udp Escuela de Informática y Telecomunicaciones
FACULTAD DE INGENIERÍA

Sistema de archivos

Adín Ramírez
adin.ramirez@mail.udp.cl

Sistemas Operativos (CIT2003-1)
1er Semestre 2015

Adaptado de Dave Eckhardt

Puntos a discutir

- Interfaz del sistema de archivos
- Archivos y directorios
- Permisos y accesos

¿Qué es un archivo?

- Abstracción de almacenamiento persistente
  - Esconde detalles de los dispositivos de almacenamiento
    - Direccionamiento de sectores: CHS vs. LBA
    - SCSI vs. IDE
  - Esconde detalles de la localización (alojamiento) en el dispositivo de almacenamiento
- Agrupamiento lógico de los datos
  - Puede estar físicamente esparcido
- Programas, datos
- Alguna estructura interna

Atributos típicos

- Nombre: 14 caracteres? 8.3? 255?
  - Unicode? ASCII? 6-bit? RADIX-50?
- Identificador: número de archivo (usualmente interno)
- Tipo (o no): pista para saber que aplicación utilizar
- Localización: dispositivo, lista de bloque
- Tamaño: dos significados (próximamente)
- Protección: quién puede hacer qué
- Tiempo, fecha, última modificación: monitoreo, y limpieza

Atributos de archivos extendidos

- BSD Unix
  - `archived`
  - `nodump`
  - `append-only` (por el usuario/sistema operativo)
  - `immutable` (por el usuario/sistema operativo)
- MacOS
  - color del ícono
- Plan 9
  - Identifica el mutador más reciente

Operaciones en archivos

- **Crear**: localizar el espacio, entrar al directorio
- **Escribir**, **leer**: a menudo a través de un puntero o cursor de posición
- **Buscar**: ajustar la posición del puntero al siguiente acceso
- **Borrar**: remover del directorio, liberar el espacio
- **Truncar**: recorta algunos datos del final del archivo (caso común: todos los datos)
- **Agregar**: escribe al final del archivo (sincronización implícita)
- **Renombrar**: cambiar el nombre del archivo dentro del directorio, o mover el archivo entre directorios

I/O a un archivo

- Usuarios leerán/escribirán archivos
- Entonces, ¿cómo leemos de y escribimos hacia un archivo?
  - `read("README.doc", input_buffer, num_bytes);`
  - `read("README.doc", input_buffer, num_bytes, start_loc);`
- ¿Qué problemas ven con estas instrucciones?
- ¿Qué solución se les ocurre?

Estado después de abrir un archivo

- Caro de especificar el nombre de cada `read`/`write`
  - Operación basada en strings
  - Búsqueda en el directorio
- Agrega una operación `open`
  - Agrega un estado
- La estructura de `archivo-abierto` almacena
  - Sistema de archivos/partición
  - Número de archivo relativo al sistema de archivos
  - Operaciones permitidas, e.g., `read` vs. `write`
  - Posición del cursor
- ¿Falta algo más?

Estado “en el núcleo”

Abrir archivos: modelo UNIX

- Estado del archivo en el núcleo —evitar ir al disco repetidamente
  - Espejo de las estructuras del disco
    - Número de archivo, tamaño, permisos, tiempo de modificación, etc.
  - Información para mantener el sistema
    - Puntero de regreso al sistema de archivos que lo contiene
    - Puntero al dispositivo que aloja al archivo
    - Quién mantiene *locks* en los distintos rangos del archivo
  - Métodos de acceso al archivo (vector de métodos)
  - Puntero a los datos específicos del archivo
- Compartido cuando el archivo es abierto múltiples veces

Estado “abierto”

Abrir archivos: modelo UNIX

- Estado abierto —resultado de una llamada `open`
  - Los resultados obtenidos se mantienen para poder ser usados por múltiples llamadas
    - Puntero al estado en-el-núcleo del archivo
    - Credenciales de los procesos (cuando abrió el archivo)
    - Modo de acceso (leer vs. escribir, auto agregar, etc.)
    - Posición del cursor
- Compartido por múltiples archivos
  - Copiado por un `fork`
  - Heredado entre `execs`

Ejemplo

```c
int fd1, fd2, fd3;
off_t pos2, pos3;
char buf[10];

fd1 = open("foo.c", O_RDONLY, 0);
fd2 = dup(fd1);
fd3 = open("foo.c", O_RDONLY, 0);

read(fd1, &buf, sizeof (buf));

pos2 = lseek(fd2, 0L, SEEK_CUR); /* => ? */
pos3 = lseek(fd3, 0L, SEEK_CUR); /* => ? */
```

Ejemplo

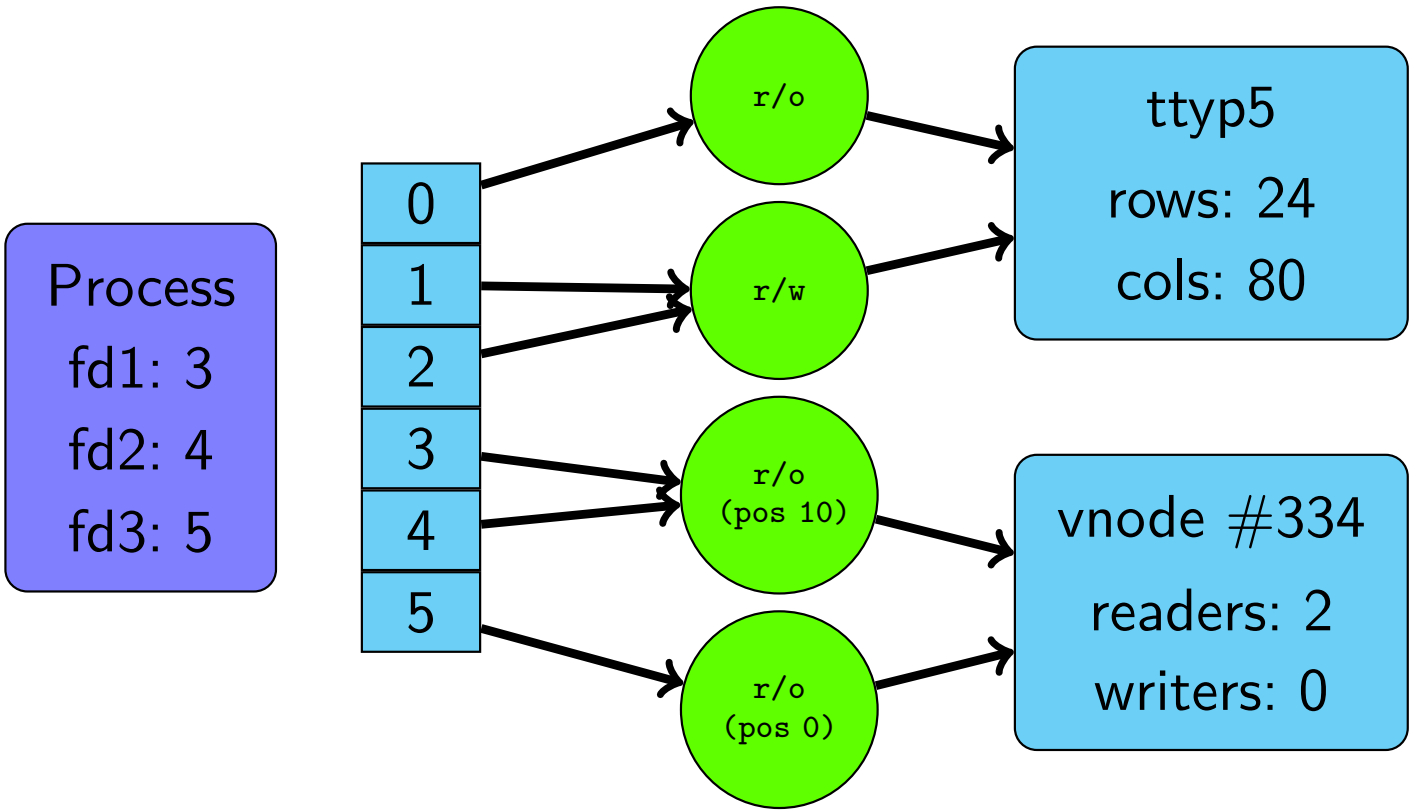
```c
int fd1, fd2, fd3;
off_t pos2, pos3;
char buf[10];

fd1 = open("foo.c", O_RDONLY, 0);
fd2 = dup(fd1);
fd3 = open("foo.c", O_RDONLY, 0);

read(fd1, &buf, sizeof (buf));

pos2 = lseek(fd2, 0L, SEEK_CUR); /* 10 */
pos3 = lseek(fd3, 0L, SEEK_CUR); /* 0 */
```

Estados abierto vs. en-núcleo



Tipos de archivo

- Meta
  - Evitar imprimir un archivo ejecutable binario
  - Encontrar un programa que entienda un archivo elegido por el usuario
- Derivar el "tipo" del archivo a partir de los nombres
  - `.exe` son ejecutables, `.c` son fuente de `C`
- Etiquetar un archivo con información (atributos extendidos)
  - MacOS: 4 byte para el tipo, e byte creador
- Unix: ambos y ninguno
  - Lo deja (en su mayoría) a los usuarios (tal vez: GUI, librerías, etc.)

Estructura de un archivo

- ¿Qué está dentro de un archivo?
  - Flujo de bytes?
  - Conjunto de caracteres: US-ASCII, Latin-1, Unicode, etc.
  - Flujo de registros?
  - Arreglo de registros? Árbol de registros?
- ¿Cuál es la estructura de un registro?
  - Fin de linea (CR, LF, CR+LF)
  - Longitud fija? Variable? Acotada?

Unix

- El cargador de programas necesita saber sobre los ejecutables
  - Números mágicos en los primeros dos bytes
  - Tipos obsoletos `A.out` —`OMAGIC`, `NMAGIC`, `ZMAGIC`[1]
  - `ELF`
  - `#!` —scripts (se recuerdan de las diapositivas de la explicación de shebang)
- De otra manera, arreglo de bytes
  - La aplicación o el usuario recuerda el significado
- Por un tiempo
  - Intentar el comando `file`
  - Leer `/usr/share/misc/magic` o `/usr/share/file/magic`
  - *Over 9000!!*: 16 000 lineas

[1] http://man.cat-v.org/unix_8th/5/a.out

MacOS “Clásico”

- Fork de datos
  - Arreglo de bytes
  - Estructura dependiente de la aplicación
- Fork de recursos
  - Tabla de recursos
    - Ícono, menú, ventana, diálogo
  - Muchos recursos son ampliamente usados y entendidos
    - Programa de escritorio despliega los íconos del fork de recursos
- Efectos emulados en OS X
  - La implementación puede variar

Métodos de acceso

- Proveídos por el sistema operativo o librería de programas opcional
- Secuencial
  - Como una cinta
  - `read()`, siguiente, `write()`, siguiente, `rewind()`
  - Algunas veces: saltar adelante o atrás
- Directo/relativo
  - Arreglo de registros de tamaño fijo
  - Leer y escribir cualquier registro por número

Indexado

- Archivo contiene registros
- Registros contienen llaves
- Índice mapea llaves $\Rightarrow$ registros
  - Ordena alguna porción de los datos según las llaves
  - Búsqueda binaria en lista multinivel
- Extensiones lujosas
  - Múltiples llaves, múltiples índices
  - ¿Ya tenemos una base de datos?
    - Nos faltan: relaciones, disparadores, consistencia, transacciones, etc.
  - Equivalente en Unix: `dbm`, `ndmb`, `gdbm`, `dbd`, etc.

Operaciones de directorios

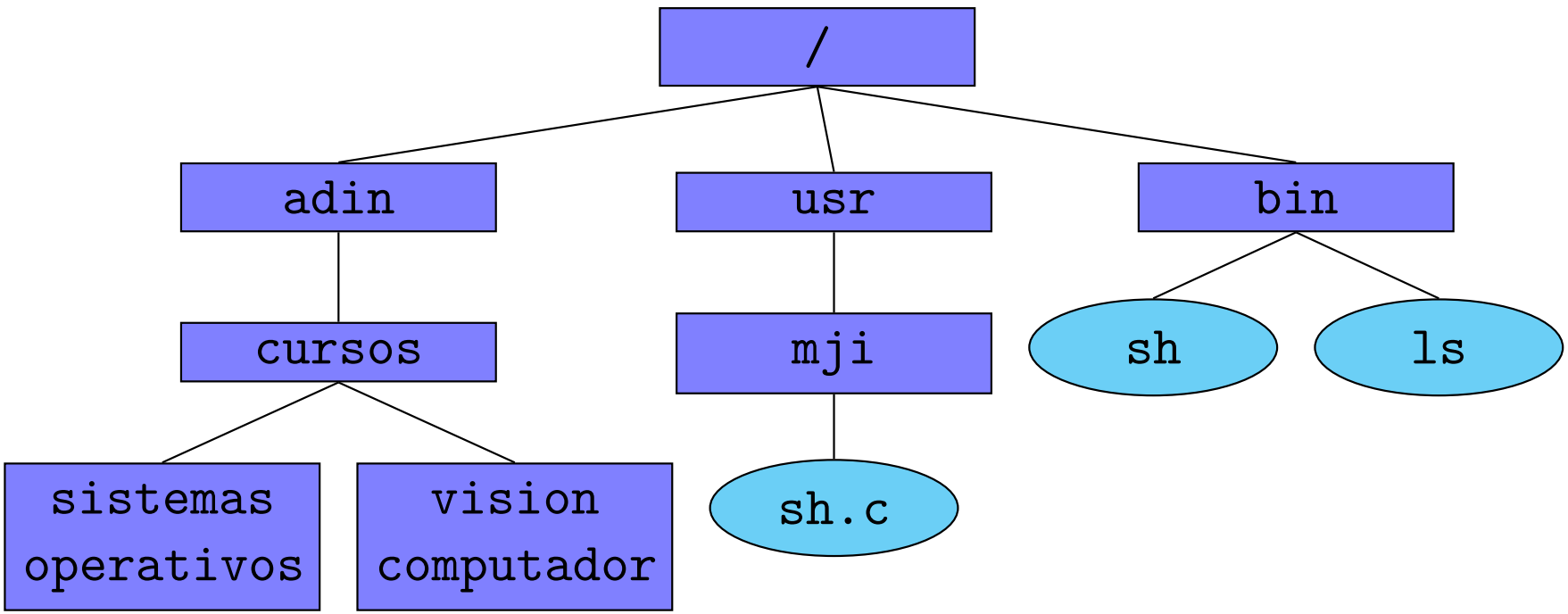
- `lookup("index.html")`
- `create("index.html")`
- `delete("index.html")`
- `rename("index.html", "index.html~")`
- Iterar sobre el contenido del directorio
- Escanear el sistema de archivos
  - Comando `find` de Unix
  - Programa de respaldo
- Bitácora de cambios en el árbol de directorios

Tipos de directorios

- Nivel simple
  - Espacio de nombres global plano —solo un `test.c`
  - Tal vez funcione en disquetes
- Nivel doble
  - Cada usuario tiene un directorio
  - Un `test.c` por usuario
  - Típico en sistemas compartidos anteriores

Árbol de directorios

- Dirección absoluta
  - Secuencia de nombres de directorio
  - Empieza por `root` (raíz)
  - Termina con un nombre de archivo

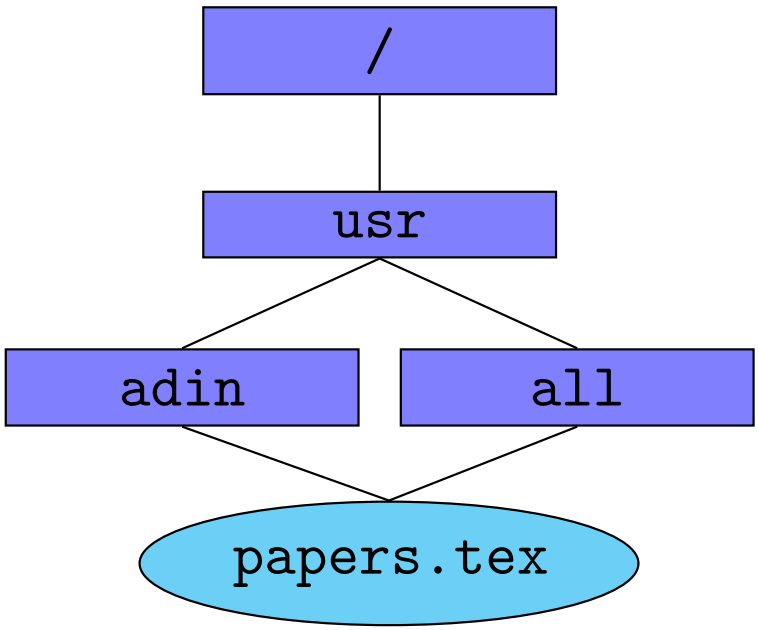


Árbol de directorios

- Los directorios son archivos especiales
  - Creados con llamadas de sistema especiales —`mkdir()`
  - Entiende de formatos y son mantenidos por el sistema operativo
- El directorio actual (`.`)
  - Sirve para saber donde estoy (e.g., `/usr/zzz`)
  - Inicio de direcciones relativas
    - `./cosas/foo.c` o `cosas/foo.c` ⇒ `/usr/zzz/cosas/foo.c`
    - `../cosas/foo.c` ⇒ `/usr/cosas/foo.c`
  - Referencia del directorio donde estamos, e.g., `p->p_fd->fd_cdir`

Directorios GDA

- GDA: Grafo dirigido acíclico
- Comparte archivos y directorios entre usuarios
- No es mio, no es tuyo, es nuestro
- Destruido cuando todos lo borran
- Unix *hard link*
  - ¿Eliminar un archivo abierto?
    - Se mantiene vivo hasta que la última persona lo cierre
  - Archivos, los directorios: problem



Soft links

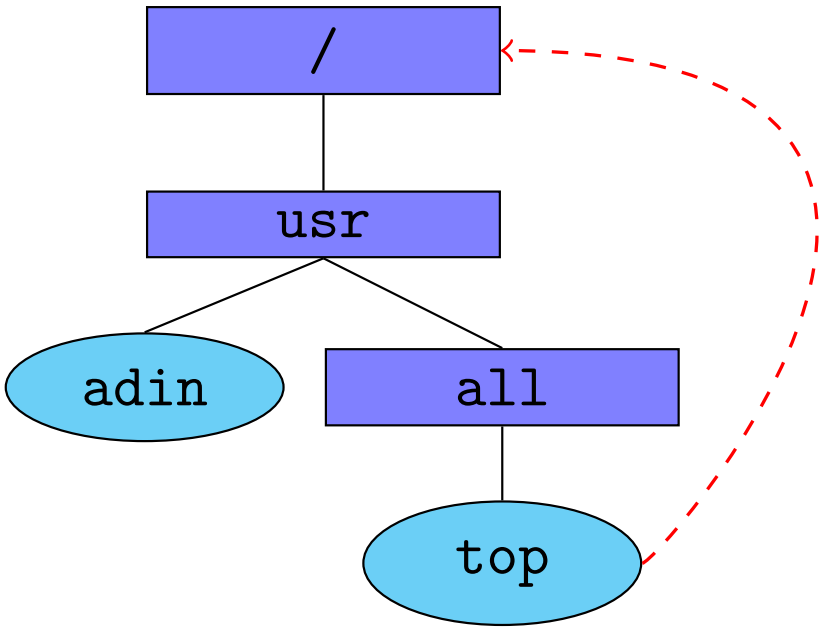
- *Hard links* muy duros
  - ¿Necesita un nivel de indirección en el sistema de archivos?
  - ¿Quiere un solo nombre *verdadero* para el archivo?
  - ¿Necesita cruzar a otro sistema de archivos de un tipo distinto?
- Alternativa: *soft link*, *symbolic link*, *shortcut*
  - Archivo pequeño, de un tipo especial
  - Contiene el nombre de otro archivo
  - SO dereferencia el link cuando lo abrimos: `open()`
    - Link puede apuntar a un archivo en cualquier parte
    - Un archivo en otro sistema de archivos de cualquier tipo
    - Un archivo remoto

Hard link vs. *soft link*

- Hard links
  - Permite compartir referencias con conteo (imaginen punteros inteligentes)
  - Ningún nombre es mejor que otro
- Soft links
  - Puede enlazar a un directorio
    - Hay solo un padre verdadero, así que no hay problemas
  - Trabaja entre sistemas de archivos y límites de máquina
  - Fácil de explicar
  - Problema del link colgante (*dangling link*)
    - El dueño del archivo verdadero puede borrarlo
    - El soft link ahora apunta a nada (☹)

Directorios GDA

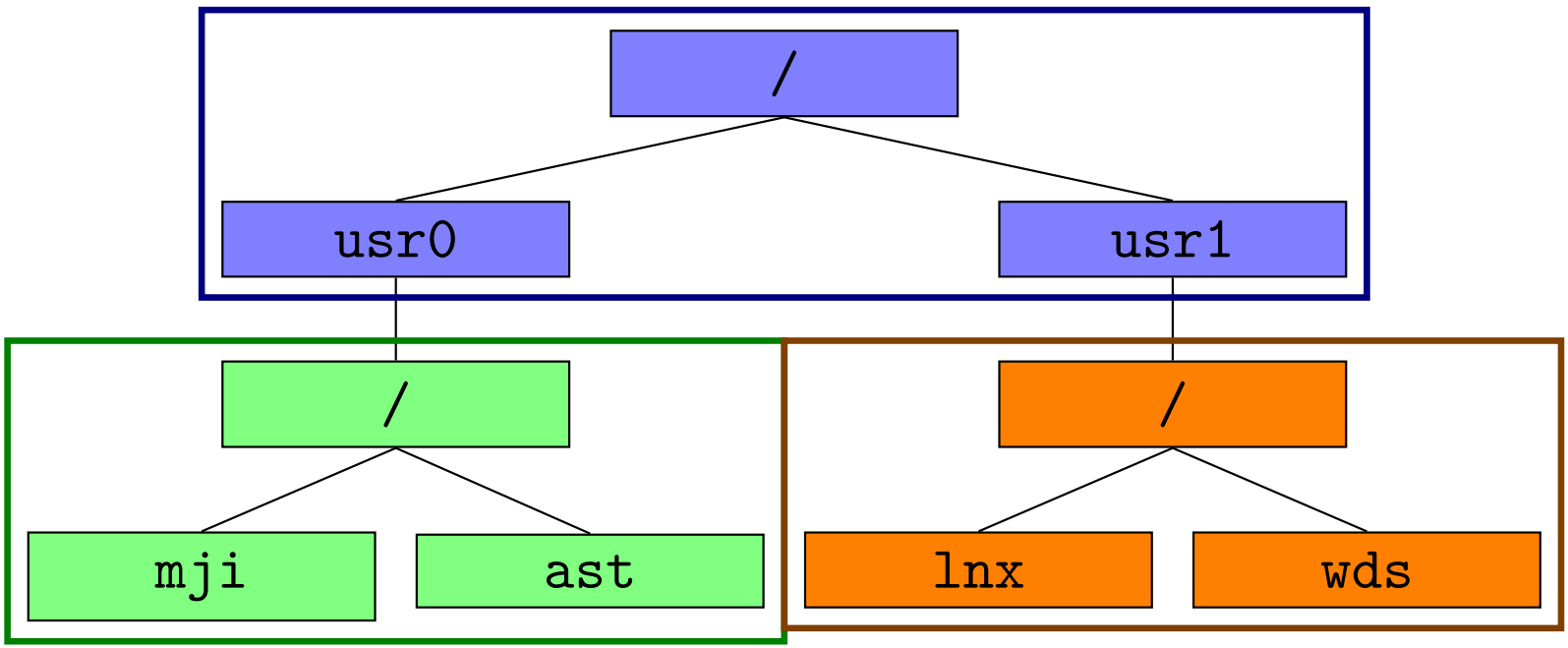
- Recorrerlo usando búsqueda en profundidad puede ser lento
- Podemos necesitar verdaderos recolectores (garbage collectors)
- ¿Realmente necesitamos ésto?



Mounting

- Múltiples discos en una computadora
- Múltiples particiones en un disco
- Un sistema de archivos dentro de una partición
  - O, dentro de un volumen, o volumen lógico, etc.
- ¿Cómo nombramos los archivos en otro sistema de archivos?
  - Manera incorrecta:
  - `C:\temp` vs. `D:\temp`

Sistemas montados



Múltiples usuarios

- Usuarios quieren compartir archivos
- ¿Qué es un usuario?
  - Los strings pueden ser difíciles de manejar
  - Los enteros son mejores para que el sistema operativo compare
  - Unix: *user ID* `uid`
  - Windows: *Security ID* `sid`
- ¿Qué es un grupo?
  - Un conjunto de usuarios
  - Típicamente tiene su propio identificador `gid` o `sid`

Protección

- Bit de sobreescritura (e.g., MS-DOG)
  - El bit dice “no borren este archivo”
  - A menos que se limpie el bit
- Contraseñas por archivo
  - Molestos cuando tenemos prisa
- Contraseñas por directorio
  - Aún molestas

Acceso

- Modos de acceso
  - Escribir, leer, ejecutar, agregar, borrar, listar, asegurar, etc.
- Lista de control de acceso (ACL)
  - El archivo almacena lista de usuarios y modos: tuplas `(user, mode)`
  - Molesto para almacenar, ver, y administrar
- Sistema de capacidad
  - Se le da al usuario una lista de tuplas `(file, access key)`
  - Problemas de revocación

Base típica

- El archivo especifica dueño y grupo
  - Permisos para el usuario, permisos para el grupo
  - Permisos para otros
- Codificación tradicional de Unix
  - `r`, `w`, `x` = 4, 2, 1
  - `rwx r-x -x` = 0751 en base octal
  - V7 Unix: 3 palabras de 16 bits especifican todos la información de permisos
    - Bits de permiso, número de usuario, número de grupo
    - 16 bits representan $2^{16} = 65\,536$, no son muchos usuarios!!

Resumen de archivos

- Abstracción de un disco o almacenamiento
  - Registros, no sectores
  - Tipo de información
- Nombres
  - Complejidad debido a la necesidad de tener enlaces
- Dueños y permisos
- Semántica de los distintos `open()`