SISTEMA OPERATIVO

ACTIVIDAD DE APRENDIZAJE 13

JESSICA ALEXANDRA MAGAÑA SALCEDO

215616229

Ingeniería en computación

28/04/2024

Departamento de ciencias computacionales

Centro Universitario de Ciencias Exactas e Ingenierías

Índice

Introducción

La administración eficiente de archivos y directorios es esencial para el correcto funcionamiento de cualquier sistema operativo. En esta investigación, exploraremos los aspectos clave relacionados con el manejo de archivos y directorios en sistemas operativos modernos. Se examinarán tanto los principios fundamentales como las tecnologías prácticas utilizadas en la gestión del almacenamiento de información.

El sistema de archivos es un componente fundamental de cualquier sistema operativo, y su comprensión es crucial para entender cómo se organiza y almacena la información en medios físicos, como discos duros y unidades de estado sólido. Desde una perspectiva física, se analizarán los mecanismos y estructuras utilizados por el sistema operativo para almacenar datos, incluyendo la estructura de los bloques de datos, los metadatos asociados y los procedimientos de acceso a estos datos físicos.

Por otro lado, la perspectiva lógica del sistema de archivos proporciona una abstracción para el usuario y facilita la manipulación de la información almacenada. Se explorarán cómo se organizan los archivos en directorios y cómo se establecen relaciones jerárquicas entre ellos para formar una estructura de árbol invertido. Además, se analizarán las acciones posibles en el manejo de archivos y directorios, incluyendo operaciones básicas como la creación, eliminación, lectura y escritura de archivos, así como acciones relacionadas con la gestión de directorios.

Además, se investigarán los diferentes métodos utilizados en sistemas operativos para indexar y acceder eficientemente a archivos almacenados en dispositivos de almacenamiento. Se explorarán los índices y su papel en la optimización de la búsqueda y el acceso a archivos, así como los métodos de dispersión (hashing) y las técnicas de resolución de colisiones utilizadas para garantizar una gestión eficiente de la información en estructuras de datos como las tablas hash.

Manejo de archivos

Una de las principales funciones de un Sistema Operativo es la administración del almacenamiento de información, para lo cual es necesario contar con un "Sistema de Archivos". Con este término se hace referencia, por un lado, a los mecanismos y estructuras que el sistema operativo utiliza para organizar la información en medios físicos tales como discos y diskettes (aspecto *físico* del sistema de archivos), y por otro a la visión que es ofrecida al usuario para permitir la manipulación de la información almacenada (una abstracción, o perspectiva *lógica* del sistema de archivos).

Un archivo es una colección de datos que se almacena en un medio físico y a la cual se le asigna un nombre. Los archivos, a su vez, están agrupados en conjuntos llamados directorios. Un directorio puede tener subdirectorios, formándose así una estructura jerárquica con la forma de un árbol invertido. El directorio inicial de esa jerarquía se denomina directorio *raíz* y se simboliza con una barra de división (/).

Acciones con archivos y directorios

El sistema de archivos de un sistema Linux típico está formado por los siguientes directorios bajo el directorio raíz:

- **/bin** Contiene los programas ejecutables que son parte del sistema operativo Linux. Muchos comandos de Linux como cat, cp, ls, more y tar están ubicados en este directorio.
- **/boot** Contienen el kernel (o núcleo) de Linux y otros archivos necesarios para el administrador de inicio LILO, que realiza la carga inicial del sistema operativo cuando la computadora se enciende.
- **/dev** Contienen todos los archivos de acceso a dispositivos. Linux trata cada dispositivo (terminales, discos, impresoras, etc.) como si fuera un archivo especial.
- **/etc.** Contiene archivos de configuración del sistema y los programas de inicialización.
- **/home** Contiene los directorios HOME de los usuarios. El directorio HOME el directorio inicial en el que se encuentra posicionado un usuario al ingresar al sistema, por lo que también se conoce como *directorio de logín* o *de conexión*.
- **/lib** Contiene los archivos de biblioteca utilizados por las aplicaciones y utilidades del sistema, así también como las librerías pertenecientes a diferentes lenguajes de programación.
- **/lost+found** Directorio para archivos recuperados por el proceso de reparación del sistema de archivos, que se ejecuta luego de una caída

del sistema y asegura su integridad luego de que el equipo haya sido apagado de manera inapropiada.

- **/mnt** Es un directorio vacío que se usa normalmente para montar dispositivos como disquetes y particiones temporales de disco.
- **/proc** Contiene archivos con información sobre el estado de ejecución del sistema operativo y de los procesos.
- **/root** Es el directorio HOME para el usuario root (administrador del sistema).
- **/sbin** Contienen archivos ejecutables que son comandos que se usan normalmente para la administración del sistema.
- **/tmp** Directorio temporal que puede usar cualquier usuario como directorio transitorio.
- **/usr** Contiene archivos de programa, de datos y de librerías asociados con las actividades de los usuarios.
- **/var** Contiene archivos temporales y de trabajo generados por programas del sistema. A diferencia de /tmp, los usuarios comunes no tienen permiso para utilizar los subdirectorios que contiene directamente, sino que deben hacerlo a través de aplicaciones y utilidades del sistema.

Índices

En sistemas operativos, los índices son estructuras de datos fundamentales utilizadas para optimizar la búsqueda y el acceso a archivos almacenados en dispositivos de almacenamiento, como discos duros o unidades de estado sólido (SSD). Estos índices actúan como mapas que proporcionan información sobre la ubicación física de los archivos en el medio de almacenamiento, lo que facilita su recuperación rápida y eficiente.

Los sistemas operativos utilizan diferentes tipos de índices para organizar la información en el sistema de archivos. Uno de los índices más comunes es el "índice de archivos maestro" o "tabla de asignación de archivos" (FAT, por sus siglas en inglés), utilizado en sistemas de archivos como FAT16, FAT32 y exFAT. En este tipo de índice, cada entrada de la tabla contiene información sobre un archivo, incluyendo su nombre, tamaño y la dirección del primer clúster donde se encuentra almacenado en el disco.

Otro tipo de índice común es el "mapa de bits", que se utiliza en sistemas de archivos como NTFS (Sistema de Archivos Nuevo Tecnológico) en sistemas operativos Windows. En un mapa de bits, cada bloque de datos en el disco tiene un bit asociado que indica si ese bloque está ocupado (1) o libre (0). Este tipo de índice facilita la asignación eficiente de bloques de datos para nuevos archivos y la recuperación rápida de bloques libres durante la escritura de archivos.

Los índices en sistemas operativos son esenciales para mejorar el rendimiento del sistema de archivos al acelerar las operaciones de lectura y escritura de archivos. Al mantener una estructura organizada de información sobre la ubicación de los archivos en el disco, los índices permiten al sistema operativo localizar rápidamente los archivos solicitados sin necesidad de buscar manualmente a través de todo el medio de almacenamiento.

Dispersión

La dispersión, también conocida como hashing, es una técnica fundamental en ciencias de la computación utilizada para asignar datos de tamaño variable a ubicaciones de memoria de tamaño fijo. Esta técnica se utiliza ampliamente en sistemas operativos para organizar y acceder a datos de manera eficiente en estructuras de datos como tablas hash.

En el proceso de dispersión, se aplica una función hash a los datos de entrada para generar un valor único y determinístico, conocido como hash. Este hash actúa como un identificador o clave que se utiliza para indexar o mapear los datos a una ubicación específica en una estructura de datos, como una tabla hash. La función hash debe ser lo más uniforme posible para minimizar las colisiones, es decir, situaciones en las que dos datos distintos producen el mismo valor hash.

Una vez que se calcula el hash, se utiliza como índice para acceder a una ubicación de memoria específica donde se almacenan los datos correspondientes. Esta ubicación de memoria se conoce como "bucket" en el contexto de una tabla hash. Si hay colisiones, es decir, dos o más datos tienen el mismo valor hash y se asignan al mismo bucket, se deben utilizar técnicas de resolución de colisiones para manejar este escenario.

Algunos métodos comunes de dispersión incluyen la división, la multiplicación y la suma. En la técnica de división, se calcula el hash dividiendo el valor de la clave por el tamaño de la tabla y utilizando el residuo como índice. En la técnica de multiplicación, se multiplica el valor de la clave por una constante y se utiliza la parte fraccionaria del resultado como índice. En la técnica de suma, se suman los valores de los caracteres de la clave y se utiliza el resultado como índice.

La dispersión es fundamental para la implementación eficiente de estructuras de datos como tablas hash, que se utilizan en sistemas operativos para diversas aplicaciones, como la gestión de archivos, la gestión de memoria y la administración de procesos. Proporciona un método rápido y eficiente para buscar y acceder a datos en grandes conjuntos de información, lo que contribuye significativamente al rendimiento y la escalabilidad de los sistemas operativos.

Métodos de dispersión

Los métodos de dispersión, también conocidos como funciones hash, son algoritmos fundamentales en ciencias de la computación y sistemas operativos utilizados para calcular valores hash a partir de datos de entrada. Estos valores hash se utilizan para indexar o mapear los datos a ubicaciones específicas en estructuras de datos como tablas hash, lo que permite un acceso rápido y eficiente a los datos.

Uno de los métodos de dispersión más simples y comunes es el método de división. En este método, se divide el valor de la clave por el tamaño de la tabla y se utiliza el residuo como índice para la ubicación en la tabla hash. Por ejemplo, si el tamaño de la tabla es 10 y la clave es 27, el valor hash sería 27 % 10 = 7. Este método es fácil de implementar y adecuado para casos simples donde el tamaño de la tabla es un número primo.

Otro método ampliamente utilizado es el método de multiplicación. En este enfoque, se multiplica el valor de la clave por una constante, se toma la parte fraccionaria del resultado y se multiplica por el tamaño de la tabla. Luego, se trunca el resultado para obtener el índice en la tabla hash. Este método es útil para evitar patrones repetitivos en los valores hash y puede proporcionar una mejor distribución de las claves en la tabla.

El método de suma es otro enfoque básico donde se suman los valores de los caracteres individuales de la clave y se utiliza el resultado como valor hash. Por ejemplo, si la clave es una cadena de caracteres, se suman los valores ASCII de los caracteres y se utiliza el resultado como índice. Este método es simple pero puede no ser tan efectivo en la distribución uniforme de claves en la tabla hash.

El método de plegamiento es un enfoque que consiste en dividir la clave en partes iguales (generalmente del mismo tamaño), sumar estas partes y luego tomar el residuo de esta suma con respecto al tamaño de la tabla. Este método es útil para claves numéricas largas y puede proporcionar una mejor distribución de claves en la tabla hash.

Otro método interesante es el método del cuadrado medio, donde se eleva al cuadrado el valor de la clave y se toman los dígitos intermedios como el valor hash. Este método es útil para claves numéricas largas y puede ayudar a evitar patrones repetitivos en los valores hash.

Además de estos métodos básicos, también existen enfoques más avanzados y especializados, como los métodos pseudo-aleatorios, que utilizan funciones pseudo-aleatorias para generar valores hash a partir de las claves. Estos métodos generan valores hash distribuidos de manera uniforme en el rango de índices de la tabla hash, lo que puede mejorar el rendimiento y la eficiencia de la tabla hash en ciertos casos de uso.

La elección del método de dispersión adecuado depende de varios factores, como la naturaleza de los datos de entrada, la distribución de las claves y los requisitos de rendimiento del sistema. Es importante seleccionar un método que proporcione una distribución uniforme de claves en la tabla hash para minimizar las colisiones y optimizar el acceso a los datos.

Solución de colisiones

La solución de colisiones es un aspecto crucial en el diseño y la implementación de estructuras de datos basadas en hash, como las tablas hash, utilizadas en sistemas operativos y otros contextos informáticos. Las colisiones ocurren cuando dos o más claves diferentes se asignan a la misma ubicación en la estructura de datos hash, lo que puede provocar pérdida de datos o degradación del rendimiento si no se manejan adecuadamente.

Existen varias técnicas comunes para resolver colisiones en estructuras de datos basadas en hash. Algunas de las más utilizadas incluyen:

1. **Encadenamiento:** En esta técnica, cada celda de la tabla hash contiene una lista enlazada de elementos que tienen el mismo valor hash. Cuando se produce una colisión, el nuevo elemento se agrega a la lista enlazada correspondiente a la celda de la tabla hash. Si bien el encadenamiento puede ser fácil de implementar y puede manejar un número ilimitado de colisiones, puede resultar en un mayor uso de memoria y en un rendimiento más lento debido a la necesidad de recorrer la lista enlazada para buscar un elemento.
2. **Direccionamiento abierto:** En este enfoque, cuando se produce una colisión, se busca una ubicación alternativa en la tabla hash para colocar el elemento en conflicto. Esto se hace utilizando una técnica de exploración que determina cómo se selecciona la nueva ubicación. Algunas técnicas de direccionamiento abierto incluyen la exploración lineal, la exploración cuadrática y la exploración de doble hash. El direccionamiento abierto puede ser más eficiente en términos de espacio y rendimiento que el encadenamiento, pero puede ser más complicado de implementar correctamente.
3. **Rehashing:** En este método, cuando se produce una colisión, se calcula un nuevo valor hash para la clave en conflicto utilizando una segunda función hash. Este nuevo valor hash se utiliza para encontrar una nueva ubicación en la tabla hash donde colocar el elemento. Si bien el rehashing puede ser efectivo para resolver algunas colisiones, puede ser costoso en términos de tiempo de procesamiento si se producen colisiones frecuentes.
4. **Doble hashing:** Esta técnica combina aspectos del direccionamiento abierto y el rehashing. Cuando se produce una colisión, se utiliza una segunda función hash para calcular una distancia adicional a la siguiente ubicación en

la tabla hash. Esta distancia se suma al índice original para encontrar la siguiente ubicación. El doble hashing puede proporcionar una distribución más uniforme de elementos en la tabla hash y puede ser más eficiente que otras técnicas de resolución de colisiones.

Cada una de estas técnicas tiene sus propias ventajas y desventajas, y la elección de la técnica adecuada depende de varios factores, como el tamaño esperado de la tabla hash, el número de elementos que se espera almacenar en la tabla y los requisitos de rendimiento del sistema.

Conclusión

En conclusión, la creación y terminación de procesos son elementos esenciales en la operación de cualquier sistema operativo. La capacidad de crear y gestionar procesos de manera eficiente permite que el sistema maneje tareas de forma ordenada y aproveche los recursos disponibles de manera óptima. La terminación de procesos, ya sea por finalización exitosa o por eventos inesperados, es crucial para liberar recursos y mantener la estabilidad del sistema.

Los algoritmos de planificación desempeñan un papel vital en la asignación de recursos de CPU, determinando qué proceso se ejecutará a continuación. Estos algoritmos, basados en diversas políticas, buscan maximizar la utilización de recursos, minimizar los tiempos de espera y garantizar una respuesta rápida a las solicitudes del usuario.

El Bloque de Control de Procesos (BCP) proporciona una estructura de datos fundamental para el seguimiento y la gestión de información detallada sobre cada proceso en ejecución. El BCP permite al sistema operativo llevar a cabo operaciones como la conmutación de procesos, la planificación de CPU y la gestión de recursos de manera efectiva.

En conjunto, la creación y terminación de procesos, junto con los algoritmos de planificación y el uso del BCP, son aspectos críticos para el funcionamiento eficiente de un sistema operativo. Una gestión adecuada de estos elementos garantiza un rendimiento óptimo del sistema, una respuesta rápida a las demandas del usuario y una mayor estabilidad operativa.

Bibliografía

Marco. (s. f.). *MANEJO DE ARCHIVOS*.

https://www.investigacion.frc.utn.edu.ar/labsis/publicaciones/apunte_linux/ma.html

Resumen de sus intentos previos

Intento	Estado	Calificación / 3.00	Calificación / 4.00	Revisión
1	Terminados Enviado domingo, 28 de abril de 2024, 21:16	1.00	1.33	No permitido
2	Terminados Enviado domingo, 28 de abril de 2024, 21:18	1.00	1.33	No permitido
3	Terminados Enviado domingo, 28 de abril de 2024, 21:20	0.00	0.00	No permitido
4	Terminados Enviado domingo, 28 de abril de 2024, 21:23	2.00	2.67	No permitido
5	Terminados Enviado domingo, 28 de abril de 2024, 23:02	1.00	1.33	
6	Terminados Enviado domingo, 28 de abril de 2024, 23:03	3.00	4.00	

Su calificación final para este examen es 4.00/4.00.