Interfaz del sistema de archivos
11

Sistemas operativos y distribuidos

Gustavo Distel
gd@cs.uns.edu.ar

DCIC - UNS

Interfaz del sistema de archivos

Contenido

- Concepto de archivo.
- Métodos de acceso.
- Estructura del directorio.
- Protección.
- Archivos mapeados en memoria.

Interfaz del sistema de archivos

- El sistema de archivos consta de dos partes:
  - Una **colección de archivos**, cada uno de los cuales almacena datos relacionados.
  - Una **estructura de directorio**, que organiza y proporciona información sobre todos los archivos del sistema.
- El SO abstrae las propiedades físicas de los dispositivos de almacenamiento para definir el archivo como una unidad lógica de almacenamiento.
- Los archivos se mapean a dispositivos físicos, los cuales generalmente son no volátiles (***NVM***, ***HDDs***, **cintas magnéticas y discos ópticos**).
- Un archivo es una colección de información relacionada, asociada a un nombre que se graba en almacenamiento secundario, con un formato libre (como archivos de texto) o con un formato rígido.
- En general, un archivo es una secuencia de *bits*, *bytes*, líneas o registros, cuyo significado está definido por el creador y el usuario del archivo.

Interfaz del sistema de archivos

- Un archivo tiene una determinada estructura definida, que depende de su tipo.
  - **Archivo de texto**: es una secuencia de caracteres organizados en líneas (y quizás páginas).
  - **Archivo fuente**: es una secuencia de funciones, con declaraciones seguidas de sentencias ejecutables.
  - **Archivo ejecutable**: es una serie de secciones de código que el cargador puede llevar a memoria y ejecutar.

Interfaz del sistema de archivos

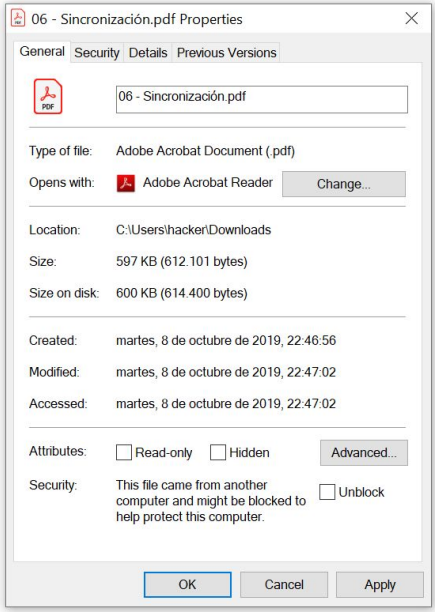
Atributos de los archivos

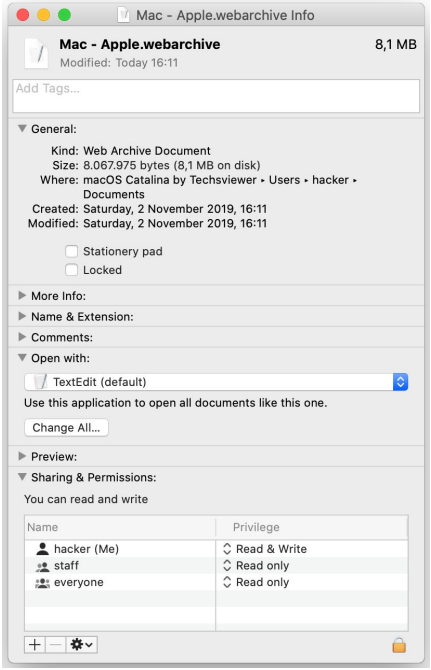
- Un archivo se nombra con una cadena de caracteres, como `i_wanna_rock.c`.
  - Algunos sistemas diferencian mayúsculas y minúsculas y otros no.
- Los atributos de un archivo varían de un SO a otro, pero generalmente son:
  - **Nombre**: información que se mantiene en forma legible para el ser humano.
  - **Identificador**: etiqueta única, generalmente un número, identifica el archivo en el sistema de archivos; *non-human-readable name*.
  - **`Tipo`**: información necesaria para sistemas que admiten diferentes tipos de archivos.
  - **Ubicación**: puntero a un dispositivo y ubicación del archivo en ese dispositivo.
  - **Tamaño**: tamaño del archivo (en *bytes*, *words* o bloques) y posiblemente el tamaño máximo permitido.
  - **Protección**: información de control de acceso; quién puede leer, escribir, ejecutar, etc.
  - **Marcas de tiempo e identificación de usuario**: información de su creación, última modificación y último uso. Estos datos pueden ser útiles para la protección, seguridad y monitoreo de uso.

Interfaz del sistema de archivos

Atributos de los archivos

- La información de todos los archivos se mantiene en la estructura de directorios, que reside en el mismo dispositivo que los archivos.
- Por lo general, una entrada de directorio consta del nombre del archivo y su identificador único. El identificador, a su vez, localiza los otros atributos del archivo.





Interfaz del sistema de archivos

Operaciones de archivo

- Un archivo es un tipo de datos abstracto, razón por la cual, para definirlo correctamente, se deben considerar las operaciones que se pueden realizar sobre estos.
  - Crear un archivo.
  - Abrir un archivo.
  - Escribir un archivo.
  - Leer un archivo.
  - Reposicionamiento en un archivo.
  - Borrar/Eliminar un archivo.
  - Truncar un archivo.

Interfaz del sistema de archivos

Operaciones de archivo

- El SO mantiene una tabla, llamada `tabla de archivos abiertos`, que contiene información sobre todos los archivos abiertos.
- Cuando se solicita una operación de archivo, el archivo se especifica a través de un índice en esta tabla, por lo que no se requiere búsqueda.
- Cuando el archivo ya no se usa activamente, el proceso lo cierra y el SO elimina su entrada de la tabla de archivos abiertos, liberando potenciales bloqueos.
- Normalmente, el SO utiliza dos niveles de tablas internas: una `tabla por proceso` y `una tabla de todo el sistema`.
- La tabla por proceso rastrea todos los archivos que un proceso ha abierto. En esta tabla se almacena información del uso del proceso sobre el archivo.

Interfaz del sistema de archivos

Operaciones de archivo

- Cada entrada en la tabla por proceso apunta, a su vez, a una tabla de archivos abiertos en todo el sistema.
- La tabla de todo el sistema contiene información independiente del proceso, como la ubicación del archivo en el disco, las fechas de acceso y el tamaño del archivo.
- Una vez que un archivo ha sido abierto por un proceso, la tabla de todo el sistema incluye una entrada para ese archivo.
- Cuando otro proceso ejecuta una llamada **`open()`**, simplemente se agrega una nueva entrada a la tabla de archivos abiertos del proceso que apunta a la entrada en la tabla de todo el sistema.
- Por lo general, la tabla de archivos abiertos tiene un ***`open count`*** asociado con cada archivo, para indicar cuántos procesos lo han abierto.
    - Cada **`close()`** lo decrementa y al llegar a cero, el archivo ya no estará en uso y se elimina la entrada de la tabla de archivos abiertos.

Interfaz del sistema de archivos

Operaciones de archivo

- En resumen, un archivo abierto cuenta con la siguiente información asociada:
  - **Puntero de archivo**: mantiene la última ubicación de lectura-escritura. Este puntero es único para cada proceso que opera en el archivo y, por lo tanto, debe mantenerse separado de los atributos del archivo en el disco.
  - **Contador de archivos abiertos**.
  - **Ubicación del archivo**: información necesaria para ubicar el archivo, que se guarda en memoria para que el sistema, en cada operación, no tenga que leerla de la estructura de directorio.
  - **Derechos de acceso**: cada proceso abre un archivo en un modo de acceso. Esta información se almacena en una tabla por proceso, así el SO puede permitir o denegar posteriores solicitudes de `E/S`.

Interfaz del sistema de archivos

Operaciones de archivo

- Algunos SOs proporcionan facilidades para bloquear un archivo abierto:
  - ***Shared lock***: similar a un *lock* de lector; varios procesos pueden adquirirlo al mismo tiempo.
  - ***Exclusive lock***: similar a un *lock* escritor; solo puede adquirirlo un proceso a la vez.
- Además, se pueden proporcionar bloqueos obligatorios (*mandatory*) o de aviso (*advisory*).
- **Bloqueo obligatorio**: una vez que un proceso adquiere un bloqueo exclusivo, el SO evitará que cualquier otro proceso acceda al archivo bloqueado.
    - Por ej., si un proceso adquiere un bloqueo exclusivo en el archivo `system.log`, si se intenta abrir desde otro proceso (como un editor de texto) el SO impedirá el acceso hasta que se libere el bloqueo exclusivo.

Interfaz del sistema de archivos

Operaciones de archivo

- **Bloqueo de aviso**: es responsabilidad de los desarrolladores asegurarse de que los bloqueos se adquieran y liberen adecuadamente.
    - El SO no impedirá que el editor de texto obtenga acceso a `system.log`.
    - En realidad, el editor de texto debe programarse de modo que adquiera manualmente el bloqueo antes de acceder al archivo, a diferencia del bloqueo obligatorio que es el SO el que garantiza la integridad del bloqueo.
- Como regla general, los SOs `Windows` adoptan el bloqueo obligatorio, y los sistemas `UNIX` emplean bloqueos de aviso.

Interfaz del sistema de archivos

Tipos de archivo

- Una técnica común para implementar tipos de archivo es incluir el tipo como parte del nombre del archivo.
  - El nombre se divide en dos partes: un nombre y una extensión, generalmente separados por un punto (`i_wanna_rock.mp3`).
  - De esta manera, el usuario y el SO pueden saber solo por el nombre qué tipo de archivo es.
- El sistema usa la extensión para indicar el tipo de archivo y el tipo de operaciones que se pueden realizar en él.
- El sistema UNIX utiliza un `número mágico` almacenado al principio de algunos archivos binarios para indicar el tipo de datos del archivo (por ej., el formato de un archivo de imagen).
- Las extensiones pueden ser utilizadas o ignoradas por una aplicación determinada, pero eso depende del programador de la aplicación.

Interfaz del sistema de archivos

Estructura de archivo

- Los tipos de archivo también se pueden usar para indicar la estructura interna del archivo.
- Algunos SOs soportan un conjunto de estructuras de archivos compatibles con el sistema, con operaciones especiales para manipular archivos con esas estructuras.
- Algunos SOs imponen (y admiten) un número mínimo de estructuras de archivos. Este enfoque ha sido adoptado en **`UNIX`**, **Windows** y otros.
- **`UNIX`** considera cada archivo como una secuencia de **`8-`*`bits bytes`***; El SO no realiza ninguna interpretación de estos *bits*.
- Este esquema proporciona la máxima flexibilidad pero poco soporte. Cada programa debe incluir su propio código para interpretar un archivo.
- Sin embargo, todos los SOs deben admitir al menos una estructura, la de un archivo ejecutable, para que el sistema pueda cargar y ejecutar programas.

Interfaz del sistema de archivos

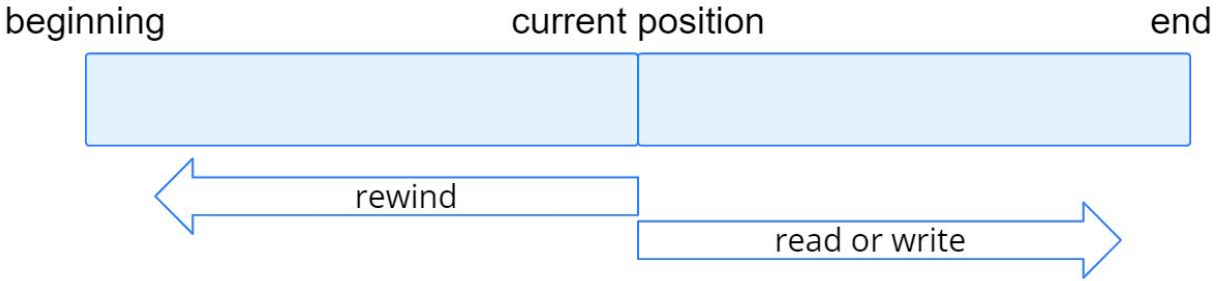
Estructura interna de archivo

- Los discos tienen un tamaño de bloque bien definido, determinado por el tamaño de un sector.
- Todas las `E/S` de disco se realizan en unidades de un bloque (registro físico), y todos los bloques son del mismo tamaño.
- El tamaño del `registro lógico`, el tamaño del `bloque físico` y la `técnica de empaquetado` determinan cuántos registros lógicos hay en cada bloque físico.
    - El empaque puede realizarse por el programa del usuario o por el SO.
- En cualquier caso, el archivo puede considerarse una secuencia de bloques. Todas las funciones básicas de `E/S` operan en términos de bloques.
- Debido a que el espacio en disco siempre se asigna en bloques, una parte del último bloque de cada archivo generalmente se desperdicia (fragmentación interna).

Métodos de acceso

Acceso secuencial

- La información en el archivo se procesa en orden, un registro después del otro.
- Este modo de acceso es el más común; por ej., los editores y compiladores suelen acceder a los archivos de esta manera.
- El acceso secuencial se basa en un modelo de cinta de un archivo y funciona bien tanto en dispositivos de acceso secuencial como en los de acceso aleatorio.



- Operaciones:
  - `read_next()`
  - `write_next()`

Métodos de acceso

Acceso directo

- Un archivo está formado por registros lógicos de longitud fija que permiten a los programas leer y escribir registros rápidamente sin ningún orden en particular.
- Se basa en un modelo de disco, ya que los discos permiten el acceso aleatorio a cualquier bloque de archivos.
- El archivo se ve como una secuencia numerada de bloques o registros.
- Por lo tanto, podemos leer el bloque 14, luego leer el bloque 53 y luego escribir el bloque 7. No hay restricciones en el orden de lectura o escritura.
- Los archivos de acceso directo son de gran utilidad para el acceso inmediato a grandes cantidades de información. Las bases de datos son a menudo de este tipo.
- Operaciones:
  - `read(n)`.
  - `write(n)`.

Métodos de acceso

Otros métodos de acceso

- Se pueden construir otros métodos de acceso sobre un método de acceso directo. Estos métodos generalmente implican la construcción de un índice para el archivo.
- El índice, como un índice en la parte posterior de un libro, contiene punteros a los distintos bloques.
- Para encontrar un registro en el archivo, primero buscamos el índice y luego usamos el puntero para acceder al archivo directamente y encontrar el registro deseado.

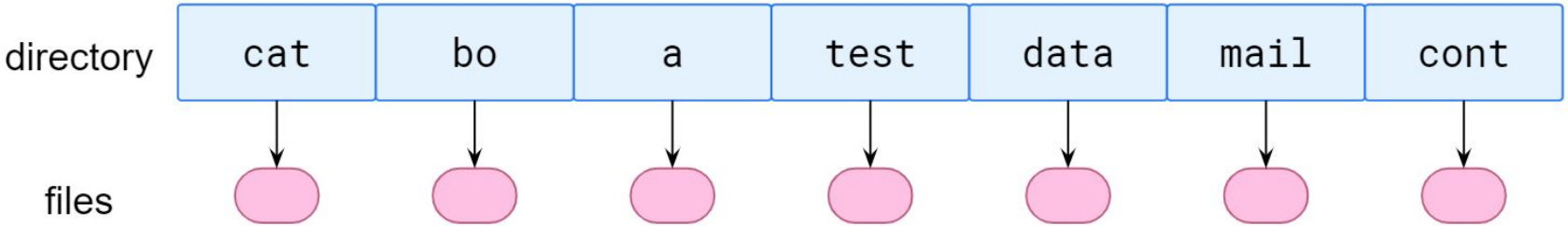
Estructura de directorios

- El directorio puede considerarse como una tabla de símbolos que traduce nombres de archivo en sus bloques de control.
- Al considerar una estructura de directorio en particular, se debe tener en cuenta las operaciones que se realizarán:
  - Buscar un archivo.
  - Crear un archivo.
  - Eliminar un archivo.
  - Listar un directorio.
  - Cambiar el nombre de un archivo.
  - Recorrer el sistema de archivos.

Estructura de directorios

Directorio de un solo nivel

- Todos los archivos están contenidos en el mismo directorio, que es fácil de mantener y comprender.

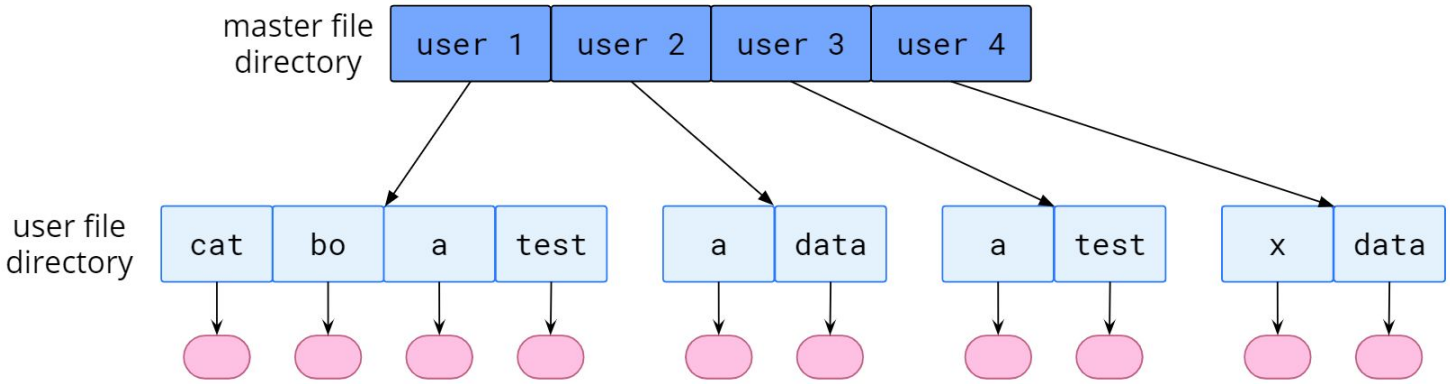


- Sin embargo, un directorio de un solo nivel tiene limitaciones significativas cuando aumenta el número de archivos o cuando el sistema tiene más de un usuario. Como todos los archivos están en el mismo directorio, deben tener nombres únicos.
- La mayoría de los sistemas de archivos admiten nombres de hasta `255` caracteres (`MS-DOS` permitia `11`), por lo que es relativamente fácil seleccionar nombres de archivo únicos.
- Incluso un usuario con directorios de un solo nivel puede tener dificultades para recordar los nombres de todos los archivos a medida que aumenta su número.

Estructura de directorios

Directorio de dos niveles

- En la estructura de directorios de dos niveles, cada usuario tiene su propio directorio de usuario (***UFD*** - *user file directory*). Los ***UFD*** tienen estructuras similares, pero cada uno tiene los archivos de un solo usuario.
- Cuando se inicia un trabajo de usuario o un usuario inicia sesión, se busca en el directorio maestro de archivos (***MFD*** - *master file directory*) del sistema.
- El ***MFD*** se indexa por nombre de usuario o número de cuenta, y cada entrada apunta al **UFD** de dicho usuario



Estructura de directorios

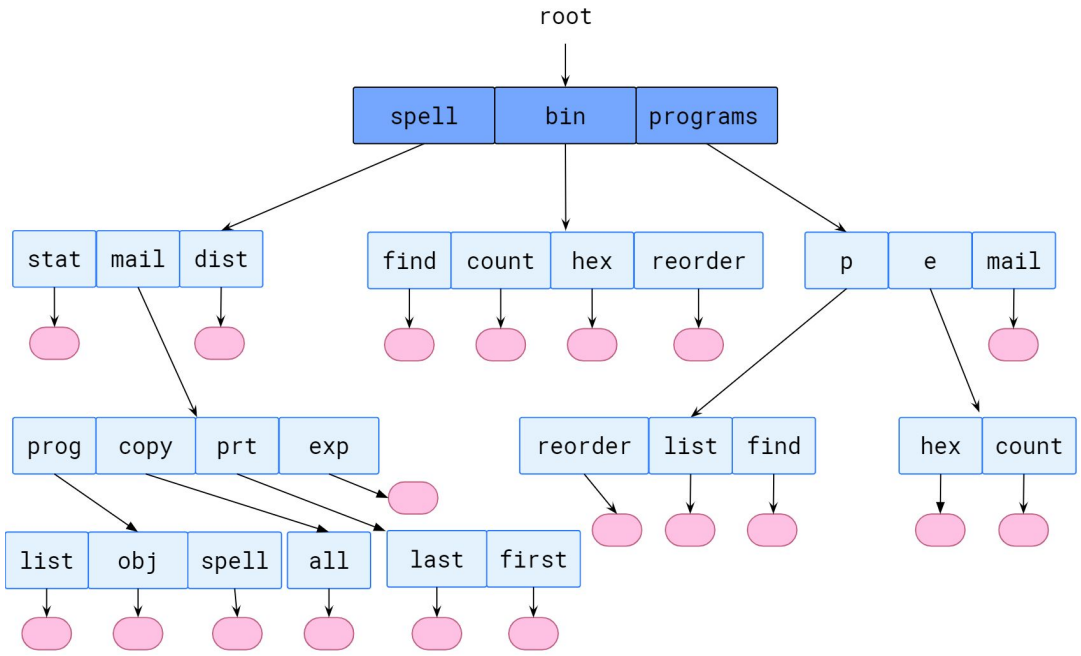
Directorio de dos niveles

- Aunque la estructura de directorios de dos niveles resuelve el problema de colisión de nombres, aún tiene desventajas. Esta estructura aísla efectivamente a un usuario de otro.
- El aislamiento es una ventaja cuando los usuarios son completamente independientes, pero es una desventaja cuando los usuarios desean cooperar en alguna tarea y acceder a los archivos de los demás.
- Algunos sistemas simplemente no permiten que un usuario acceda a los archivos locales de otro usuario.

Estructura de directorios

Directorios estructurados en árbol

- Los árboles son la estructura de directorios más común.
- El árbol tiene un directorio raíz, y cada archivo en el sistema tiene una única ruta.
- Un directorio (o subdirectorio) contiene un conjunto de archivos o subdirectorios.
- Un directorio es simplemente otro archivo, pero se trata de manera especial.



- Un bit en cada entrada del directorio define la entrada como un archivo (`0`) o como un subdirectorio (`1`).

Estructura de directorios

Directorios estructurados en árbol

- Cada proceso tiene un directorio actual (`current directory`).
- El directorio actual debe contener la mayoría de archivos que son de interés actual para el proceso.
- Cuando se hace referencia a un archivo, se busca en el directorio actual.
- Si se necesita un archivo que no está en el directorio actual, entonces el usuario generalmente debe especificar un nombre de ruta o cambiar el directorio actual para indicar el directorio que contenga ese archivo.
- Cuando se inicia un trabajo de usuario o cuando el usuario inicia sesión en la ***`shell`***, se establece un directorio actual.
- A partir de esa ***`shell`***, se pueden generar otros procesos. El directorio actual de cualquier subproceso suele ser el directorio del padre cuando fue creado.

Estructura de directorios

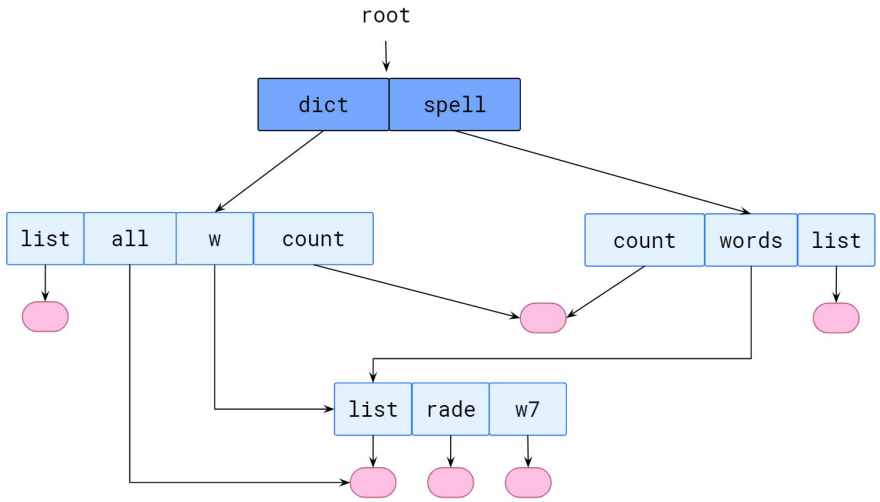
Directorios estructurados en árbol

- Los nombres de ruta pueden ser de dos tipos: **`absolutos`** y **`relativos`**.
  - En **`UNIX`** y **`Linux`**, un nombre de ruta absoluto comienza en la raíz (que se designa con una "`/`" inicial) y sigue una ruta hasta el archivo especificado, indicando los nombres de directorio en la ruta.
  - Un nombre de ruta relativo define una ruta desde el directorio actual.
    - Por ej., si el directorio actual es **`/spell/mail`**, el nombre de ruta relativo **`prt/first`** se refiere al mismo archivo que el nombre de ruta absoluto **`/spell/mail/prt/first`**.
- Con esta estructura se puede permitir a los usuarios acceder a los archivos de otros usuarios.
  - Por ej., el usuario `B` puede acceder a un archivo del usuario `A` especificando la ruta.
  - El usuario `B` puede especificar una ruta absoluta o relativa. Alternativamente, el usuario `B` puede cambiar su directorio actual para que sea el directorio del usuario `A` y acceder al archivo por su nombre.

Estructura de directorios

Directorios de grafos acíclicos

- Una estructura de árbol prohíbe compartir archivos o directorios.
  - Cada directorio o archivo compartido existirá en el sistema de archivos en dos (o más) lugares simultáneamente.
- Un grafo acíclico permite a los directorios compartir subdirectorios y archivos.
- El mismo archivo o subdirectorio puede estar en dos directorios diferentes.



Estructura de directorios

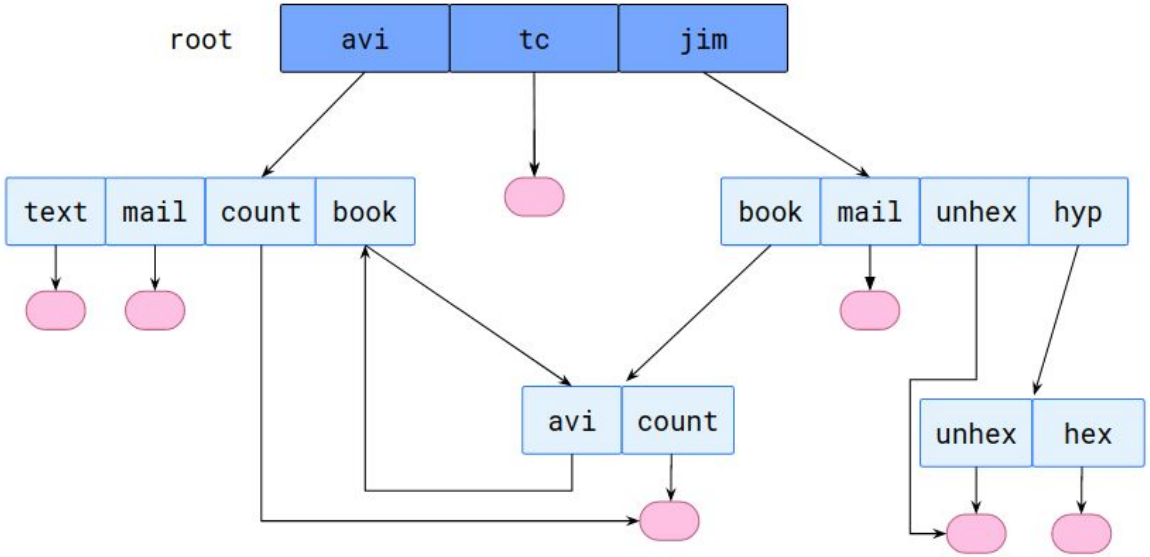
Directorios de grafos acíclicos

- Los archivos y subdirectorios compartidos se pueden implementar de varias maneras; en `UNIX` por ej., se crea una nueva entrada de directorio llamada enlace (*link*).
- Un enlace es, en la práctica, un puntero a otro archivo o subdirectorio.
- Por ej., un enlace puede implementarse como un nombre de ruta absoluto o relativo. Cuando se hace una referencia a un archivo, buscamos en el directorio. Si la entrada del directorio está marcada como un enlace, el nombre del archivo real se incluye en la información del enlace.
- Resolvemos el enlace utilizando ese nombre de ruta para localizar el archivo real.
- El SO ignora estos enlaces al atravesar árboles de directorios para preservar la estructura acíclica del sistema.
- Una estructura de directorio de grafo acíclico es más flexible que una estructura de árbol simple, pero también más compleja. Varios problemas deben considerarse:
    - Un archivo puede tener múltiples nombres de ruta absolutos.
    - Otro problema implica la eliminación, dado que pueden quedar punteros colgados.

Estructura de directorios

Directorios de grafos general

- Cuando se agregan enlaces a un estructura de árbol ésta se destruye, lo que resulta en una estructura de grafo simple.



- Si se permite que existan ciclos en los directorios, se debe evitar buscar un componente dos veces, por razones de corrección y rendimiento. Un alg. mal diseñado podría dar como resultado un bucle infinito que busca continuamente a través del ciclo y nunca termina.

Estructura de directorios

Directorios de grafos general

- Existe un problema similar cuando intentamos determinar cuándo se puede eliminar un archivo.
- Con los grafos acíclicos, un valor de `0` en las referencias significa que no hay más referencias al archivo o directorio, y el archivo se puede eliminar.
- Sin embargo, cuando existen ciclos, la cantidad de referencias puede no ser `0` incluso cuando ya no sea posible hacer referencia a un directorio o archivo. Esta anomalía resulta de la posibilidad de autorreferencia (o un ciclo) en la estructura del directorio.
- En estos casos generalmente se necesita usar un *garbage collection*.
- Un *garbage collector* recorre todo el sistema de archivos, marcando todo lo que se puede acceder. Luego, en una segunda pasada recoge todo lo que no está marcado en una lista de espacio libre.
- Un *garbage collector* consume mucho tiempo y, por lo tanto, rara vez se utiliza.

Protección

- La información en un sistema informático se debe resguardar de daños físicos (fiabilidad - *reliability*) y del acceso inadecuado (protección - *protection*).
- La confiabilidad generalmente se logra teniendo copias duplicadas de archivos.
- La protección se puede proporcionar de varias maneras.
  - Para una *notebook*, se puede brindar protección al requerir nombre de usuario y contraseña para autenticación en el SO, encriptar el almacenamiento secundario y utilizar *firewall*.
  - En un sistema multiusuario se necesitan mecanismos más avanzados para permitir solo el acceso válido de los datos.

Protección

Tipos de accesos

- Los mecanismos de protección proporcionan acceso controlado al limitar los tipos de acceso que se pueden realizar.
- Se pueden controlar varios tipos de operaciones:
  - **Leer**: leer del archivo.
  - **Escribir**: escribir o reescribir el archivo.
  - **Ejecutar**: cargar el archivo en la memoria y ejecutarlo.
  - **Añadir**: escribir nueva información al final del archivo.
  - **Eliminar**: eliminar el archivo y liberar espacio para su posible reutilización.
  - **Lista**: listar el nombre y los atributos del archivo.
  - **Cambio de atributo**: cambiar los atributos del archivo.

Protección

Control de acceso

- El esquema más general es asociar a cada archivo y directorio una lista de control de acceso (*access control list - ACL*) que especifique los nombres de usuario y los tipos de acceso permitidos para cada usuario.
- Cuando un usuario solicita acceso a un archivo en particular, el SO verifica la lista de acceso asociada con ese archivo.
  - Si ese usuario aparece en la lista para el acceso solicitado, se permite el acceso.
  - De lo contrario, se produce una violación de protección y el usuario no tendrá acceso al archivo.
- Este enfoque tiene la ventaja de permitir metodologías de acceso complejas.
- El principal problema con las listas de acceso es su longitud. Si queremos permitir que todos lean un archivo, debemos enumerar todos los usuarios con acceso de lectura.

Protección

Control de acceso

- Esta técnica tiene dos consecuencias indeseables:
  - La construcción de la lista puede ser una tarea tediosa, especialmente si no conocemos de antemano la lista de usuarios en el sistema.
  - La entrada del directorio, anteriormente de tamaño fijo, ahora debe ser de tamaño variable, lo que resulta en una administración de espacio más complicada.
- Estos problemas pueden resolverse mediante el uso de una versión condensada de la lista de acceso; muchos sistemas reconocen tres clasificaciones de los usuarios:
  - **Propietario**: el usuario que creó el archivo es el propietario.
  - **Grupo**: un conjunto de usuarios que comparten el archivo y necesitan acceso similar conforman un grupo o grupo de trabajo.
  - **Otros**: todos los demás usuarios en el sistema.
- Un enfoque común es combinar *ACLs* con el esquema de control de acceso de propietario, grupo y universo.

Protección

Control de acceso: permisos en Linux

```
[hacker@localhost etc]$ ls -la | head -n 20
total 2176
drwxr-xr-x. 164 root root     12288 Oct 27 12:58 .
dr-xr-xr-x.  18 root root      4096 Sep 25 22:58 ..
drwxr-xr-x.   3 root root      4096 Aug  1 09:35 abrt
-rw-r--r--.   1 root root        16 Oct 13  2016 adjtime
-rw-r--r--.   1 root root      1529 Oct  9 05:18 aliases
drwxr-xr-x.   3 root root      4096 Jul 24 14:50 alsa
drwxr-xr-x.   2 root root      4096 Oct 27 12:57 alternatives
drwxr-xr-x.   4 root root      4096 Oct 14 09:52 anaconda
-rw-r--r--.   1 root root       541 Jul 24 18:15 anacrontab
-rw-r--r--.   1 root root       769 Jun 16 17:33 appstream.conf
-rw-r--r--.   1 root root        55 Jul 24 14:49 asound.conf
-rw-r--r--.   1 root root         1 Jul 24 15:43 at.deny
-rw-r--r--.   1 root root       186 Jul 25 20:50 atmsigd.conf
drwxr-x---.   4 root root      4096 Aug  1 18:28 audit
drwxr-xr-x.   3 root root      4096 Jul 24 15:54 authselect
drwxr-xr-x.   4 root root      4096 Aug 22 16:15 avahi
drwxr-xr-x.   2 root root      4096 Sep 25 23:09 bash_completion.d
-rw-r--r--.   1 root root      3019 Oct  9 05:18 bashrc
-rw-r--r--.   1 root root       429 Sep  5 10:56 bindresvport.blacklist
```

Protección

Control de acceso: permisos en FreeNAS (FreeBSD)

Shell

```
-rw-r--r--   1 root  wheel   119 Sep 19 14:36 .warning
-rw-r--r--   1 root  wheel    88 Sep 19 14:36 .zlogin
-rw-r--r--   1 root  wheel   634 Sep 19 14:36 .zshrc
root@freenasdcic:~ # cd /mnt/
root@freenasdcic:/mnt # ls
disknasdcic     md_size
root@freenasdcic:/mnt # cd disknasdcic/
root@freenasdcic:/mnt/disknasdcic # ls -la
total 43
drwxr-xr-x+  6 DCIC\administrator  DCIC\domain users    7 May 28 17:55 .
drwxr-xr-x   3 root                wheel              128 Oct 24 13:54 ..
drwxr-xr-x   8 root                wheel               10 Mar 16  2019 .system
-rwxrwxr-x+  1 DCIC\administrator  DCIC\domain users    0 Apr 16  2015 .windows
drwx------+ 17 DCIC\administrator  DCIC\domain users   17 Jun  4 15:02 Backups
drwxr-xr-x   9 root                wheel               10 Oct 24 13:55 iocage
drwxr-xr-x+ 10 DCIC\administrator  DCIC\domain users   10 Aug 12  2015 Publico
root@freenasdcic:/mnt/disknasdcic # getfacl ./Backups/
# file: ./Backups/
# owner: DCIC\administrator
# group: DCIC\domain users
group:DCIC\gdistel:rwxpDdaARWcCo-:fd-----:allow
            owner@:rwxpDdaARWcCo-:fd-----:allow
group:DCIC\backups:rwxpDdaARWcCo-:fd-----:allow
root@freenasdcic:/mnt/disknasdcic #
```

Protección

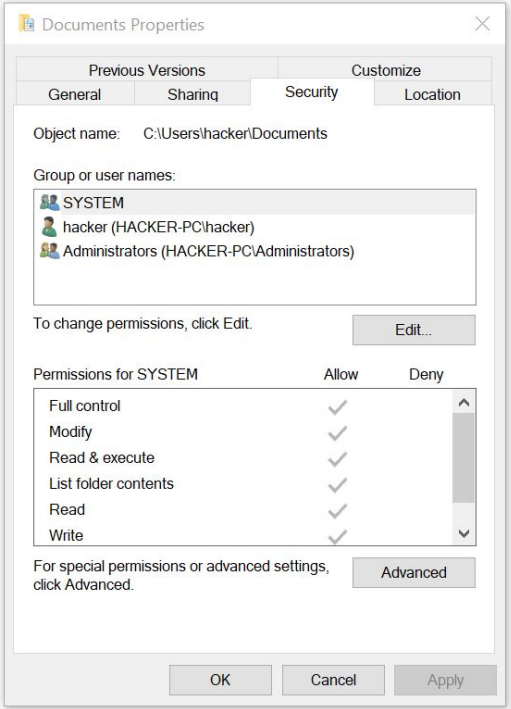
Control de acceso: permisos en masOS Catalina

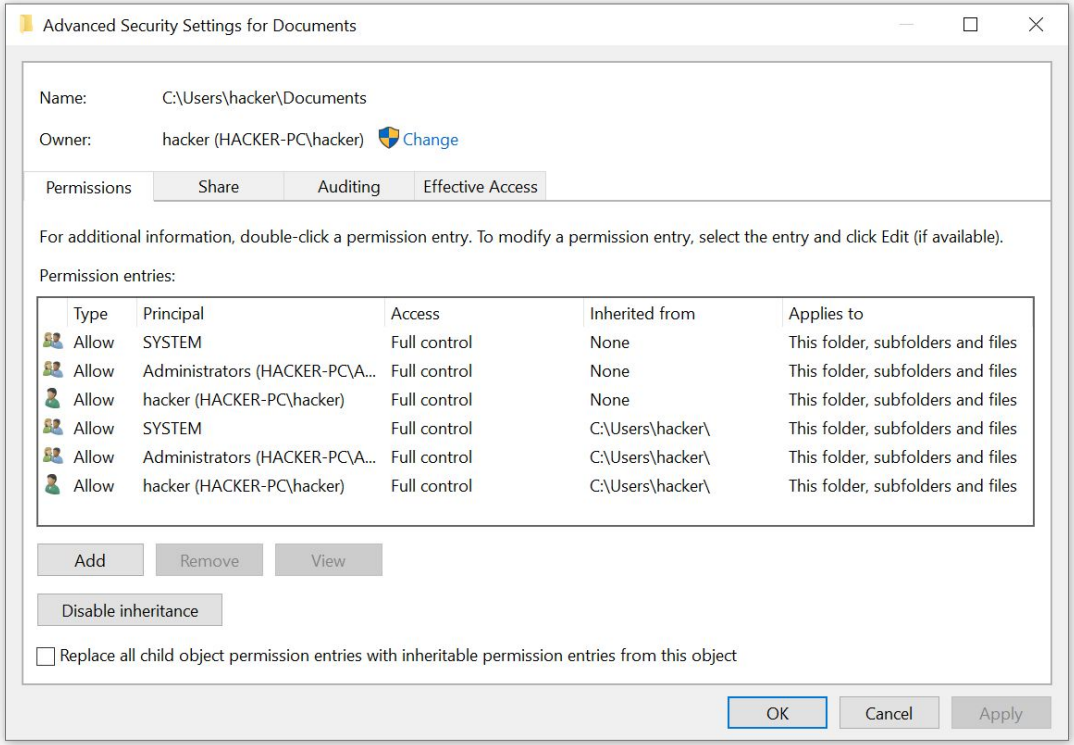
Terminal — -zsh — 118×36

```
Last login: Wed Nov  6 18:54:00 on ttys000
hacker@hackers-Mac ~ % ls
Desktop         Downloads       Movies          Pictures
Documents       Library         Music           Public
hacker@hackers-Mac ~ % ls -la
total 40
drwxr-xr-x+ 14 hacker  staff    448 Nov  6 18:54 .
drwxr-xr-x   4 root    admin    128 Jul  4 07:16 ..
-r--------   1 hacker  staff      7 Jul  4 07:16 .CFUserTextEncoding
-rw-r--r--@  1 hacker  staff  10244 Nov  2 16:12 .DS_Store
drwx------   2 hacker  staff     64 Nov  6 18:07 .Trash
-rw-------   1 hacker  staff    224 Nov  6 18:54 .zsh_history
drwx------+  3 hacker  staff     96 Jul  4 07:16 Desktop
drwx------+  4 hacker  staff    128 Nov  2 16:11 Documents
drwx------+  3 hacker  staff     96 Jul  4 07:16 Downloads
drwx------@ 55 hacker  staff   1760 Nov  2 16:16 Library
drwx------+  4 hacker  staff    128 Jul  8 16:08 Movies
drwx------+  5 hacker  staff    160 Jul  8 16:08 Music
drwx------+  5 hacker  staff    160 Jul  8 16:38 Pictures
drwxr-xr-x+  4 hacker  staff    128 Jul  4 07:16 Public
hacker@hackers-Mac ~ % ls -lae
total 40
drwxr-xr-x+ 14 hacker  staff    448 Nov  6 18:54 .
 0: group:everyone deny delete
drwxr-xr-x   4 root    admin    128 Jul  4 07:16 ..
-r--------   1 hacker  staff      7 Jul  4 07:16 .CFUserTextEncoding
-rw-r--r--@  1 hacker  staff  10244 Nov  2 16:12 .DS_Store
drwx------   2 hacker  staff     64 Nov  6 18:07 .Trash
-rw-------   1 hacker  staff    224 Nov  6 18:54 .zsh_history
drwx------+  3 hacker  staff     96 Jul  4 07:16 Desktop
 0: group:everyone deny delete
drwx------+  4 hacker  staff    128 Nov  2 16:11 Documents
 0: group:everyone deny delete
drwx------+  3 hacker  staff     96 Jul  4 07:16 Downloads
 0: group:everyone deny delete
drwx------@ 55 hacker  staff   1760 Nov  2 16:16 Library
```

Protección

Control de acceso: permisos en Windows





Archivos mapeados en memoria

- Sea una lectura secuencial de un archivo en disco utilizando las llamadas estándar del sistema `open()`, `read()` y `write()`.
  - Cada acceso al archivo requiere una llamada al sistema y un acceso al disco.
- Alternativamente, podemos usar las técnicas de memoria virtual para tratar la `E/S` de archivos como accesos a memoria.
- Este enfoque permite que una parte del espacio de direcciones virtuales se asocie lógicamente con el archivo.
- Esto puede aumentar significativamente en el rendimiento.

Archivos mapeados en memoria

Mecanismo básico

- La asignación de memoria de un archivo se realiza asignando un bloque de disco a una página (o páginas) en la memoria.
- El acceso inicial al archivo se realiza mediante paginación de demanda ordinaria, lo que resulta en un *page fault*.
- Sin embargo, una parte del archivo del tamaño de una página se lee desde el sistema de archivos en una página física.
- Las lecturas y escrituras posteriores en el archivo se manejan con rutinas de accesos a memoria.
- Manipular archivos a través de memoria, en lugar de incurrir en la sobrecarga de usar las llamadas al sistema `read()` y `write()`, simplifica y acelera el acceso y uso de los archivos.
- Se debe considerar que las escrituras en el archivo asignado en la memoria no son necesariamente escrituras inmediatas (sincrónicas) en el archivo en el almacenamiento secundario.
- En general, el archivo se actualiza cuando se cierra.

Archivos mapeados en memoria

Mecanismo básico

- Se puede permitir a múltiples procesos mapear el mismo archivo permitiendo intercambiar datos.
- Las escrituras de cualquiera de los procesos modifican los datos en la memoria virtual y serán vistos por todos los que mapean la misma sección del archivo.
- Las llamadas al sistema de mapeo de memoria también pueden admitir la funcionalidad de ***`copy-on-write`***, lo que permite que los procesos compartan un archivo en modo de lectura pero tengan las copias de cualquier dato que modifiquen.