PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2333 — Sistemas Operativos y Redes — 1/2018
Interrogación 3

Martes 5-Junio-2018

Duración: 2 horas
SIN CALCULADORA

1. **[21p]** El nuevo competidor tecnológico *gitsoft* desea diseñar un nuevo sistema de archivos. Todos los bloques en el sistema de archivos deben ser del mismo tamaño. Los tamaños posibles son 1KB, 2KB, 4KB u 8KB. El tamaño de cada puntero, en Byte, debe ser una potencia de 2. Los nombres de archivos deben permitir hasta 256B, incluyendo el caracter `'\0'`, deben almacenar al menos 3 *timestamp* de 64 bit, junto con dos enteros de 16-bit cada uno que indican el dueño y grupo del archivo, respectivamente, y 8 bit para indicar los permisos. El sistema debe permitir usar subdirectorios de profundidad arbitraria, sin embargo la cantidad máxima de archivos en cada directorio puede estar limitada.

   El sistema de archivos debe poder administrar discos de al menos 1GB. En cuanto al tamaño de los archivos (espacio para datos de los archivos), se estudió que una gran mayoría son menores a 100KB, por lo tanto la lectura de estos archivos debería ser muy rápida, sin embargo debe ser capaz de soportar archivos de, al menos, 1TB.

   Diseñe un sistema de archivos que cumpla con los requisitos solicitados. Debe mostrar que su diseño cumple con los siguientes puntos:

   a) **[3p]** Soportar el tamaño de disco requerido.

      R.

      Para soportar un disco de al menos 1GB, se necesitan punteros capaces de direcciones bloques suficientes. Con un punteros de 2 Byte (16 bit) se pueden direccionar hasta $2^{16} = 65536$ bloques. Con este tamaño y los bloques más grandes (de 8KB), se pueden utilizar discos de hasta hasta 2^{19}KB $= 512MB$, por lo tanto tienen que ser dimensiones mayores. Los punteros deben ser de, al menos 4 Byte. (32bit)).

   b) **[5p]** Poseer una estructura de bloque apropiada para representar archivos, incluyendo su *metadata*. Puede usar un esquema para describirlo.

      R.

      2p. Contenido de *metadata*. La *metadata* total descrita corresponde a 3 *timestamp* de 64bit $=$ 8Byte cada uno, dos enteros de 16bit $=$ 2Byte y finalmente 8bit $=$ 1Byte, sumando 11Byte. El nombre de archivo (256Byte) puede ir aquí, en el bloque índice, o bien en un bloque directorio (o en ambos, sin embargo esto podría ser fuente de inconsistencias si no se actualizan apropiadamente). Si se deciden agregar el nombre de archivo al bloque índice, hay que sumar 256Byte a *metadata*. También es posible agregar al menos 1bit más a la *metadata* para indicar el caso en que se trate de un bloque de subdirectorio (en caso que sea apropiado).

      2p. Composición del bloque. Debe indicar cuántos punteros, ya sea directos o indirectos se encuentran en el bloque. Para los bloques indirectos se debe indicar los niveles de indirección.

      1p. Tamaño del bloque. La cantidad de punteros más la *metadata* debe ser menor o igual al tamaño de bloque elegido.

   c) **[3p]** Proveer un acceso eficiente a archivos pequeños.

      R.

      2p. Debe cumplir que haya punteros directos a bloques de datos que se llenarían en primer lugar.

      1p. La cantidad de punteros directos debe permitir apuntar a archivos de tamaño 100KB. Esto depende de la cantidad de punteros disponibles y el tamaño de cada bloque.

d) **[5p]** Permitir el tamaño de archivo requerido.

R.

5p. Debe mostrar que la estructura de punteros permite alcanzar el tamaño de archivo máximo de, al menos, 1TB. No es necesario dar un número preciso para el tamaño de archivo máximo, pero sí debe quedar claro que alcanza o supera el límite de 1TB. Como referencia, con punteros de 4Byte (el mínimo posible) y bloques de 4KB, un puntero de indirección triple permite alcanzar archivos de 4TB. Sin embargo, con punteros de 8Byte, se necesitan bloques de al menos 8KB para llegar superar el límite. Por otro lado, con solo un puntero de indirección doble, bloques de 8KB y punteros de 8Byte, solo se llega hasta los 8GB.

e) **[5p]** Poseer una estructura apropiada para representar directorios. Puede usar un esquema para describirlo.

R.

Para describir directorios se puede extender la idea similar propuesta para la Tarea 3 (que solo permite un nivel de directorio). Se pueden plantear tres tipos de bloque: bloque índice para archivos, bloques de datos para archivos, y bloques de directorio. Los bloques de directorio pueden almacenar los nombres de archivos y mantener un puntero al bloque índice correspondiente, o bien dejar los nombres de archivos en la *metadata* de cada archivo. Sea cual sea la alternativa, debe ser consistente con la manera en que cumplió el ítem (b). La cantidad de archivos por directorio puede estar limitada por el tamaño del bloque. El contenido apuntado por cada entrada del directorio puede ser, o bien, un bloque índice (que representa un archivo), o bien un bloque directorio y así se permite una cantidad arbitraria de subdirectorios.

3p. Describir la estructura de un bloque que permita almacenar directorios, o bien como re-utilizar un bloque índice para esto.

2p. Debe mostrar que su estructura permite almacenar todos los niveles de directorio que el espacio en disco permita.

NOTA: Esta es una pregunta bastante abierta. No hay una solución única.

2. **[9p]** Preguntas cortas sobre sistemas de archivos. Su respuesta debe ser breve y precisa.

2.1) **[3p]** Considere la implementación de *links* simbólicos (*soft-links*) y de *hard-links* en un sistema de archivos. ¿Qué implementación considera que es más fácil de proveer en un sistema de archivos, y por qué? Puede pensar en cómo lo hubiese hecho para su Tarea 3 (pero no se pide que describa su Tarea 3).

R.

La respuesta correcta depende de cómo se describan comparativamente las implementaciones respectivas. Sin embargo, es más común considerar que la implementación de *soft-links* es más sencilla ya que se trata de un tipo especial de archivo cuya única función es referenciar otra entrada de directorio (y no otro bloque índice). De esta manera, eliminar el *soft-link* es tan fácil como eliminar un archivo, y no está relacionado con el contenido (entrada de directorio) apuntado. El archivo original puede ser eliminado de manera independiente del *soft-link* y el *soft-link* pasaría a ser un *dead-link*.

Por otro lado, un *hard-link* suele ser más difícil de implementar ya que un *hard-link* **debe** apuntar a un bloque índice y no a una entrada de directorio. Esto implica que si se elimina el *hard-link* puedo eliminar el archivo original también. Sin embargo, el archivo original debe ser eliminado **solamente** si no hay otros *hard-links* que apunten a él. Una manera eficiente de saber si hay otros *hard-link* es utilizar un contador de referencias en la metadata del bloque índice del archivo, que se incremente cada vez que se cree un *hard-link*. De esta manera solo el último *hard-link* que se elimine se debería encargar de eliminar el archivo.

1p. Mencionar la facilidad/dificultad del *soft-link*

1p. Mencionar la facilidad/dificultad del *hard-link*

1p. Justificación de por qué uno es más fácil.

2.2) **[3p]** ¿Qué ventaja tiene la existencia de una capa VFS (*Virtual File System*) en un sistema operativo?

R.

Permite agregar fácilmente un nuevo sistema de archivos al sistema operativo, ya que la implementación del sistema de archivos solamente debe comunicarse con la capa VFS y no requiere modificaciones importante al *kernel*.

2.3) **[3p]** Los sistemas operativos mantienen el registro de archivos abiertos usando dos tablas: una tabla de archivos abiertos para todo el sistema, y una tabla de archivos abiertos *por proceso*. ¿Para qué se mantienen dos tablas en lugar de usar solo una tabla para todo el sistema?

R.

Mantener dos tablas permite separar el uso por cada procesos, e implementar una semántica de uso compartido. La tabla de archivos abiertos por proceso permite mantener el estado del archivo de manera independiente por cada proceso. De tener solamente una tabla, habría que restringir el acceso a un archivo a máximo un proceso simultáneamente, o bien permitir que, sin un proceso lee una cantidad de *bytes* del archivo, otro proceso que también lo tenga abierto leería los siguientes *byte*.

3. **[14p]** Preguntas generales de redes.

3.1) **[8p]** Describa en una frase la **función principal** de los siguientes componentes de una red. Sea breve.

a) Router

R.

2p. Puede ser una de las siguientes: conectar dos o más subredes, ó enrutar paquetes a través de sus puertas.
Puntaje parcial para: conectar dispositivos a la red.
No es correcto: dar acceso a Internet, entregar direcciones IP.

b) Switch

R.

2p. Puede ser una de las siguientes: conectar dispositivos a una red, ó compartir acceso a una red entre múltiples dispositivos, ó extender una subred, ó conectar dos segmentos de una misma subred (*bridge*)
Puntaje parcial para: mandar mensajes entre dispositivos de una misma subred
No es correcto: convertir IPs a MAC o viceversa, asignar direcciones IP, enrutar paquetes.

c) IXP (*Internet-Exchange Point*)

R.

2p. Intercambiar tráfico ente las redes de distintos ISPs.
Puntaje parcial: conectar un ISP a Internet, o agrupar múltiples ISPs.
No es correcto: dar acceso a la red/Internet a usuarios individuales, transformar el tráfico de una red a otra, almacenar contenido de Internet (como una CDN)

d) Modem

R.

2p. Convertir señales analógicas a digitales y viceversa. También puede ser: convertir señal telefónica a señal de datos y viceversa.
Puntaje parcial: utilizar la señal telefónica para acceder a Internet, codificar información digital (sin mencionar la decodificación), usar una red DSL ó ADSL
No es correcto: conectarse a Internet o dar acceso a Internet a un usuario sin hacer mención a una señal analógica o telefónica.

3.2) **[3p]** Las velocidad de transmisión en los estándares WiFi han ido aumentando en los últimos años, pero siempre se encuentran por detrás de los estándares cableados. Mencione y justifique dos razones (distintas) que hacen que la comunicación WiFi sea más lenta que la comunicación por cable.

R.

Pueden ser: (1) el medio aéreo tiene mucha más pérdida por interferencia que el medio cableado, ya que no tiene protección, (2) el medio aéreo requiere más retransmisiones, ya que tiene más pérdidas, (3) el medio aéreo requiere una codificación con más redundancia, ya que puede perder más paquetes, (4) el medio aéreo puede sufrir más congestión porque no hay limitaciones a la cantidad de dispositivos que pueden intentar acceder, (5) el medio aéreo es sensible a obstáculos físicos ya que las ondas se disipan, (6) el medio aéreo utiliza *broadcast* en lugar de comunicación punto-a-punto.

3.3) **[3p]** Respecto a los modelos usados para construir redes, recordemos que éstas se construyen siguiendo un modelo de capas. Dado que la comunicación entre capas introduce una sobrecarga (*overhead*) en la

comunicación, ¿sería posible construir una red sin usar el modelo de capas (dicho de otro forma, usando solo una capa)? ¿habría alguna desventaja, o sería mejor?

R.

1p. Es posible construir una red sin usar el modelo de capas, y podría eventualmente ser más rápida que usando el modelo tradicional de capas.

2p. Sin embargo la construcción de una red así sería mucho más compleja ya que todos los protocolos deberían estar fusionados en un único protocolo que se encargue de direccionamiento, reconstrucción, codificación, transmisión, interpretación, etc y todas las preocupaciones que se han planteado de manera independiente en cada capa. Esto podría generar que existan múltiples redes que se diferencien en pequeñas partes del protocolo y que no podrían hablar entre sí.

4. **[16p]** Respecto de capa de aplicación.

4.1) **[4p]** La definición de un *socket* en capa de aplicación requiere, entre otros parámetros, especificar una dirección IP (o un nombre de dominio) y un puerto. Sin embargo, tanto la dirección IP como el puerto son conceptos relacionadas a capas inferiores. Si en el modelo de capas, cada capa tiene una responsabilidad separada de las demás, ¿por qué debo especificar esos parámetros al construir el *socket*?

R.

Se especifican porque son datos utilizados en las capas inferiores. Esos parámetros no los usará el protocolo de aplicación, pero sí los protocolos de transporte y red que son necesarios para que el protocolo de aplicación pueda funcionar.

Si bien es cierto que en el modelo de capas, cada una tiene una responsabilidad separada, también es cierto que cada una necesita parámetros específicos para funcionar.

4.2) **[4p]** El protocolo HTTP está definido como un protocolo *stateless* (sin estado). ¿Cómo consiguen los sitios web recordar información de los clientes? ¿Esto hace que HTTP deje de ser *stateless*? Justifique su respuesta.

R.

2p. Los sitios web pemiten recordar información de los clientes utilizando *cookies* que son almacenadas en los clientes, y que posteriormente los clientes se comprometen a enviar cada vez que se efectúa una nueva conexión. Opcionalmente los sitios web pueden mantener información complementaria de los clientes utilizando bases de datos.

2p. Esto no hace que HTTP deje de ser *stateless*. Es la construcción de clientes y servidores que hace que se produzca un comportamiento *stateful*, pero el protocolo mismo no considera el estado del usuario.

4.3) **[4p]** ¿Cuál es el rol del protocolo DNS en la comunicación por Internet? ¿Por qué este protocolo es opcional en el *stack* TCP/IP (o bien debería ser obligatorio)?

R.

2p. El rol del protocolo DNS es mantener el *mapping* entre nombres de dominio y direcciones IP.

2p. Es opcional porque no es necesario tener un nombre de dominio para establecer comunicación con otro nodo. Es posible utilizar sin nombres de dominio y utilizando sólo direcciones IP y los paquetes llegarían sin problemas ya que el protocolo IP no depende en nada de DNS. Solo sería más difícil para las personas hacer las asociaciones de los servidores ya que un nombre de dominio es más fácil de recordar.

4.4) **[4p]** ¿Por qué existen servidores DNS autoritativos y no autoritativos? ¿Podrían ser todos autoritativos o todos no autoritativos?

R.

2p. Existen para descentralizar el acceso a las entradas. Permiten definir responsabilidades entre distintas zonas. Los autoritativos mantienen respuesta definitivas para una zona, mientras que los no autoritativos mantienen respuestas rápidas, pero no necesariamente actualizadas.

2p. Si fueran todos autoritativos, entonces cada actualización debería ser propagadaa a todos los miembros de la red y tomaría mucho tiempo porque todos los nodos mantendrían la base de datos completa. Si fueran

todos no autoritativos no habría una fuente oficial para mantener las entradas, o bien habría solo una, y el protocolo se haría centralizado.

---

b = bit
B = Byte

i	2^i B
1	2 B
2	4 B
3	8 B
4	16 B
5	32 B

i	2^i B
6	64 B
7	128 B
8	256 B
9	512 B
10	1024 B

i	2^i B
10	1 KB
20	1 MB
30	1 GB
40	1 TB
50	1 PB