Sistemas Operativos
2° Rec 2° Parcial 1C2017 - Resolución

Aclaración: La mayoría de las preguntas o ejercicios no tienen una única solución. Por lo tanto, si una solución particular no es similar a la expuesta aquí, no significa necesariamente que la misma sea incorrecta. Ante cualquier duda, consultar con el profesor del curso.

Teoría

1) Con el esquema tradicional de paginación, se podría compartir memoria teniendo distintas tablas de páginas apuntando a los mismos frames, es decir, para un proceso puede ser la pág 4 y para otro la pág 2, pero en memoria física estarán en el mismo frame. Ejemplo:

TP PA

0	0
1	2
2	4

Memoria Física

0	P0 PA
1	P1 PB
2	COMPARTIDA
3	P2 PB
4	P2 PA

TP PB

0	-
1	1
2	3
3	2

Con el esquema de TP invertida, no se puede compartir memoria en forma directa, ya que la entrada para un frame indica la página de qué proceso se encuentra. Para poder implementarlo se debería hacer alguna modificación, por ejemplo agregando alguna estructura por proceso que contenga una referencia a las páginas compartidas.

2) *open*: se validan los permisos del archivo con respecto al usuario, se crea una entrada en la tabla global de archivos abiertos (si es que no existía antes) y una entrada en la entrada de archivos abiertos del proceso en cuestión (seteando el puntero apropiadamente).

close: se elimina la entrada en la tabla global (si es que no está abierto por otros procesos) y se elimina la entrada de la tabla del proceso. Se escriben bytes que hayan quedado en el buffer (cache)

3) a. *Falso*. Que sea no bloqueante significa que va a volver el control al proceso inmediatamente, con el resultado si es que estaba disponible o con un error. Sin embargo, si es asincrónica, va a programar la E/S y va a avisar al proceso de la respuesta una vez que esté completa.
b. *Falso*. N-Step-SCAN al ir atendiendo los pedidos de a tandas (formadas por las colas de tamaño N) respeta más el orden de llegada, que F-SCAN, que tiene solo una cola pasiva (siempre y cuando N no sea un valor demasiado grande).

4) Problemática que se da cuando el sistema operativo invierte más tiempo intercambiando paginas de procesos entre memoria swap y memoria física, en lugar de realizar procesamiento útil (ejecución de los procesos). Esto se debe una alta tasa de page faults. Si se incrementa la memoria (lo cual implica que habria mas paginas en memoria por proceso) o se agrandan las páginas de los procesos (lo cual implica más bytes en memoria por proceso), el thrashing disminuye. Si se aumenta el nivel de multiprogramación, el thrashing tiende a aumentar (menor cantidad de páginas en memoria por proceso).
Nota: La respuesta podría variar con consideraciones particulares en relación a la asignación de frames por proceso (si es local/global, y si puede variar o no para cada alternativa considerada)

5) La situación podría ser: un archivo de un trabajo practico sobre el que el propietario quiera tener todos permisos y les de permisos de lectura y escritura al resto de sus compañeros. El resto de los usuarios no deberían tener permisos
Se vería así: TP - rwx rw- ---o
Luego se agrega la necesidad de darle permiso de lectura al ayudante que va a revisar el TP. Para esto se podría:

- Usar ACL en la cual habría un permiso para el propietario, otro para el grupo de tp, otro para el ayudante y otro para el resto
- Usar una matriz de accesos, en este caso habría que completar los permisos para todos los users.
- Usar combinación de Propietario|Grupo|Resto con las ACLs extendidas (que unix provee). En este caso se definiría el caso particular del ayudante con permiso de lectura.

Práctica

Ejercicio 1 - Memoria

Los frames son de 64KiB => 2^16 => 16 bits utilizados para offset
Por lo tanto, quedan 16 más para identificar nro de pág, siendo 8 para el primer nivel y 8 para el segundo

Ej, DL PA AAAA1234h se interpreta como la pág AAh dentro de la primera TP, la pág AAh dentro de la segunda , con un offset de 1234h en el frame final.

PA AAAA1234h -> nro pág AAAA , no se encuentra en la TLB, en la TP P = 0 => PF

Hay frames libres, se le asigna el 9 => la DF será 91234h

Accesos: a la TLB se debe acceder 2 veces, la primera vez para buscar la pág, obteniendo TLB miss y la segunda para agregar la pág AAAAh.
A TPs. Primero se accede a la TP principal, donde P = 0, luego se va a buscar la sección de la TP requerida, por lo que se actualiza dicha TP, luego se accede a la TP de segundo nivel obteniendo nuevamente P = 0, por lo que se carga la página del proceso en el frame 9, actualizando posteriormente dicha TP . Por lo tanto serían 4 accesos.

Estado TLB

Pág	Frame
AAAA	9

PA AAAA2234h -> nro. pág. AAAA , se encuentra en lA TLB -> TLB hit, no hay PF
De la TLB se recupera el frame 9 => la DF será 92234h

Accesos: un acceso a la TLB para obtener el frame. No hay accesos a Tps.

Estado TLB

Pág	Frame
AAAA	9

PB AAAA0101h -> nro pág AAAA , como se cambia de proceso, hay que limpiar la TLB (no hay indicador de proceso en la misma) => no se encuentra en l0 TLB, en la TP P = 0 => PF
Hay frames libres, se le asigna el 5 => la DF será 50101h

Accesos: a la TLB se debe acceder 2 veces, la primera vez para buscar la pág, obteniendo TLB miss y la segunda para agregar la pág AAAAh.
A TPs. Primero se accede a la TP principal, donde P = 0, luego se va a buscar la sección de la TP requerida, por lo que se actualiza dicha TP, luego se accede a la TP de segundo nivel obteniendo nuevamente P = 0, por lo que se carga la página del proceso en el frame 5, actualizando posteriormente dicha TP . Por lo tanto serían 4 accesos.

Estado TLB

Pág	Frame
AAAA	5

PB AA102334h -> nro pág AA10 , no se encuentra en la TLB, en la TP P = 0 => PF
Hay frames libres, se le asigna el 7 => la DF será 72334h

Accesos: a la TLB se debe acceder 2 veces, la primera vez para buscar la pág, obteniendo TLB miss y la segunda para agregar la pág AA10h.
A TPs. Primero se accede a la TP principal, donde P = 0, luego se va a buscar la sección de la TP requerida, por lo que se actualiza dicha TP, luego se accede a la TP de segundo nivel obteniendo nuevamente P = 0, por lo que se carga la página del proceso en el frame 7, actualizando posteriormente dicha TP . Por lo tanto serían 4 accesos.

Estado TLB

Pág	Frame
AAAA	5
AA10	7

PB AAAA4452h -> nro. pág. AAAA , se encuentra en la TLB => TLB hit => no hay PF
Se recupera el frame 5 de la TLB => la DF será 54452h

Accesos: a la TLB se debe acceder 1 vez para recuperar el frame. No hay accesos a TPs

Estado TLB

Pág	Frame
AAAA	5
AA10	7

PB 32544111h -> nro pág 3254 , no se encuentra en la TLB, en la TP P = 0 => PF
No hay frames libres, se elige una víctima, la pág AA10 que es la que hace más tiempo no se referencia, por lo que se asignará el frame 7 => la DF será 74111h

Accesos: a la TLB se debe acceder 3 veces, la primera vez para buscar la pág, obteniendo TLB miss, la segunda cuando se reemplace la pág AA10 hay que borrar dicha entrada y finalmente para agregar la pág 3244h.
A TPs. Primero se accede a la TP principal, donde P = 0. Luego hay que reemplazar la pág AA10 por lo que hay que actualizar la TP de segundo nivel para indicar que ya no está presente. Luego se va a buscar la sección de la TP requerida, por lo que se actualiza dicha TP, luego se accede a la TP de segundo nivel obteniendo nuevamente P = 0, por lo que se carga la página del proceso en el frame 7, actualizando posteriormente dicha TP . Por lo tanto serían 5 accesos.

Estado TLB

Pág	Frame
AAAA	5
3244	7

Ejercicio 2 - File System

Los inodos tienen 10 punteros directos, 1 puntero indirecto simple y 1 puntero indirecto doble.
Los bloques son de 4KiB y los punteros de 4 bytes.

a) Tamaño máximo teórico de un archivo:

Cada bloque de punteros contiene 4 KiB / 4 Bytes = 1024 punteros

Punteros Directos:
Cantidad de punteros directos * tamaño de bloque = 10 * 4.096 bytes = 40.960 Bytes = 40 KiB

Puntero indirecto simple
Cantidad de punteros indirectos simples * cantidad de punteros por bloque * tamaño de bloque = 1 * 1024 * 4096 bytes = 4.194.304 Bytes = 4 MiB

Puntero indirecto doble
Cantidad de punteros indirectos dobles * cantidad de punteros por bloque^2 * tamaño de bloque = 1 * 1024^2 * 4096 Bytes = 4.294.967.296 bytes = 4 Gib.

Tamaño Máximo Teórico = 4 Gib + 4 Mib + 40 Kib aproximado 4 Gib.

b) parcial.txt tiene un tamaño de 175 KiB. Por lo tanto está compuesto por 175 Kib / 4KiB bloques = 43,75 Bloques. Se requiere leer 44 bloques de datos.

Además de leer los 44 bloques de datos es necesario leer 1 bloque de punteros. Total 45 bloques.

c)
1. Los alumnos no tienen acceso al directorio /home/docente
   Propietario: docente permisos RWX.
   Grupo: docente permisos R-X.
   Otros: permisos ---.
2.
   a) No pueden acceder por medio de un softlink debido a que estarían intentando acceder a un directorio al que no tienen accesos "/home/docente/parcial.txt"
   b) Pueden acceder debido a que acceden directamente al inodo que corresponde el archivo. El archivo tiene permisos de lectura para el grupo alumnos.
   c) En caso del softlink no tienen acceso.
   En caso del hardlink, los alumnos continúan teniendo acceso al archivo porque el docente al borrar el archivo (la entrada de directorio), resta en uno el contador de hardlinks y como el valor queda en 1 el archivo no se borra.

Ejercicio 3 - Entrada / Salida

Pedido	40	89	37	56	43	9	2	15
Instante	0 ms	9 ms	0 ms	1 ms	3 ms	25 ms	14ms	15 ms

SSTF

T = 0 --| 35 -> 37
T = 6 --| 37 -> 40
T = 12 --| 40 -> 43
T = 18 --| 43 -> 56
A partir de este instante, todos los pedidos están disponibles
T = 44 --| 56 -> 89 -> 15 -> 9 -> 2

CSCAN

T = 0 --| 35 -> 37
T = 6 --| 37 -> 40
T = 12 --| 40 -> 43
T = 18 --| 43 -> 56
A partir de este instante, todos los pedidos están disponibles (99 y 0 son el tope y la base)
T = 44 --| 56 -> 89 -> **99 -> 0** -> 2 -> 9 -> 15