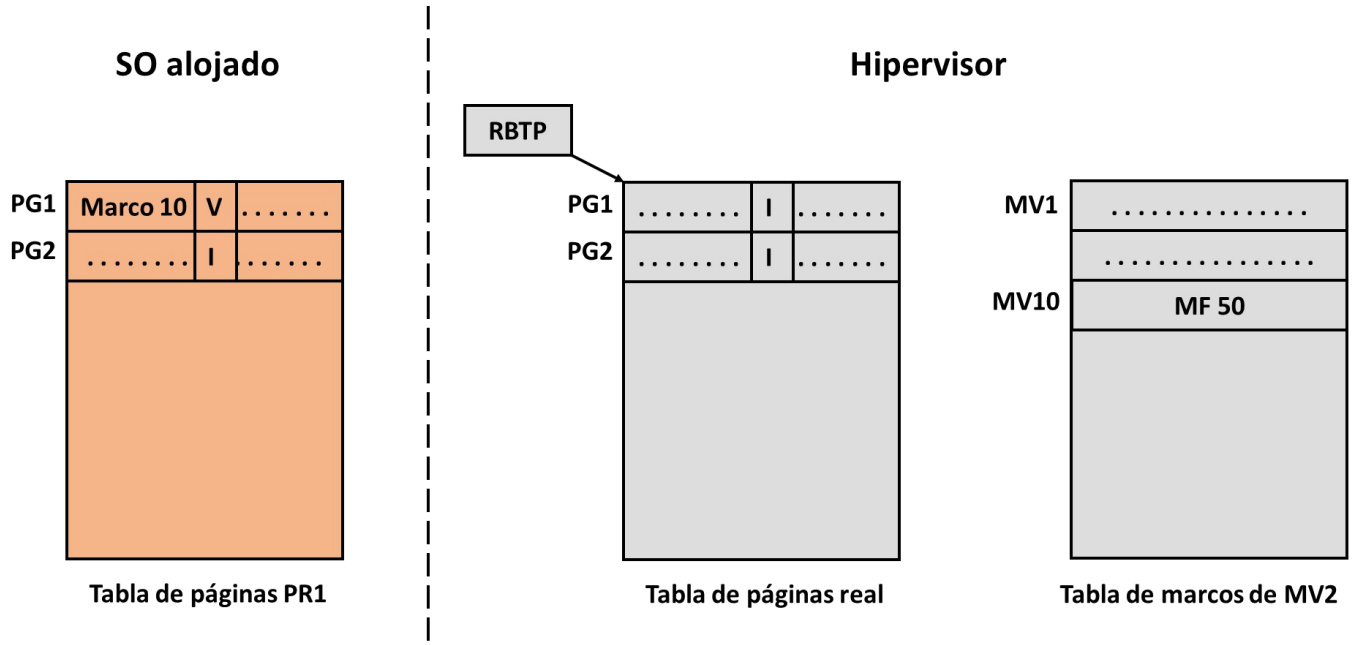
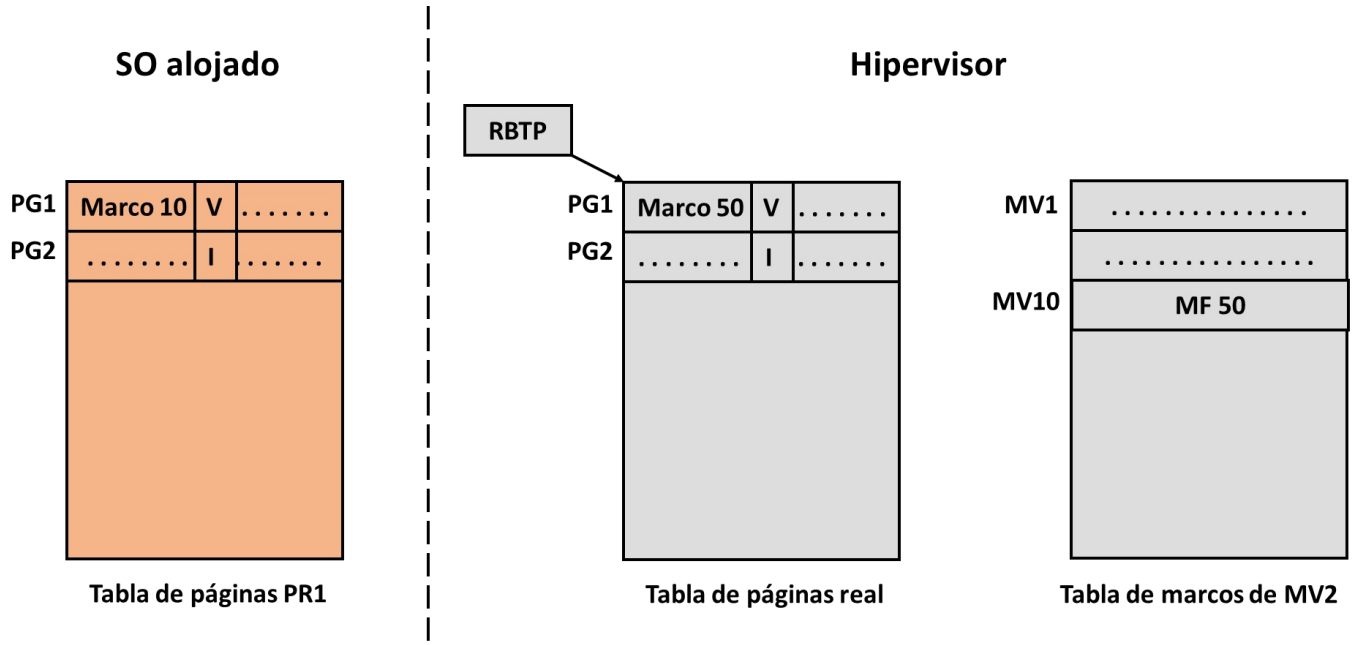
Solución del 2º parcial de Sistemas Operativos Avanzados (19/6/2018)

1) *Considere un hipervisor que usa la técnica denominada "Virtual TLB" para virtualizar la memoria de un sistema. Suponga que un proceso PR1 de la máquina virtual MV2 tiene residente su primera página (PG1), asignada al marco 10 de MV2 que corresponde al marco físico 50, pero no ha accedido todavía a la segunda página (PG2). Asimismo, considere que los marcos de MV2 a partir del 20 están libres, mientras que en la máquina física están libres a partir del marco 90. Suponga que acaba de ocurrir un cambio de contexto a PR1 que, a continuación, accede primero a PG1 y luego a PG2. Se pide que dibuje el estado de* ***todas*** *las tablas relativas a PR1 y MV2 requeridas por esta técnica, identificando cuál está siendo gestionada por la MMU, en los siguientes instantes:* ***(a)*** *justo después del cambio de contexto;* ***(b)*** *después del primer acceso;* ***(c)*** *después del segundo acceso.*

(a) Cuando hay un cambio de contexto, el sistema operativo alojado asigna la dirección de la tabla de páginas del proceso seleccionado por el planificador (*PR1*) al registro base de la tabla de páginas del procesador (*RBTP*). Al tratarse de una instrucción privilegiada, esa operación causa una excepción y toma control el hipervisor que guarda el valor que el sistema operativo alojado ha intentando escribir en el registro, construye una tabla de páginas vacía y hace que el *RBTP* del procesador apunte a esa tabla vacía. El hipervisor gestiona para esa máquina virtual una tabla que almacena la correspondencia entre los marcos que asigna el sistema operativo alojado y los marcos físicos reales.

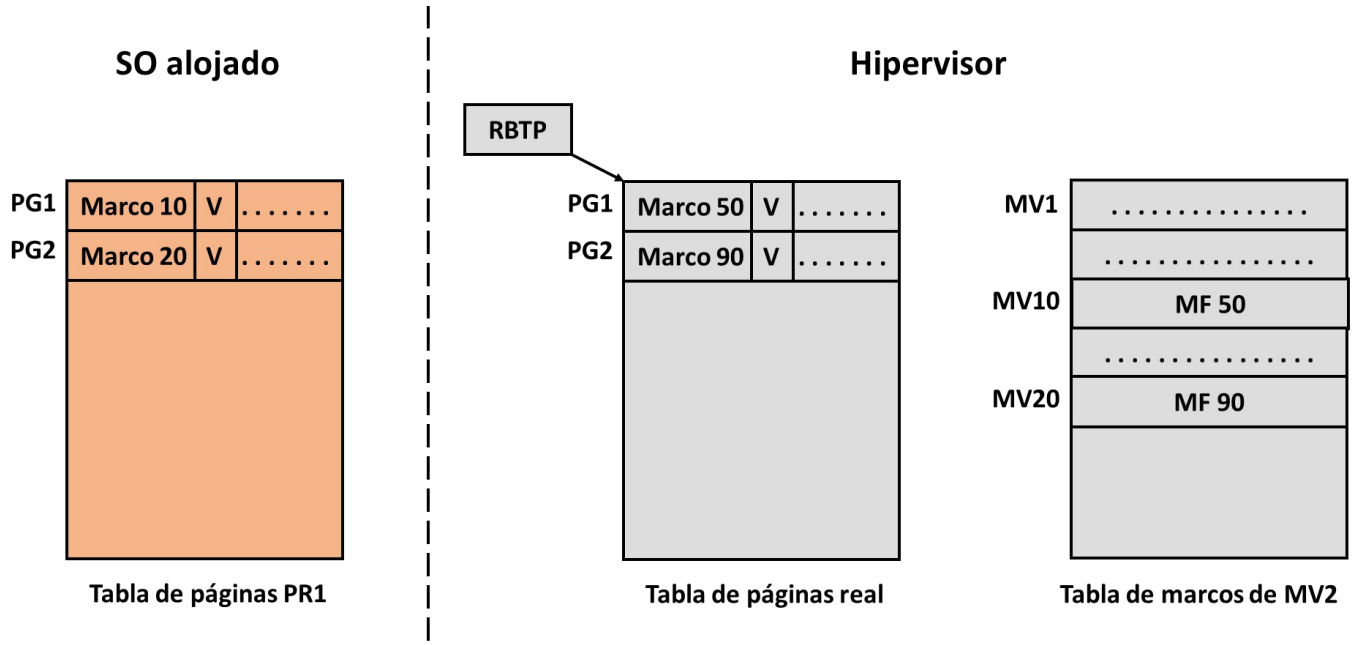


(b) Cuando *PR1* accede a su primera página, se produce un fallo de página, puesto que en la tabla de páginas real esa entrada está marcada como inválida, y toma control el hipervisor. El hipervisor conoce la dirección de la tabla de páginas de PR1 que gestiona el sistema operativo alojado y la consulta obteniendo que se trata de una página residente en el marco 10. Usando la tabla de correspondencia entre los marcos que gestiona el sistema operativo alojado y los reales, determina que al marco virtual 10 le corresponde el marco físico 50. En consecuencia, copia la entrada de la tabla de páginas del sistema operativo alojado a la tabla de páginas real pero asignándola el marco 50. La rutina de tratamiento del fallo de página se completa y prosigue la ejecución del proceso.



(c) Cuando *PR1* accede a su segunda página, se produce un fallo de página, puesto que en la tabla de páginas real esa entrada está marcada como inválida, y toma control el hipervisor. El hipervisor comprueba que en la tabla de páginas de *PR1* que gestiona el sistema operativo alojado la entrada es también inválida, por lo que le propaga la excepción al sistema operativo alojado. Este realiza el trabajo habitual, asignándole un marco libre (el marco 20) y poniendo la entrada como válida. La rutina de tratamiento del fallo de página se completa y prosigue la ejecución del proceso, produciéndose nuevamente un fallo de página ya que la entrada de la tabla real está marcada como inválida.

El tratamiento de ese segundo fallo de página en cadena es básicamente el mismo que en el apartado anterior. El hipervisor detecta que en la tabla de páginas del sistema operativo alojado la entrada es válida y tiene asignado un marco (el 20). En este caso, sin embargo, al consultar la tabla de correspondencia de marcos, el hipervisor encuentra que ese marco virtual no tiene asignado ningún marco físico. Por tanto, selecciona un marco físico libre (el 90), incluye la correspondencia entre el marco virtual 20 de MV2 con el 90 de la memoria real y, como en el apartado anterior, copia la entrada de la tabla del sistema operativo alojado a la real cambiando el marco.



2) *En el contexto de los sistemas de archivos FAT32:* ***(a)*** *Describa la estructura de datos FAT: tamaño, contenido, uso, etc.;* ***(b)*** *Describa cada campo de la entrada de directorio: tamaño, contenido, uso, etc.;* ***(c)*** *Explique los tamaños máximos (limitaciones) en el FAT32.*

---

3) *La llamada de UNIX rename (`int rename(const char *oldpath, const char *newpath)`) mueve el fichero del primer directorio al segundo. Para asegurar que no hay problemas en el acceso a esos directorios mientras se está realizando la operación de renombrado, se bloquea el acceso a los mismos durante la operación.* ***(a)*** *Proponga un ejemplo de interbloqueo que involucre a 3 procesos que usen esta llamada especificando la traza de ejecución conflictiva.* ***(b)*** *Dibuje el diagrama de asignación de recursos correspondiente al ejemplo explicando cómo se detecta el interbloqueo en el mismo.* ***(c)*** *Plantee cómo se podría incorporar a la implementación de esta llamada una técnica de prevención de interbloqueos.*

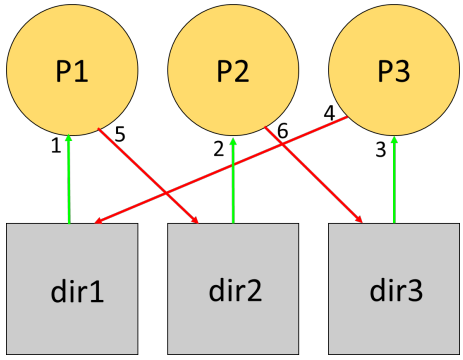
(a) Considere que tres procesos ejecutan simultáneamente las siguientes operaciones:

```
P1: renombrar("/dir1/fA", "/dir2/fB");
P2: renombrar("/dir2/fC", "/dir3/fD");
P3: renombrar("/dir3/fE", "/dir1/fF");
```

Se podría dar la siguiente ejecución intercalada que conduce a un interbloqueo:

1. Llamada de `P1`: solicita bloquear el acceso a `"/dir1"` → lo consigue
2. Llamada de `P2`: solicita bloquear el acceso a `"/dir2"` → lo consigue
3. Llamada de `P3`: solicita bloquear el acceso a `"/dir3"` → lo consigue
4. Llamada de `P3`: solicita bloquear el acceso a `"/dir1"` → se bloquea
5. Llamada de `P1`: solicita bloquear el acceso a `"/dir2"` → se bloquea
6. Llamada de `P2`: solicita bloquear el acceso a `"/dir3"` → interbloqueo

(b) La siguiente figura muestra el diagrama de asignación de recursos de la traza del punto anterior.



En el diagrama se puede detectar la existencia de un interbloqueo entre los tres procesos por la existencia de un ciclo en el grafo que involucra a los tres procesos.

(c) Para resolver el problema puede usarse la técnica de prevención de interbloqueos basada en la ordenación de los recursos, de manera que las solicitudes de bloqueo de los directorios se hagan siempre teniendo en cuenta un cierto orden total arbitrario, como, por ejemplo, el número de descriptor (en UNIX, inodo) asociado al directorio.

```
renombrar(rutaPrevia, rutaNueva) {
    dirOrg = directorio padre de rutaPrevia
    dirDst = directorio padre de rutaNueva
    if (dirOrg != dirDst) {
        if (dirOrg->descriptor < dirDst->descriptor)
```

```
                    Bloquea acceso a dirOrg; Bloquea acceso a dirDst;
              else
                    Bloquea acceso a dirDst; Bloquea acceso a dirOrg;
              Elimina entrada rutaPrevia de dirOrg
              Añade entrada rutaNueva en dirDst
              Desbloquea acceso a dirOrg
              Desbloquea acceso a dirDst
        }
        else ..........
  }
```

Suponiendo que el descriptor de `/dir1` es menor que el de `/dir2` y este menor que el de `/dir3`, se produciría la siguiente traza de ejecución libre de interbloqueos:

1. Llamada de `P1`: solicita bloquear el acceso a `"/dir1"` → lo consigue
2. Llamada de `P2`: solicita bloquear el acceso a `"/dir2"` → lo consigue
3. Llamada de `P3`: solicita bloquear el acceso a `"/dir1"` → se bloquea
4. Llamada de `P1`: solicita bloquear el acceso a `"/dir2"` → se bloquea
5. Llamada de `P2`: solicita bloquear el acceso a `"/dir3"` → lo consigue

4) *Sea una organización que usa un modelo de control de acceso DAC basado en ACLs en la que un miembro de la misma (usuario J) posee un fichero F donde almacena información confidencial. Una persona malintencionada consigue una cuenta de invitado (usuario I) en la organización e intenta conseguir una copia de los datos confidenciales en su fichero C usando la técnica del caballo de troya. Para ello, convence a J de que ejecute un programa que, además de su funcionalidad conocida y deseada, realiza el ataque a la seguridad previsto.* ***(a)*** *Explique cómo llevaría a cabo su acción maliciosa este programa indicando qué información de control de acceso deberían tener F (la que debería haber especificado J pensando en su seguridad) y C (la definida por I teniendo en cuenta el modus operandi de su ataque).* ***(b)*** *Describa cómo añadir un modelo MAC de Bell-Lapadula puede evitar este problema especificando qué información de control de acceso deberían tener J, F, I y C.*

(a) Dado el carácter confidencial del fichero *F*, el usuario *J* especificaría mediante una ACL que solo puede acceder al fichero el mismo. Por su parte, el usuario *I* fijaría unos permisos en *C* mediante una ACL de manera que el usuario *J* (o, incluso, todo el mundo) pueda escribir en ese fichero, para poder realizar la filtración de la información, pero solo el dueño pueda leerlo (*I* no quiere que los datos filtrados sean públicos).

Cuando el usuario *J* ejecuta el programa manipulado, además de su labor bienintencionada, este puede realiza la copia de *F* a *C* puesto que el usuario *J* puede leer de *F*, al ser su dueño, y escribir en *C*, ya que el dueño de ese fichero le ha otorgado permiso para ello.

(b) Existirían, al menos, dos niveles de seguridad (por ejemplo, alta y baja) y la siguiente asignación de niveles a sujetos y objetos:

- El usuario *J* tendría asignado un nivel de seguridad alto.
- El usuario *I* tendría asignado un nivel de seguridad bajo.
- El fichero *F* tendría asignado un nivel de seguridad alto.
- El fichero *C* tendría asignado un nivel de seguridad bajo.

Con esta disposición, cuando el usuario *J* ejecuta el programa malintencionado, este puede leer el fichero confidencial *F*, pero no puede copiar su información en *C*, gracias a la propiedad * (no *write-down*) del modelo MAC de Bell-Lapadula que impide que un sujeto pueda escribir en objetos que tengan asociado un nivel de seguridad más bajo.