación de lectura en disco, el SO comprueba si los datos solicitados se encuentran almacenados en la caché de buffers. En caso afirmativo, sirve los datos al proceso. Si los datos solicitados no están en la caché de buffers, entonces es necesario realizar una operación E/S. Cuando el subsistema de E/S asigna buffers para una operación de E/S en disco debe comprobar si existen buffers libres en la caché de buffers. Si no es así, tendrá que utilizar algún algoritmo de reemplazamiento, como los comentados al estudiar paginación (FIFO, LRU, etc), ara seleccionar los buffers cuyo contenido va a ser reemplazado.

8.15. ¿Qué es el spooling y cómo se implementa?

En sistemas multiprogramados el SO se debe encargar de asignar y controlar el uso de los dispositivos E/S. En el caso de los dedicados como la impresora se utiliza la técnica de control llamada spooling. Se implementa con un proceso demonio y con un directorio especial. El proceso spooling es el único autorizado para escribir en el dispositivo. Si un proceso de usuario quiere escribir en el dispositivo se deben enviar los archivos al directorio de spooling.

8.16. ¿En qué casos se usa la técnica de spooling?

En el control de la impresora o en la transferencia de paquetes de datos de una red.

8.17. Describe las partes que componen un reloj programable y su funcionamiento.

Entre otros elementos un contador, un registro y un cristal de cuarzo.

8.18. ¿Qué es un tic de reloj?

Es el tiempo entre dos señales periódicas.

8.19. Enumerar y explicar brevemente las tareas del sistema operativo en las que utiliza relojes programables.

- ***Planificación de procesos.*** Si el SO utiliza el algoritmo de turno rotatorio para la planificación de procesos debe vigilar que el proceso en ejecución no exceda el cuanto de uso de CPU asignado. Para ello carga en un reloj programable el valor del cuanto y en cada tic de reloj lo va decrementando. Cuando llega a 0, significa que se ha terminado el cuanto, entonces despierta al planificador para que planifique otro proceso.
- ***Mantenimiento del tiempo real.*** Para mantener el tiempo real es necesario fijar un punto de referencia temporal inicial a partir del cual se va llevando la cuenta.
- ***Disparo de alarmas.*** Los procesos pueden solicitar al SO mediante alguna llamada al sistema que éste avise cuando haya transcurrido un determinado intervalo de tiempo. Una forma de implementar una alarma sería usar un reloj programable que fuese decrementando. El SO suele gestionar las alarmas manteniendo una lista de alarmas ya que si el número de alarmas es superior al número de relojes programables disponibles sería un error.
- ***Invocación de tareas periódicas del sistema.***

8.20. Explicar la interacción entre el sistema operativo y el reloj del computador.)

El SO utiliza el driver del reloj para interactuar con el reloj del computador. Cuando el reloj genera una interrupción, el manipulador de la interrupción desbloquea el driver y éste debe realizar diferentes tareas, como por ejemplo: decrementar el cuanto del proceso en ejecución, incrementar el tiempo de uso del procesador de dicho proceso, incrementar el reloj de tiempo real y decrementar el tiempo de disparo en la lista de alarmas.

8.21. ¿En qué consiste el formateo a bajo nivelo formateo físico de un disco duro?

Consiste en dividir cada superficie de los platos en pistas y cada pista en sectores.

8.22. Describir los elementos en qué se descompone un sector de un disco duro.

- ***Cabecera o preámbulo.*** Contiene entre otras informaciones un cierto patrón de bits que permite reconocer al controlador del disco el comienzo del sector, el número de sector y el número de cilindro.
- ***Área de datos*****.** Almacena la información accesible para el SO y usuarios, su tamaño típico suele ser 512 bytes.
- ***Código de corrección de errores*****.** Suele ocupar 16 bytes, contiene información que permite al controlador del disco detectar y corregir posibles errores de lectura.

8.23. ¿En qué consiste el método de acceso LBA?

Direccionamiento de bloques lógicos (logic block addresing). Con esta técnica un disco duro se considera un array de bloques lógicos de datos. Cada bloque tiene asignado un número de bloque identificativo o dirección lógica. El 0 suele ser el de la primera pista del cilindro más externo.

El driver del disco solo debe especificar al controlador del disco el número de bloque lógico que hay que leer o escribir. El controlador del disco se encarga de traducir dicho número en una tripleta CHS. En los discos que no soportan LBA, el driver tiene que pasarle en cada operación de E/S una tripleta CHS al controlador del disco.

8.24. ¿En qué consiste el particionamiento de un disco duro?

Consiste en establecer por software una o más particiones en el disco. Una partición es un conjunto de cilindros contiguos. Desde el punto de vista lógico cada partición se considera como un disco distinto.

La información sobre el cilindro de comienzo, el tamaño de cada partición y la partición activa se mantiene en el propio disco en una estructura denominada tabla de particiones que se localiza al final del registro de arranque maestro (MBR) dentro del sector 0 del disco.

8.25. ¿En qué consiste el formateo a alto nivel lógico de un disco duro?

También llamado formateo lógico, consiste en establecer el bloque o sector de arranque y las estructuras de datos de un sistema de archivos (lista o mapas de espacio libre y asignado, y un directorio raíz o inicial vacío). También coloca en la tabla de particiones información sobre el tipo de sistema de archivos colocado en cada partición.

8.26. ¿Bajo qué condiciones el algoritmo de planificación de peticiones de E/S a disco se implementen en el driver del disco? (Respuesta en sección 8.6.2)

Por software en el driver del disco, ya que es la capa de software que conoce los detalles de la estructura interna del disco. Por otro lado, si el controlador del disco puede aceptar un lote de peticiones, el algoritmo de planificación se puede implementar dentro del propio controlador del disco.

En los HD que soportan LBA el algoritmo de planificación del disco se implementa forzosamente en el controlador del disco ya que es el único que conoce los detalles de organización física del disco. Recuérdese que en dicho caso el driver del disco trabaja exclusivamente con direcciones o número de bloques lógicos. Con dicha información no es posible implementar los algoritmos SSTF (shortest search time first) ni LOOK, los cuales necesitaban conocer el número de cilindro de cada petición de E/S al disco.

8.27. Señalar dos técnicas para realizar el reemplazamiento de un sector defectuoso en un disco duro ¿En qué caso el sistema operativo se debe encargar de realizar esta tarea?

- ***Sustitución directa.*** Supóngase que el sector j de una pista está dañado, entonces este método consiste en usar un sector de reserva como sector j. El principal atractivo de esta estrategia es que se realiza bastante rápido, simplemente hay que posicionar la cabeza de lectura-escritura sobre el sector de reserva y escribir la cabecera del sector defectuoso al que sustituye. Sin embargo, puede alterar la organización del entrelazado del disco, empeorando el rendimiento del disco.
- ***Desplazamiento de sectores.*** Consiste en desplazar una posición cada sector situado entre el sector de reserva y el sector defectuoso. El sector de reserva se utiliza para contener al primer sector que se desplaza. La posición ocupada por el último sector desplazado es utilizada para

albergar al número de sector que estaba defectuoso. Esta estrategia presenta como ventaja que mantiene la organización de una pista por lo que no repercute en el rendimiento del disco. Sin embargo, se trata de una operación lenta, ya que múltiples sectores de una pista tienen que ser reescritos de nuevo.

8.28. Explicar la gestión de la salida por pantalla.

La salida es una red de puntos denominados píxels. Cada píxel tiene asociado un número binario que indica el color con que se representa en pantalla. Por ejemplo, se pueden utilizar un número de 24 bits para representar cada pixel, dedicando 8 bits para cada intensidad de rojo, verde y azul. Para evitar el parpadeo se debe mostrar una imagen de 60 a 100 veces por segundo.

La pantalla se suele conectar a través de un adaptador o controlador gráfico que puede estar integrado o en una tarjeta sobre la placa. Tiene una memoria de video (VRAM) que es una memoria RAM en la que se almacena los pixeles de la imagen que se mostrará en la pantalla.

La imagen que se carga en la VRAM es generada por paquetes de rutinas de creación y manipulación de gráficos independientes del hardware, las cuales son invocadas por las aplicaciones o por el gestor de ventanas de un GUI. Estos paquetes pueden formar parte del núcleo del SO (Windows) o ser externos al mismo (UNIX y Linux). En el primer caso, dichas rutinas se ejecutarán en modo núcleo, en el segundo caso en modo usuario. Para interactuar con el adaptador gráfico las propias rutinas (Windows) o el subsistema de E/S (Unix y Linux) invocan al driver del adaptador, que es el que conoce los detalles del hardware.

8.29. Explicar la gestión de la entrada del teclado.

Es un dispositivo modo carácter de entrada. Cuando el usuario pulsa o suelta una tecla el teclado envía el número de tecla a un registro del controlador E/S que lo supervisa. Éste genera una interrupción que al ser atendida por el manejador correspondiente provoca que se despierte al driver del teclado.

El driver extrae del registro de E/S el número de tecla y lo traduce en un carácter usando unos mapas de teclas, también llamados páginas de códigos, que mantiene el SO en función del idioma que haya elegido el usuario al instalar el SO. Suele tener un

tamaño de 8 bits, el más significativo se utiliza para indicar si la tecla ha sido presionada o no.

El driver también tiene que llevar la cuenta de las teclas que han sido presionadas y siguen presionadas.

Una vez establecido el carácter o la combinación de teclas pulsadas el driver almacena dicha información en un buffer para que sea leída y procesada por el proceso adecuado.

8.30. Explicar la gestión de la entrada del ratón.

Cada vez que el usuario desplaza el ratón o pulsa un botón del mismo se transmite un mensaje al registro del controlador de E/S que lo supervisa. Suele ser de 3 bytes y contiene la siguiente información: eje “x”, eje “y” y estado (pulsado o sin pulsar) en los diferentes botones del ratón. Los desplazamientos son relativos, es decir, se miden en relación a la posición del ratón indicada en el último mensaje enviado.

Cada vez que recibe un mensaje, el controlador de E/S que supervisa al ratón genera una interrupción, el manejador de la interrupción despierta el driver del ratón que lee el mensaje y lo coloca en una cola de mensajes para que sea leído y procesado por el proceso adecuado.

Preguntas de autoevaluación tema 9

9.1. ¿Qué es el subsistema de gestión de archivos?

El SO es el responsable de la gestión de los archivos de diferentes sistemas de archivos existentes en la memoria secundaria. Además debe fijar los tipos, atributos, la estructura interna y los mecanismos de acceso que soporta. También debe definir la estructura de los directorios en la que los usuarios y aplicaciones organizan sus archivos, así como mecanismos de búsqueda de archivos y directorios. Por otra parte el SO se encarga de asignar espacio a los archivos y de administrar el espacio libre en la memoria secundaria.

Al componente del SO que realiza todas estas tareas se le denomina subsistema de gestión de archivos.

9.2. ¿Qué es un archivo?

Un archivo es un conjunto de información relacionada que se almacena en memoria secundaria y que se identifica mediante un nombre, que es una cadena de caracteres.

¿Y un directorio?

Un directorio es un archivo que almacena una lista de los archivos y subdirectorios que contiene.

9.3. ¿Qué diferencia hay entre un archivo binario y un archivo ASCII?

Binario es aquél que contiene información de cualquier tipo codificada en binario con una estructura determinada que únicamente puede ser interpretada por los programas que lo utilizan. Ejemplos son archivos ejecutables que contienen código compilado y datos. La estructura de estos archivos solo puede ser interpretada por el programa cargador encargado de cargar el código y los datos en memoria principal. Algunos binarios contienen una cabecera con metadatos que ayudan a interpretar correctamente la información que contiene.

Por su parte, un archivo ASCII es aquél que está compuesto de líneas de caracteres ASCII codificadas en binario que no requieren un programa que los interprete. La impresión de un archivo ASCII genera una salida comprensible: Las líneas de caracteres ASCII que lo componen.

9.4. Enumerar los atributos de un archivo más frecuentemente mantenidos por un sistema operativo.

- ***Tipo de archivo.*** Compuesto por uno o varios bits informa al SO sobre el tipo de archivo.
- ***Tamaño.***
- ***Localización.***
- ***Creador y propietario.***
- ***Permisos de acceso.*** Lectura, escritura, ejecución.
- ***Información asociada al tiempo.***

9.5. Enumerar y explicar brevemente las posibles formas en que se puede estructurar la información contenida en un archivo.

- ***Secuencia de bytes.*** Se estructura como un conjunto de bytes contiguos. Las operaciones de lectura y escritura se realizan a nivel de byte. UNIX o Windows estructuran el contenido de esta forma debido a su flexibilidad. El SO no tiene que interpretar la información contenida en cada byte, lo

tienen que hacer los programas de aplicación a nivel usuario como procesadores de textos o compiladores.

- ***Secuencia de registros.*** El archivo se estructura como un conjunto de registros contiguos de igual longitud. Cada registro posee a su vez una cierta estructura interna. Las operaciones de lectura o escritura se realiza a nivel de registro.
- ***Registros indexados.*** Conjunto de registros de longitud variable. Cada registro contiene un campo clave o índice que permite identificarlo. El archivo se organiza en función de la clave de los registros que lo componen. Para realizar una operación de lectura o escritura se debe especificar la clave del registro que se desea leer o escribir. Si se añade un nuevo registro al archivo, se debe especificar su clave. Es el SO el que se encarga de ubicarlo dentro del conjunto de registros. Esta forma de estructurar la información de un archivo es usada en algunos SO de macro computadores usados para gestión de datos de transacciones comerciales.

9.6. Enumerar y explicar brevemente los diferentes métodos para acceder a la información contenida en un archivo.

- ***Acceso secuencial.*** Los bytes o registros que componen el archivo se leen o se escriben en orden secuencial El SO debe mantener un puntero de R/W que marca la posición (byte o registro) donde la próxima operación de lectura o escritura debe comenzar. Cada vez que se lee o se escribe un byte o registro se incrementa el puntero. Es posible volver al principio del archivo para leerlo o escribirlo tantas veces como sea necesario o ir al final para escribir nueva información sin sobrescribir la ya existente. Es el método de acceso más sencillo. Compiladores y editores de texto.
- ***Acceso aleatorio.*** También conocido como acceso directo. Los bytes o registros pueden ser leídos o escritos en cualquier orden. Es de gran utilidad cuando se tiene una gran cantidad de información, como en BBDD y se desea acceder a una información en particular. Las operaciones de R/W se pueden acceder de dos formas: especificando el número del byte o registro al que se desea acceder como parámetro de la operación de R/W o usar primero una operación adicional de posicionamiento o búsqueda para ubicarse al comienzo del byte o registro que se desea leer. Las operaciones de R/W se realizarán secuencialmente comenzando desde dicha

posición. De esta forma además se puede simular acceso secuencial.

9.7. ¿En qué consiste la estructura de directorios de dos niveles?

Consiste en un directorio raíz (maestro), que almacena una lista de directorios, uno por usuario. Todos los archivos de un usuario se encuentran referenciados por la lista almacenada en su directorio. Es útil si se trabaja independientemente pero el inconveniente de si se hace en colaboración. Los archivos de sistema y programas de aplicación son copiados en cada directorio de usuario, con el desperdicio de espacio que supone.

Este último problema y la duplicidad de archivos puede ser solucionado si el SO es capaz de manejar nombres de rutas y dando permisos entre usuarios. En la duplicidad de archivos el SO crea un usuario ficticio especial al que se le asigna un directorio donde se referencian todos estos archivos.

9.8. Describir la estructura de árbol de directorios.

La lista de archivos almacenada en cada directorio, independientemente de su ubicación, puede contener entradas de otros directorios y archivos. Ofrece una gran flexibilidad. Si un usuario desea acceder a un archivo debe especificar la ruta de acceso al archivo. A cada uno de los nombres de directorios y archivo que aparecen en la ruta se les denomina componentes del nombre de la ruta.

9.9. Explicar las diferencias entre un nombre de ruta absoluto y un nombre de ruta relativo.

Absoluto contiene los nombres de todos los directorios a los que hay que acceder en orden para llegar al archivo comenzando por la raíz. Relativo especifica la ubicación del archivo comenzando desde el directorio de trabajo del usuario, también denominado directorio actual.

9.10. Explicar el significado de las entradas "." y " .. " de un directorio.

“.” Hace referencia al directorio actual y “..” al directorio padre.

9.11. ¿Qué ventaja ofrece la estructura de directorios de gráfica acíclica frente a la de árbol?

Permite que el mismo archivo o directorio pueda ser referenciado por dos o más directorios diferentes siempre que no se

produzcan ciclos, es decir, que un directorio X referencie a otro directorio Y el cual referencia de nuevo a X.

9.12. Explicar la diferencia entre un enlace duro y un enlace simbólico.

En los enlaces duro se crea en el directorio origen una nueva entrada que sea una copia parcial o completa de la entrada del directorio destino asociada al archivo o subdirectorio compartido. La nueva entrada creada en el directorio origen es marcada con algún identificador especial que permita reconocer al SO que se trata de una entrada asociada a un enlace.

Otra forma de implementar un enlace es como un tipo especial de archivo que contiene el nombre de ruta absoluta o relativa del archivo o subdirectorio compartido. A esta forma se les llama enlaces simbólicos.

El principal inconveniente es que consumen más espacio los simbólicos que los duros al ocupar bloques de datos frente a solo entradas de directorios.

9.13. ¿Qué define un sistema de archivos?

Define un esquema de almacenamiento de la información relativa a los archivos en un dispositivo de almacenamiento secundario.

¿Pueden existir diferentes sistemas de archivos en un mis disco duro?

Puesto que pueden existir varias particiones también es posible la existencia de varios sistemas de archivos en un disco, uno por partición disponible.

9.14. Enumerar y describir brevemente las diferentes áreas que se distinguen de forma general en estructura de un sistema de archivos.

- ***Bloque de arranque.*** Se sitúa al principio de la partición y puede contener el código necesario para arrancar un SO. La tabla de particiones indica la partición activa. Por defecto todas las particiones comienzan con un bloque de arranque y solo una necesita tener el código de arranque (partición activa)
- ***Estructura de datos con metadatos del sistema de archivos*****.** Contiene información administrativa y estadística del sistema de archivos, por ejemplo: un identificador del tipo de sistema de archivos, nº de bloques del sistema de

archivos, nº de bloques libres, etc… Este *superbloque* se copia en memoria cuando arranca el SO o se accede por primera vez al sistema de archivos

- ***Estructura de datos con información sobre los bloques libres en el sistema de archivos.*** Generalmente denominada como lista de bloques libres.
- ***Estructura de datos para localizar los bloques asignados a los archivos.*** Lista de nodos índice cada nodo tiene asociado un número entero positivo que lo identifica de manera única. Tabla de asignación de archivos (FAT) esta tabla tiene una entrada por cada bloque físico existente en la partición del sistema de archivos.
- ***Área de datos.*** Contiene los bloques libres y los asignados a archivos o directorios.

9.15. Explicar brevemente en qué consiste el montaje de un sistema de archivos y cuándo se realiza.

Cuando se instala un SO éste crea en una partición mediante un formateo a alto nivel un determinado sistema de archivos.

Para poder acceder a los contenidos de un determinado sistema de archivos ubicado en memoria secundaria, éste debe ser montado en algún punto dentro de la estructura de directorios del sistema de archivos principal o asignarle una estructura de archivos independiente. A dicho punto se le suele denominar punto de montaje.

9.16. ¿Qué ventajas e inconveniente tiene el método de asignación contigua de espacio de disco?

Minimiza las operaciones de búsqueda en disco, aumentando así su rendimiento. Permite soportar archivos tanto de acceso secuencial como aleatorio, ya que la localización de los bloques de un archivo se realiza de forma rápida y sencilla.

Las desventajas son que produce fragmentación externa, y esta es un problema si ninguno de los huecos existentes es del tamaño que necesita el SO para asignárselo a un archivo. Para solucionarlo se hace necesario compactar el disco, es decir, desplazar todos los archivos a un lado de la partición y eso requiere mucho tiempo. Otro inconveniente es que el SO debe conocer el tamaño que ocupará el archivo que se va a crear, esto en la mayoría de ocasiones no se puede predecir a priori.

9.17. Describir el método de asignación de espacio de disco conocido como asignación enlazada.

Consiste en almacenar al principio de cada bloque físico asignado a un archivo la dirección física del siguiente bloque físico del archivo. De esta forma el espacio ocupado por un archivo en el disco se organiza como una lista enlazada de bloques físicos. El SO para localizar un determinado bloque del archivo únicamente necesita conocer la dirección física del primer bloque. Dicha información está almacenada dentro de la entrada del directorio que contiene al archivo o en un nodo índice.

¿Cuáles son sus ventajas e inconvenientes?

Evita todos los inconvenientes de la asignación continua ya que cuando un archivo necesita más espacio se buscan bloques físicos libres y se enlazan al final de la lista.

La principal desventaja es que resulta muy lento si se utiliza para asignar espacio a archivos de acceso aleatorio, ya que para poder localizar la ubicación del bloque j es necesario buscar y leer en disco los anteriores j-1 bloques del archivo.

Otra desventaja proviene del hecho de que al principio de cada bloque del archivo se almacena la dirección física del siguiente bloque. Esta dirección consume unos pocos bytes del espacio del bloque.

9.18. ¿Qué es una tabla de asignación de archivos o FAT? Señalar las principales ventajas e inconvenientes de su uso.

Se ubicaría después de los bloques contiguos después del sector de arranque. Esta tabla tiene una entrada por cada bloque físico existente en la partición del sistema de archivos. Así la entrada j de la tabla, hace referencia al bloque físico j. Si el bloque físico j está asignado a un archivo, entonces la entrada j en la FAT contiene la dirección física del siguiente bloque del archivo. En una entrada de la FAT se almacena lo que antes se almacenaba al principio de cada bloque. De esta forma el espacio ocupado por un archivo en el disco se organiza como una lista enlazada de entradas enlazadas de la FAT, en vez de como una lista enlazada de bloques físicos. Al igual que antes la dirección física del primer bloque del archivo se almacena o en la entrada del directorio que contiene al archivo o en un nodo índice.

Si un bloque físico es el último asociado a determinado archivo entonces la entrada del bloque físico de la fat contendrá un identificador especial de fin de archivo (ej:-1). Si esta libre tendrá un numero entero especial (ej:0)

Las ventajas es un mejor soporte a archivos de acceso aleatorio, ya que para acceder a un determinado bloque únicamente debe seguirse la cadena leyendo entradas de la FAT. Además los bloques físicos solamente almacenan datos de los archivos.

La desventaja es que para un uso eficiente es conveniente tener una copia de la FAT en memoria principal y puede ocupar una cantidad de memoria no despreciable si la partición de disco es grande.

9.19. Describir el método de asignación de espacio de disco conocido como asignación indexada,

Consiste en almacenar en un nodo-i los atributos de un archivo y las direcciones físicas de los ocho o diez primeros bloques de un archivo. También se almacenan las direcciones físicas de uno o varios bloques de indirección simple, doble o triple. Un bloque de indirección simple es un bloque físico que almacena direcciones físicas de bloques del archivo, doble es cuando almacena bloques de direcciones de simple y triple cuando lo hace de doble. El tamaño de un nodo-i suele ser pequeños, unos pocos bytes, por lo que dentro de un bloque físico de discos se pueden almacenar múltiples nodos-i.

¿Cuáles son sus ventajas e inconvenientes?

Se puede usar tanto para archivos de acceso secuencial como aleatorio. No produce fragmentación externa y permite que el espacio de los bloques físicos que contienen datos del archivo pueda ser utilizado en su totalidad. Además requiere el uso de menos espacio en memoria principal que el método de asignación enlazada con FAT ya que solo es necesario mantener en memoria principal los nodos-i de los archivos abiertos.

La principal desventaja es que el tiempo de acceso a un bloque de archivo depende de la estructura interna del nodo-i, del tamaño de archivo y de la posición del bloque al que se desea acceder. Si se tratan de bloques del archivo a los que se accede a través de bloques de indirección simple, doble o triple entonces habrá que hacer una, dos o tres lecturas adicionales en disco respectivamente.

9.20. ¿Qué es un nodo índice?

Cada archivo tiene asociado un nodo-i que queda identificado mediante un número entero positivo denominado número de nodo-i. Al principio de la partición asociada al sistema de archivos se mantiene una lista con todos los nodos-i existentes. El número de

nodo-i asociado a un archivo se almacena en la entrada del directorio que contiene el archivo. Cuando se abre un archivo, su nodo-i se lee en la lista de nodos-i y se carga en memoria principal para poder conocer las direcciones físicas de los bloques del archivo y sus atributos.

9.21. Enumerar diferentes implementaciones de una lista de bloques libres y sus principales ventajas.

- ***Mapa de bits.*** Cada bloque de disco de la partición del sistema de archivos tiene asignado un bit. Si el bloque está ocupado se marca un 0 y si está libre 1 o viceversa. Si la partición asociada al sistema de archivos ocupa N bloques de disco entonces el mapa de bits ocupará N bits. La principal ventaja es la sencillez para encontrar bloques libres, el principal inconveniente es que para que sea eficiente es necesario mantener todo el mapa de bits en memoria principal.
- ***Lista enlazada.*** La lista de bloques libres también se puede implementar como lista enlazada. Supóngase que en un bloque de disco se puede almacenar Nd direcciones de bloque, entonces cada bloque de la lista contendrá Nd-1 direcciones de bloques libres y la dirección del siguiente bloque de la lista. La principal ventaja es que solo es necesario mantener simultáneamente en memoria principal un bloque de la lista enlazada. Cuando se necesitan bloques libres se utilizan los apuntados por el bloque de la lista enlazada, solo cuando estos se agotan es necesario cargar en memoria el siguiente bloque de la lista enlazada. Por otra parte, cuando se libera espacio asignado a un archivo, las direcciones de los nuevos bloques libres son incluidas en el bloque de la lista enlazada cargado en memoria principal. Si un bloque de la lista ya no tiene capacidad para almacenar más direcciones de bloques libres se escribe en el disco y se añade un nuevo bloque al principio de la lista.

9.22. ¿Qué información se suele almacenar de forma general en una entrada de un directorio?

Algunos sistemas de archivos como FAT32 almacenan en cada entrada de directorio el nombre de un archivo o subdirectorio y sus atributos, esta incluye información necesaria que permite localizar los bloques de datos del archivo. Otros como UFS o ext2 almacenan en cada entrada de directorio el nombre de un archivo y subdirectorio y un puntero (numero de nodo-i) a la estructura de datos (nodo-i) donde se almacenan los atributos del archivo. Esta

organización supone un acceso más al disco pero facilita la compartición de archivos mediante enlaces.

9.23. Señalar y describir brevemente tres formas diferentes de implementación de directorios.

- ***Directorios de igual tamaño.*** Todas las entradas de un directorio poseen el mismo tamaño. Primero se almacenan las informaciones asociadas y luego el nombre. La ventaja que es fácil de gestionar, la desventaja es que para dar soporte a nombres de archivos largos se requiere reservar en cada entrada el espacio máximo que puede ocupar el nombre.
- ***Directorios con entradas de tamaño variable*.** Cada directorio tiene un determinado tamaño. En cada entrada primero se almacena el tamaño que ocupa la entrada, a continuación las informaciones asociadas al archivo y por último el nombre que no puede superar un tamaño máximo. Al nombre del archivo se le añade un carácter extra que indique el final del mismo. Además se añaden caracteres de relleno para que el tamaño de una entrada sea un entero positivo de palabras de memoria principal. De este modo se evita que dentro de una misma palabra haya información sobre dos entradas de directorios distintas. La ventaja que se reduce el espacio desperdiciado dentro de un directorio, como máximo el tamaño desperdiciado por entrada es de casi una palabra. El inconveniente es que al borrar archivos se crean huecos de longitud variable.
- ***Directorios con entradas de igual tamaño y uso de un montículo (heap) para almacenar los nombres de los archivos.*** Cada entrada del directorio tiene el mismo tamaño, en ella se almacena un puntero al comienzo del nombre archivo dentro del montículo y las informaciones asociadas al archivo, al final del directorio se almacena la estructura de datos de tipo montículo para almacenar de nombres de archivos. Dentro del montículo los nombres se almacenan de forma contigua incluyendo un carácter especial para marcar el final de un nombre de archivo. Elimina la fragmentación externa a nivel de directorio, ya que si existen entradas libres siempre es posible añadir una nueva entrada al directorio. Además como las entradas son de tamaño fijo no es necesario incluir caracteres de relleno para que el tamaño de una entrada sea igual a un número entero de palabras. Su inconveniente es que las operaciones sobre el montículo complican la administración del directorio.

9.24. Señalar y describir brevemente diferentes formas de búsqueda de un archivo en un directorio.

Una posible búsqueda consiste en ir examinando una a una de las entradas contenidas en un directorio, comenzando desde la primera, en busca de aquella entrada que contenga el nombre de archivo sobre el que desea operar un proceso. Es fácil de implementar pero consume mucho tiempo si en el directorio están alojados cientos o miles de archivos.

Para ahorrar tiempo algunos SO mantienen una memoria caché software denominada caché de directorios con las entradas de los directorios recientemente accedidas. Cuando se tiene que buscar la entrada de un archivo primero se busca en la caché. En caso de acierto se ahorra el tiempo de leer en disco y buscar dentro de la lista.

Otra posibilidad consiste en implementar cada directorio con una tabla hash y su lista de archivos y subdirectorios. La tabla hash tiene un tamaño de N entradas. Cada entrada de la tabla contiene un puntero a una entrada de la lista. Cuando se desea buscar un archivo en el directorio se introduce su nombre en una función hash que genera un valor comprendido entre 0 y N-1. Dicho valor indica la entrada de la tabla hash que hay que leer. Si la entrada está vacía significa directamente que el archivo no está en el directorio. Si la entrada no está vacía hay que seguir el puntero contenido en la entrada de la tabla hash a la entrada correspondiente de la lista de archivos y comprobar que se corresponde con la del archivo buscado.

Para tratar las colisiones que se pueden producir debido a que la función hash puede generar el mismo valor con diferentes nombres de archivos, cada entrada de la tabla hash se puede implementar como una lista enlazada. En dicho caso, en vez de comprobar una única entrada de la lista de archivos habrá que ir comprobando a una de las entradas a las que apunta la lista enlazada de la entrada de la tabla hash. Obviamente esto ralentiza la búsqueda pero todavía sigue siendo mucho más rápida que una búsqueda lineal. El inconveniente es que su administración es más complicada, por ello solo es recomendable en sistemas con directorios con un número elevado de archivos.

9.25. ¿Cómo se puede verificar la consistencia de un sistema de archivos?

Con un verificador de consistencia al arrancar el computador si el sistema no ha sido apagado de forma adecuada. También

puede ejecutarlo el administrador del sistema cuando lo considere oportuno. Dicho verificador informa y corrige si son posibles las inconsistencias.

9.26. Enumerar diferentes tipos de inconsistencias que se pueden dar en un sistema de archivos y se pueden solucionar.

- ***Un bloque no aparece en la lista de bloques libres y tampoco está asignado a ningún archivo.*** La solución de esta inconsistencia consiste en añadir al archivo a la lista de bloques libres.
- ***Un bloque figura en la lista de bloques libres y también está asignado a un archivo***. La inconsistencia se soluciona eliminando el bloque de la lista de bloques libres.
- ***Un bloque figura varias veces en la lista de bloques libres.*** Esta inconsistencia solo se puede presentar en el caso de que la lista de bloques libres se implemente como una lista enlazada. Su solución pasa por reconstruir la lista de bloques libres dejando una única aparición del bloque.
- ***Un bloque está asignado a N archivos con N>1.*** Se trata de la peor inconsistencia posible. Una posible solución consiste en coger N-1 bloques libres, copiar en ellos el contenido del bloque y asociar cada copia a un archivo. Así se recupera la consistencia del sistema de archivos aunque seguramente la información contenida en N-1 de los archivos contendrá errores.

9.27. ¿En qué consiste la técnica de registro por diario (journaling)?

En que el SO genera un informe con las operaciones que tiene que realizar sobre el SO antes de realizarlas. Dicho informe se almacena en un área reservada de la partición de disco denominada diario (journal). Una vez que el informe ha sido escrito en el diario se comienzan a realizar las operaciones planificadas, cuando han sido completadas el informe se borra. Si se produce algún fallo antes de que se puedan completar todas las operaciones planificadas, cuando se reinicia el SO éste lee el diario para comprobar si existe un informe. En caso afirmativo se repiten una a una todas las operaciones para que el sistema de archivos quede de nuevo en un estado consistente.

9.28. ¿Qué diferencia existe entre copia de seguridad física y una copia de seguridad lógica?

La lógica copia directorios y archivos mientras que la física copia los bloques.

9.29. Describir los diferentes tipos de copias de seguridad lógicas que se pueden realizar.

- ***Copia completa*.** Contiene todos los archivos seleccionados por el administrador o el usuario.
- ***Copia diferencial.*** Contiene únicamente los archivos que han sido creados o modificados desde una última copia completa determinada
- ***Copia incremental.*** Contiene únicamente los archivos que han sido creados o modificados desde la última copia de seguridad completa o incremental.

9.30. ¿Qué es una instantánea (snapshot)?

Algunos sistemas de archivos como ZFS de Solaris cuando el SO necesita modificar contenido en un bloque de disco, en primer lugar localiza un bloque libre y copia el contenido modificado en dicho bloque, y a continuación modifica los punteros en memoria principal que apuntaban al bloque antiguo de tal forma que referencien al nuevo. Cada una de las modificaciones de las estructuras de control se etiqueta con un número de versión diferente basado en el tiempo de modificación de las mismas. De esta forma, la versión actual del sistema de archivos coexiste con las versiones anteriores del mismo. Cada vez que el número de versión actual se modifica se crea una nueva versión del sistema de archivos llamada snapshot.

9.31. ¿Qué diferencias existen entre una instantánea y una copia de seguridad?

- ***Las instantáneas se toman de forma prácticamente inmediata y no ocupan espacio en el momento de su creación.***
- ***Pueden ser restauradas con gran rapidez.***
- ***Residen en el mismo disco físico que los datos que respaldan*.**

9.32. Señalar tres formas de reducir el número de accesos a disco asociados a la gestión de archivos.

- ***Implementando en memoria principal una caché de buffers de disco.*** Cuando se necesita acceder a un bloque de disco, el SO comprueba en primer lugar si existe una copia del bloque en el caché de bloques.

- ***Para mejorar la tasa de aciertos usar lectura por adelantado de* bloques.** Consiste en copiar en la caché no

solo el bloque que ha producido un fallo sino varios bloques contiguos a éste, en previsión de que próximamente se tenga que acceder a ellos. Solo resulta útil en archivos de acceso secuencial.

Preguntas de autoevaluación tema 10

10.1. ¿Cuáles son los objetivos, desde el punto de vista de la seguridad, que debe cumplir un sistema informático?

- ***Confidencialidad.*** Los datos solo deben ser leidos por aquellos usuarios autorizados o por el propietario de los datos.
- ***Integridad.*** El contenido de los datos únicamente pueden ser modificados por su propietario y usuarios autorizados por el.
- ***Disponibilidad.*** Los recursos del sistema deben encontrarse disponibles para los usuarios autorizados.

10.2. ¿Cuándo se considera que un sistema informático se encuentra en un estado seguro?

Cuando sus recursos hardware y software son accedidos y utilizados únicamente por los usuarios autorizados según una política de seguridad preestablecida.

10.3. ¿Qué afirma el principio del mínimo privilegio?

Cada usuario debe tener los mínimos derechos de acceso necesarios para completar tareas que está autorizado a realizar.

10.4. ¿En qué consisten las políticas de seguridad de control de acceso discrecional?

Son definidas por el propietario de un recurso y especifican cuáles son los derechos de acceso de los restantes usuarios a dicho recurso.

¿Y las políticas de control de acceso obligatorio?

Definidas a nivel de diseño del sistema, regulan el acceso y flujo de información dentro del mismo.

10.5. Señalar algunas técnicas de autenticación de usuarios.

Nombre de usuario y contraseña, objeto físico, rasgo fisiológico.

10.6. ¿Cómo se mide la eficacia de un mecanismo de autenticación?

Se mide por el porcentaje de intrusos a los que se les concede el acceso y el porcentaje de usuarios legítimos a los que se les deniega acceso.

10.7. ¿Qué técnicas se pueden emplear para reforzar la seguridad del mecanismo de autenticación por contraseñas?

Envejecimiento de contraseñas: Las contraseñas solo son validas durante un periodo de tiempo, transcurrido el cual expiran y deben ser cambiadas.

Contraseñas de un solo uso: La contraseña expira cuando finaliza la sesión de usuario, por lo que cuando inicia otra vez debe usar una nueva contraseña. Para ello posee un libro de contraseñas que se han de usar en orden.

Reto dinámico: El sistema presenta al usuario un reto que debe resolver para acceder o para poder continuar trabajando.

10.8. ¿Qué es el software malicioso?

Aquel diseñado para causar daños o utilizar recursos de un computador sin el conocimiento y consentimiento de sus usuarios legítimos.

10.9. ¿Qué es una bomba lógica?

Un fragmento de código insertado en un programa legítimo que únicamente se activa cuando se cumplen o dejan de darse unas condiciones prestablecidas, como por ejemplo, una fecha y hora determinadas. Una vez activada puede provocar diferentes acciones: modificar, encriptar o borrar los archivos del disco duro, detener el computador, mostrar un mensaje por pantalla, enviar correos electrónicos, reproducir una canción o un video, etc.

10.10. ¿Qué es una puerta secreta?

Un fragmento de código insertado en un programa o sistema con la finalidad de poder saltarse los procedimientos de autentificación o ganar privilegios. Se activa al introducir ciertas secuencias especiales de entrada o una determinada secuencia de eventos.

10.11. ¿Qué es un caballo de Troya?

Es un programa aparentemente inofensivo que aparte de realizar aparentemente la función para la que está diseñado realiza una función desconocida y no deseable por el usuario del programa,

como borrado de archivos, envío de información al intruso o permitir acceso remoto del intruso al sistema.

10.12. ¿Qué es un virus?

Un fragmento de código insertado dentro del código de un programa anfitrión que se ejecuta si el usuario abre el programa anfitrión. Un virus puede realizar diferentes funciones nocivas, como por ejemplo el borrado de archivos, funciones molestas como mensajes en pantallas. Además un virus se propaga insertando copias de su código en otros archivos entre usuarios. Un computador se infecta cuando el usuario ejecuta un programa infectado.

Tienen hasta cuatro fases: latente, propagación, activación y ejecución en este orden.

Otra forma de virus utilizada en los últimos años son los virus insertados en macros, que son programas ejecutables embebidos en un documento de un procesador de texto, hoja de cálculo u otro archivo.

10.13. ¿Qué es un gusano informático?

Es un programa capaz de reproducirse y propagarse a otros computadores, generalmente a través de una red informática. Básicamente provoca una denegación del servicio o un servicio deficiente ya que en su función de reproducción realiza un consumo desproporcionado de recursos de los computadores y del ancho de banda de la red por la que se propaga.

10.14. ¿Qué es un programa espía?

Es un programa que se instala de forma furtiva en el computador por un virus o troyano y que se dedica a recopilar información sobre la actividad de sus usuarios para enviárselas a terceros.

¿Qué tipo de información recopila?

Entre la información que recopila se encuentran datos de conexión a Internet, clave de e-mail, mensajes enviados y recibidos, páginas web visitadas, descargas e información intercambiada a través de formularios electrónicos.

10.15. ¿Qué establece un dominio de protección?

Establece para un subconjunto de objetos las operaciones emitidas o derechos de acceso de cada uno de ellos. De esta forma

un dominio queda definido por un conjunto de pares de la forma [objetos, derechos].

10.16. ¿Qué información contiene la matriz de acceso?

Cada fila i está asociada a un dominio Di, mientras que cada columna j a un objeto Oj. Cada elemento Aij contiene los derechos de acceso asociados al objeto Oj dentro del dominio Di.

10.17. Explicar la implementación de la matriz de acceso como listas de control de acceso.

Consiste en asociar con cada objeto una lista denominada lista de control de acceso (ACL). Cada entrada contiene pares de la forma [dominio, derechos].

Cuando un proceso va a acceder a un objeto, se consulta su ACL usando como parámetro de búsqueda el dominio de protección. Si un dominio Di no tiene una entrada en la lista eso significa que no tiene ningún derecho sobre el objeto y se produce una excepción. Si el dominio Di tiene una entrada en la lista, se comprueba que los derechos en el dominio Di sobre el objeto permiten realizar la operación solicitada. En caso contrario se deniega el acceso.

La ventaja es que permite fácilmente revocar derechos de forma selectiva sobre objetos que pueden estar en varios dominios. El inconveniente es que la ACL se comprueba en cada acceso al objeto. Si los accesos son frecuentes y la lista larga se pierde tiempo en la búsqueda. Se puede reducir el número de entradas de las ACL.

10.18. Explicar la implementación de la matriz de acceso como listas de capacidades.

Cada entrada de la lista contiene un identificador de objeto, que indica la localización del objeto o de una estructura de datos que contiene su localización, y sus derechos de acceso. A la dupla (objeto, derechos) se le denomina capacidad.

Una posible implementación es almacenarla en el espacio de direcciones del núcleo y asociar a cada capacidad un identificador numérico que coincide con su posición en la lista. Cuando un proceso está ejecutándose dentro de un cierto dominio desea realizar una cierta operación sobre un objeto, éste debe especificar

como parámetro en la correspondiente llamada al sistema el identificador de la capacidad.

¿Cuáles son sus ventajas e inconvenientes?

Una de las ventajas es que el acceso al contenido es muy rápido, ya que no se realizan búsquedas dentro de la lista puesto que el propio proceso indica al SO la posición de la capacidad en la lista. Otra ventaja es que permite encapsular a los procesos fácilmente, basta con asociarles un determinado dominio, algo que no es posible con las ACLs.

Un inconveniente de esta implementación es el coste que supone para el sistema la revocación de los permisos para un objeto en concreto, ay que requiere acceder a todas las listas de capacidades existentes, y buscar en cada una si existe la capacidad que se desea revocar.

10.19. ¿Cuándo se puede afirmar que un sistema es confiable?

Cuando cumple con los requerimientos de seguridad o política de seguridad que se había impuesto.

10.20. ¿Qué es una TCB?

Base de computador confiable (Trusted computer base) es un conjunto de elementos hardware y software que le permiten implementar los requisitos de seguridad establecidos.

¿Cuáles son las funciones del sistema operativo que debe incluir?

Entre las funciones que debe incluir se encuentran las relativas a creación de procesos, cambio de contexto, gestión de memoria y parte de la gestión de la E/S y del sistema de archivos.

10.21. ¿Qué función realiza el monitor de referencia de una TCB?

Se encarga de comprobar todas las llamadas al sistema que puedan comprometer la seguridad del sistema, como la creación de procesos o la apertura de archivos, y decidir si pueden ser procesadas o rechazadas. Nótese que el monitor de referencia toma las decisiones en base a la política de seguridad que se haya decidido seguir en el sistema que se plasma en el contenido de la matriz de acceso.

10.22. ¿Qué es un sistema de seguridad multinivel?

Un modelo que distingue varios niveles de seguridad. A cada objeto y usuario se le asigna un determinado nivel. El modelo debe

especificar un conjunto de reglas para establecer el acceso de los usuarios a la información (objetos).

10.23. ¿Qué reglas sigue el modelo de seguridad multinivel de Bell y La Padula?

- ***Imposibilidad de leer objetos de niveles superiores.***
- ***Imposibilidad de escribir objetos de niveles inferiores.***

Si se cumplen estas dos reglas se puede garantizar que no existirán fugas de información desde un nivel de seguridad superior hacia un nivel inferior.

10.24. Señalar los principios de diseño de sistemas operativos seguros propuestos por Saltzer y Sehr.

- ***El diseño debe ser público*.**
- ***El estado por defecto es el de no acceso.***
- ***Comprobación de los derechos de acceso.***
- ***Principio del mínimo privilegios.***
- ***Los mecanismos de protección deben ser simples y estar integrados en las capas más bajas del sistema.***
- ***Los mecanismos de protección deben ser aceptados por los usuarios.***
- ***Los mecanismos de protección deben ser aceptados por los usuarios.***
- ***Los mejores diseños son los más sencillos.***