Ana Daysi Montecino*

Proyecto de integración de sistemas de archivos Linux – MySQL

Resumen

Este proyecto de desarrollo trata sobre la integración de dos sistemas totalmente diferentes en sus funciones, como son: un sistema operativo (Linux) y un gestor de base de datos (MySQL); y busca como objetivo permitir a MySQL aumentar el desempeño en sus procesos de lectura/escritura, disminuyendo tiempo de acceso a disco, controlando su propio sistema de archivos sin utilizar los del Sistema Operativo.

En el presente artículo se examina las relaciones entre los diferentes sistemas y la manera que Linux maneja los archivos; además se proporciona un ejemplo del código de Linux y se plantean las diferentes alternativas para poder desarrollar los cambios necesarios. El resultado final que se pretende es, introducir al usuario en el concepto de integración que se busca realizar y dar a conocer cuál de las propuestas presentadas es la mejor opción a desarrollar. Sin embargo, el proyecto no concluye con el artículo; es un trabajo amplio de investigación que pretende desarrollar la codificación de un sistema de archivos integrado con una base de datos.

Palabras clave: Linux, kernel, núcleo, Mysql, sistema de archivos, ext2, vfs, nodo.

Introducción

Los sistemas de computación integran diversos elementos con el fin de solucionar un problema o superar una deficiencia. La mayoría de sistemas de cómputo tienen tres elementos claves: plataforma, almacenamiento e interfases. Estos tres elementos integrados permiten que un sistema sea eficiente o no.

Cuando se busca una solución para un problema, y se desea o necesita que la solución sea informática, se trata de buscar la mejor relación entre los diversos elementos, y esto garantiza que si la solución es efectiva también será lo más eficiente que se pueda. Es por ello que empresas como Oracle Corporation buscan integrar nueva tecnología, utilizando Linux como plataforma para su base de datos.

Esa integración le ha permitido a Oracle, utilizando un cluster de servidores, superar la barrera del millón de transacciones, esto utilizando Red Hat Entrerprise en servidores de bajo costo[1].

Resumir todo el trabajo que conlleva un proyecto de integración como este, en un solo artículo, es una tarea sino imposible, lo más cercano a serlo; sin embargo se seleccionarán algunos puntos que permitan mostrar y ejemplificar el proceso de manera general. En este documento se presentarán los primeros pasos para integrar software libre de sistemas operativos y de bases de datos; analizando el núcleo y el sistema de archivos, se expondrá además, un ejemplo para entender cómo se utilizan las estructuras en la programación de núcleo.

Por el momento este es un esfuerzo individual, no un proyecto de una comunidad de programadores, como tradicionalmente se trabaja en Linux y MySQL.

MySQL como aplicación de base de datos

El desempeño de las bases de datos se encuentra completamente ligado al desempeño del sistema operativo y del hardware; esto es una realidad, sobre todo cuando se habla de bases de datos transaccionales y más aún si estas están ligadas a procesos críticos, dentro del negocio o producción; por ello lograr sistemas con tiempos de respuesta cada vez más cortos es uno de los puntos fuertes en el desarrollo de gestores de bases de datos.

* Docente tiempo completo de la Universidad Don Bosco. Licenciada en Ciencias de la Computación por la Universidad Centroamericana José Simeón Cañas (UCA).
E-mail: ana.montecino@udb.edu.sv

1. http://www.diarioti.com/gate/n.php?id=4455

Abstract

This project is about the integration of two systems that have totally different functions: an operating system (Linux) and a database manager (MySQL). The main objective of the project is to allow MySQL to increase its writing/reading performance processes, diminishing disk access time, controlling its own files system without using those of an operating system.

This paper examines the relationship between different systems and the way Linux manages the files. It also provides an example of a Linux code and presents the alternatives to carry out the necessary changes. The goal of the paper is to introduce the users to the integration concept sought and to show which of the presented proposals is the best choice. This article is just part of the results obtained in the project which is a larger research process that seeks to develop the codification of a files system integrated with a database.

Keywords: Linux, kernel, Mysql, file system, ext2, vfs, node.

No obstante existen muchas opciones de bases de datos en el mercado, quizás la de mayor uso en Internet sea MySQL. Aunque es muy difícil determinarlo, esto puede estimarse debido principalmente al auge que ha tenido el lenguaje PHP en el desarrollo de aplicaciones en Internet y otros lenguajes de programación que pueden utilizar fácilmente MySQL como gestor . Otra de las razones, probablemente sea su gratuidad, esto ha permitido a desarrolladores independientes contar con una base de datos rápida (principal fortaleza de MySQL) sin tener que pagar algún tipo de licenciamiento y ser utilizada en diversas plataformas, incluidos Windows y Linux; este último comparte características con MySQL como su licenciamiento GNU[2] y su código abierto.

MySQL, como se dijo anteriormente, es una base de datos rápida, sin embargo todavía tiene que pasar sobre la capa del sistema operativo, ver figura1:

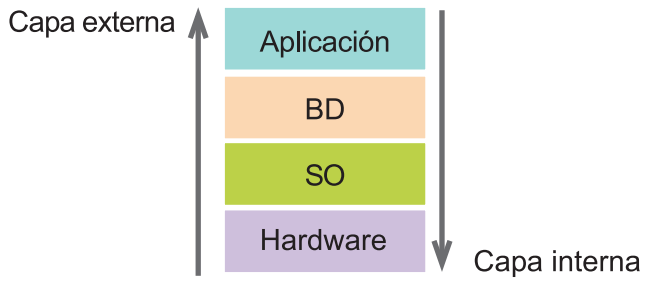

Figura1 Esquema de capas de elementos de un sistema Informático.

Como se puede observar en el esquema de capas (figura1), existen 4 capas las cuales funcionan de la siguiente manera: la capa más externa corresponde a la aplicación de usuarios y es la que directamente interactúa con este, la siguiente es la del gestor de base de datos y es la que le permite recuperar información y mostrarla coherentemente, con el fin ser utilizada por la aplicación, para realizar cálculos, reportes, etc. La siguiente capa es la del sistema operativo, quien decide qué proceso es el más importante y cuándo ejecutarlo, controla además el sistema de archivos. Por último tenemos la capa del hardware, que es la parte física del sistema y es la que permite almacenar la información. De las cuatro capas presentadas, la más importante para el desempeño de un sistema, en lo que se refiere a manejo de procesos es la del sistema operativo; ya que esta maneja entre otras cosas la comunicación entre el hardware y el resto del software, controlando la manera en que se almacenan los datos (sistema de archivos), los cálculos (procesos lógicos-matemáticos) y cualquier tipo de excepciones que se presente (ver figura 2).

2. Tipo de licenciamiento que permite la distribución, uso y modificación de software y lo protege de cualquier intento de apropiación.

Figura 2 Elementos de la capa del sistema operativo

El sistema operativo Linux

En la década de los noventas Linus Torvalds desarrolló un sistema operativo de 32 bits basado en UNIX[3], pero que podía ejecutarse en equipos con procesadores X86. En la actualidad existen muchas distribuciones (o implementaciones) de Linux, pero todas tienen su base en el núcleo original desarrollado por Torvalds. Talvez una de las distribuciones más conocidas es Red Hat, ya que fue diseñada y desarrollada para PC's de uso doméstico.

Red Hat fue desarrollado por Bob Young, quien fundó una empresa con el mismo nombre, que se encargaba de distribuir en CD's el sistema operativo, al mismo tiempo de mejorarlo con ayuda de comentarios hechos por usuarios alrededor del mundo, quienes reportaban los fallos y hacían contribuciones a los paquetes. Luego en mayo de 2003, se deriva el Proyecto Fedora, que es un proyecto que involucra a desarrolladores de software libre que quieran integrase; es en su mayoría patrocinado por Red Hat, quien se beneficia incluyendo parte del código en su versión de Red Hat Enterprise. Este proyecto ha dado como resultado nuevas distribuciones de Fedora Core.

El Kernel (Núcleo) de Linux

Linux inició como un sistema con núcleo monolítico, es decir un gran número de líneas de código en un solo programa que se cargaba al iniciar el sistema operativo (Linus Torvalds desarrollo el núcleo y fue lo que originalmente se llamo Linux). A medida que más desarrolladores fueron implementando sus aportaciones, se fue haciendo difícil mantener dicha estructura. Actualmente las implementaciones recientes son modulares, lo que facilitó la administración y el desarrollo de nuevas aplicaciones, ya que si una parte del sistema operativo no trabaja correctamente o se desea actualizarla, basta con solo descargar la nueva versión compilarla y reiniciar el servicio. Otra de las ventajas de usar un kernel modular es la de poder instalar controladores de dispositivos sin necesidad de incrementar el código del propio núcleo, y a la vez, solamente se levanta el servicio durante el tiempo que sea necesario, optimizando el recurso de memoria del sistema.

Los módulos del núcleo de Linux son archivos objetos (de extensión .o) producidos por el compilador de C, y son almacenados en la carpeta *lib/modules/versión-núcleo*, donde versión-núcleo es la cadena que identifica

3. Sistema operativo desarrollado en los laboratorios Bell de AT&T, a finales de 1960.

la versión del núcleo, por ejemplo la versión 2.6.18 estaría instalada en la carpeta *lib/modules/2.6.1*. En una sola instalación de Linux puede existir diferentes versiones compiladas del núcleo, pero esta estructura previene que se pueda utilizar un módulo de una versión en otra por error, ya que cada vez que se instala una nueva versión del núcleo, se crea una estructura de directorios específica para ella.

Los módulos son agrupados en subcarpetas, y cada una de ellas representa una categoría; entre algunas de las categorías en las que podemos agrupar los módulos del núcleo están[4]:

block	Módulos para unos pocos bloques específicos de dispositivos como son controladoras RAID o dispositivos de cinta IDE.
Cdrom	Módulos controladores para unidades de CD no estándares.
fs	Módulos para controlar sistemas de archivos como ext2, ext3, ms-dos, etc.
ipv4	Características del núcleo que pueden usarse en redes TCP/IP, como módulo de firewalls.
msc	Cualquier módulo que no pueda clasificarse en las categorías existentes puede colocarse aquí.
net	Módulos para controlar las interfases de red.
SCSI	Módulos controladores para tarjetas SCSI.
video	Módulos para los dispositivos de video.

Entendiendo el Sistema de Archivos (File System) de Linux

En 1993, la comunidad de Linux inició el esfuerzo de estandarizar el sistema de archivos, en todas las distribuciones para propósitos generales de Linux, tratando de reducir la proliferación de los diversos sistemas de archivos que existían hasta el momento. Este intento dió como resultado en 1994, un documento describiendo los estándares del Sistema de Archivos (*Filesystem Standard*), a dicho documento se le conoció usualmente como *FSSTND*. Al año siguiente el grupo comenzó a reducir las especificaciones y a refinar los estándares para incluir otros sistemas operativos basados en Unix.

Como el FSSTND atrajo mucha simpatía, fue renombrado *Filesystem Hierarchy Standard* (*Estándares de Jerarquía del Sistema de Archivos o FHS*), y si bien FHS no es un requerimiento para los desarrolladores y los distribuidores de Linux, la comunidad en general entiende la importancia de los estándares.

4. LPI Linux Certification in a Nutshell, 2nd Edition; Jeff Dean, Bruno Gomes Pessanha, Nicolai Langfeldt, Steven Pritchard, James Stanger

De este documento se desprenden recomendaciones sobre cómo son los tipos de datos (Data Type) de los cuales se definen dos categorías, cada cual con sus respectivos subtipos opuestos.

Tipos y subtipos[5]

Data Sharing:

Esta categoría define los tipos de datos que se pueden compartir en red, los subtipos son:

> ***Sharable***: Datos que pueden ser compartidos en diferentes sistemas host sobre una red. Los archivos compartibles contienen información de propósito general, sin ataduras de algún tipo a un host específico, como por ejemplo archivos de usuarios, programas ejecutables de aplicación o manuales del sistema.
>
> ***Non-sharable:*** Datos que no pueden ser compartidos, ya que están ligados a un host específico.

Data modification:

Esta categoría indica la manera en que los datos cambian, los subtipos son:

> ***Variable:*** Los datos son variables cuando cambian de manera natural, frecuentemente por procesos, ejemplo: los archivos de registro del sistema (archivos log), los archivos de usuarios, etc.
>
> ***Static:*** los datos estáticos representan la mayoría de información en un sistema, generalmente permanecen inmutables durante días, o aún durante años. Un ejemplo de éstos son los módulos, o programas binarios como el ls y el bash, que no cambian sino hasta que el administrador del sistema los actualiza.

Algunas carpetas del sistema de archivos de Linux pretenden mantener tipos específicos de datos, por ejemplo, los archivos ejecutables dentro de la carpeta /usr raramente cambian, así pueden ser llamados estáticos, al mismo tiempo estos archivos son compartidos por todos los usuarios que se conectan al sistema, lo que los coloca en la categoría *sharable*; manteniendo este tipo de organización respecto a sus atributos se pueden simplificar tanto el compartir y administrar los archivos como la complejidad del backup.

El documento FHS ofrece además, una detallada documentación sobre dónde se encuentran los archivos, utilizando el razonamiento derivado de

5. LPI Linux Certification in a Nutshell, 2nd Edition; Jeff Dean, Bruno Gomes Pessanha, Nicolai Langfeldt, Steven Pritchard, James Stanger.

las definiciones Shrable/No-sharable y Variable/Static descritas anteriormente, y coloca al directorio root (directorio raíz del sistema) al inicio del directorio jerárquico; define además sus características que son las siguientes:

- Debe contener la mayoría de los archivos necesarios para iniciar el sistema, incluyendo la habilidad de montar otros sistemas de archivos.

- Obliga a contener las utilidades para reparar o restaurar el sistema, obviamente solo podrá hacerlo el administrador.

- Necesita ser relativamente pequeño, ya una partición pequeña es menos corruptible cuando el sistema colapse o existan fallas en el suministro de poder. Además deberá contener solamente datos Non-sharables, maximizando el resto del espacio para datos compartidos.

- Las diversas aplicaciones no podrán crear archivos o directorios del sistema de archivos del */root*.

Por ello el directorio /root incluye otros subdirectorios, entre los más básicos se encuentran:

/bin:
El directorio */bin* contiene los comandos del sistema como cp, date, ln, ls, medir y otros. Estos comandos son considerados esenciales para la administración del sistema en caso de problemas.

/dev:
Archivos de dispositivos, necesarios para acceder a discos y otros dispositivos son almacenados en este directorio. Como ejemplo aquí se incluyen las particiones de disco, las terminales como tty1, etc. Los dispositivos están presentes al momento de iniciar el sistema para su correcto montaje y configuración. La excepción a esto son sistemas utilizando */devf*, el cual es una reciente adición al kernel de Linux, que hace a */dev* un sistema de archivos virtual.

/etc:
El directorio */etc* contiene información de configuración única del sistema y es requerida para el arranque. No existen archivos binarios almacenados. Las anteriores costumbres en versiones anteriores de Unix tenían ejecutables administrativos almacenados en */etc*. Estos han sido movidos a */sbin* bajo el FHS, como por ejemplo passwd, hosts y login.defs.

/lib:
El directorio */lib* contiene librerías compartidas y los módulos del kernel, ambos esenciales para el inicio del sistema.

/mnt:
Este directorio ha sido puesto para uso del administrador local del sistema, pero también puede llegar a ser configurado para uso de otros usuarios. Generalmente, se encuentra vacío, salvo algunos puntos de montaje de particiones temporales, incluyendo el cdrom y floppies.

/root:
Es el directorio por defecto de usuario *root*. Habitualmente ningún otro usuario puede ver la información de este directorio por seguridad. No es un directorio esencial para el sistema de archivos, sin embargo puede utilizarse para mantener archivos de configuración disponibles del usuario root, como mantenimiento del sistema o recuperación del mismo.
/sbin: Utilidades esenciales utilizadas por el administrador son almacenadas en este directorio. Ejemplo: fdisk, fsck y mkfs.

Ext2 (Extended File System 2)
Es el más popular de los sistemas de archivos, su origen se remonta a los primeros días de Linux, ya que su antecesor (Extended File System) fue implementado en la versión 0.96c en la primavera de 1992, luego sufrió una serie de cambios que lo llevaron hasta la versión actual.

Estructuras del sistema de archivos

Como se ha mencionado, Linux basa su sistema de archivos en un árbol jerárquico de objetos conteniendo otros objetos. Este modelo permite una buena organización y que dos o varios objetos puedan tener el mismo nombre aunque en diferente localización. No obstante existen diferentes tipos de objetos en Linux, los más comunes son archivos y directorios.

Directorios
Los directorios son objetos contenedores que mantienen otros objetos como archivos y directorios.

Archivos
Un archivo es un objeto que existe dentro de un directorio y su propósito es ser el repositorio de la información.

Es de aclarar que en Linux, pueden existir dos o más sistemas de archivos trabajando al mismo tiempo en diferentes particiones, sin que esto dificulte el poder compartir información entre ellas. Esto último es posible gracias a que Linux maneja una estructura común de información de archivos, que se "replica" entre los diferentes FS (File System); esas estructuras forman parte del Sistema de Archivos Virtual (Virtual File System VFS)[6].

En la figura 3 se muestra el esquema de interacción entre los diferentes elementos en el proceso de lectura/escritura. Cuando un proceso de usuario (una aplicación) debe interactuar con el Kernel, el proceso de usuario utiliza

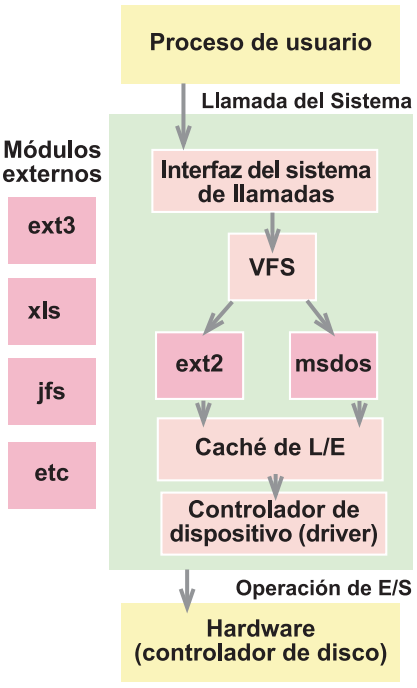

Figura 3. Esquema de Interacción[7]

6. Se utilizará indiscriminadamente cualquiera de los tres términos de sistemas de archivos virtual aquí mencionados.
7. http://www.zator.com/Hardware/H8_1_2a2.htm

la interfaz del sistema de llamadas, la cual nunca cambia; esta interfaz traslada las llamadas al módulo del sistema de archivos correspondiente a través del VFS. En el esquema los módulos utilizados como ejemplo son ext2 y msdos, aunque esto es valido para otros sistemas de archivos que hayan sido incluido en la compilación del núcleo y que han sido representados como "módulos externos", y hasta ser cargados, estos módulos permanecerán solamente en el disco como ficheros especiales.

El VFS es un sistema de archivos independiente del resto, que permite al kernel utilizar los archivos de cualquier FS, traduciendo la información en un formato entendible para el núcleo. Toda la información del FS se registra en un *Grupo de Bloque*, el cual contiene todas las estructuras de información de cada sistema de archivos.

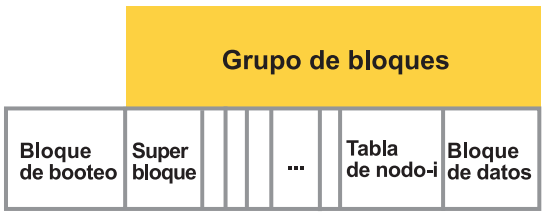

Figura 4. Representación de la configuración de un Grupo de Bloques

Como se puede observar en la gráfica, el grupo de bloques es un conjunto de diferentes metadatos (objetos con información específica de archivos y directorios), y contienen los datos necesarios que el kernel utiliza para manejar las particiones.

Todas estas estructuras o metadatos que utiliza Linux están definidos en cada partición, esto quiere decir que tendremos tantos *Grupos de Bloques* como particiones en nuestro sistema. A continuación se explicarán los metadatos más importantes.

Bloque de boot:
Es el primer bloque que encontramos en el sistema de archivos, por lo regular en el primer sector y contiene todos los archivos necesarios para el arranque del sistema.

Superbloque:
Contiene las características básicas del sistema de archivos y su tamaño, permitiéndole al administrador del FS usar y mantener el sistema de archivos. Normalmente cuando se monta el sistema de archivos sólo se lee el Superbloque del Grupo de Bloque 0, sin embargo existe una copia en *Grupo de Bloque* como respaldo en caso que se corrompa el FS. Algunos de los componentes del Superbloque son [8]:

8.http://www.netpecos.org/docs/linux/kernel.html

Magic Number:
Esto permite al software de montaje comprobar qué es realmente el Superbloque para un sistema de archivos EXT2. Para la versión actual de EXT2 éste es 0xEF53. (es un identificador que le indica al sistema operativo que el superbloque es de un sistema de ficheros ext2. ejemplo: para las versiones del núcleo 2.4 el valor es 0xEF53).

Revision Level:
Los niveles de revisión mayor y menor permiten al código de montaje determinar si este sistema de archivos soporta o no características que sólo están disponibles para revisiones particulares del sistema de archivos. También hay campos de compatibilidad que ayudan al código de montaje determinar qué nuevas características se pueden usar con seguridad en ese sistema de archivos.

Mount Count and Maximum Mount Count:
Juntos permiten al sistema determinar si el sistema de archivos fue comprobado correctamente. El contador de montaje se incrementa cada vez que se monta el sistema de archivos y cuando es igual al contador máximo de montaje muestra el mensaje de aviso «maximal mount count reached, running e2fsck is recommended».

Block Group Number:
Es el número del Grupo de Bloque que tiene la copia de este Superbloque. Block Size: Es el tamaño de bloque para este sistema de archivos en bytes, por ejemplo 1024 bytes.

Blocks per Group:
Es el número de bloques en un grupo. Como el tamaño de bloque, éste se fija cuando se crea el sistema de archivos.

Free Blocks:
Es el número de bloques libres en el sistema de ficheros.

Free Inodes:
Es el número de Inodos libres en el sistema de archivos.

First Inode:
Este es el número de inodo del primer inodo en el sistema de archivos. El primer inodo en un sistema de archivos EXT2 raíz sería la entrada directorio para el directorio '/'.

La estructura del superbloque para el FS ext2 se encuentra en la librería *include/linux/ext2_fs.h*, en la cual se encuentran además las otras estructuras y constantes que utilizan el sistema de archivos ext2 (ver figura 5).

Inodo:
La información de cada directorio o archivo es almacenada en una metadato en forma de tabla la cual es parte del sistema de archivos, y es numerado singularmente en dicha tabla. Esta información se le conoce como inodo, y es la forma en que son identificados los objetos por cualquier FS.

```
/*
 * Structure of the super block
 */
struct ext2_super_block {
__le32  s_inodes_count;/* Inodes count*/
__le32  s_blocks_count;/* Blocks count*/
__le32  s_r_blocks_count;      /* Reserved blocks count */
__le32  s_free_blocks_count;   /* Free blocks count */
__le32  s_free_inodes_count;   /* Free inodes count */
__le32  s_first_data_block;    /* First Data Block */
__le32  s_log_block_size;      /* Block size */
__le32  s_log_frag_size;       /* Fragment size */
__le32  s_blocks_per_group;    /* # Blocks per group */
__le32  s_frags_per_group;     /* # Fragments per group */
__le32  s_inodes_per_group;    /* # Inodes per group */
__le32  s_mtime;               /* Mount time */
__le32  s_wtime;               /* Write time */
__le16  s_mnt_count;           /* Mount count */
__le16  s_max_mnt_count;       /* Maximal mount count */
__le16  s_magic;               /* Magic signature */
__le16  s_state;               /* File system state */
__le16  s_errors;      /* Behaviour when detecting errors */
__le16  s_minor_rev_level;     /* minor revision level */
```

Figura 5. Segmento de código de la librería include/linux/ext2_fs.h en la cual se define el superbloque.

Los inodos son tradicionales en sistemas operativos basados en UNIX. Cada sistema de archivos ext2 (o cualquier otro sistema de archivo) de Linux es creado con un número finito de inodos los cuales son calculados en base al tamaño del sistema de archivos y otras opciones del comando mke2fs[9].

El origen del termino "*inodo*" no está claro, Dennis Ritchie[10] lo explicó así: "Era simplemente el nombre que comenzamos a utilizar. "*Índice*" es lo mejor que se me ocurre, debido a la estructura algo inusual de un sistema de archivos que almacenaba la información del acceso a los archivos como una lista plana en disco, dejando al margen toda la información jerárquica de los directorios. Así el número "i'" es un índice sobre la lista, el nodo "i" es el elemento seleccionado de la lista. (En la primera edición del manual se empleó la notación "i-nodo"; el guión fue desapareciendo poco a poco)"[11].

Al igual que el superbloque, la estructura del inodo se encuentra definida en la librería *include/linux/ext2_fs.h*, mientras que la estructura de los inodos del VFS se encuentra en la librería *include/linux/fs.h*.

9. Comando utilizado para crear sistemas de archivos en Linux.
10. Físico estadounidense que participó en el desarrollo del sistema operativo Unix y es el creador del lenguaje de programación C.
11. http://es.wikipedia.org/wiki/Inodo

```
struct inode {
        struct hlist_node      i_hash;
        struct list_head       i_list;
        struct list_head       i_sb_list;
        struct list_head       i_dentry;
        unsigned long          i_ino;
        atomic_t               i_count;
        umode_t                i_mode;
        unsigned int           i_nlink;
        uid_t                  i_uid;
        gid_t                  i_gid;
        dev_t                  i_rdev;
        loff_t                 i_size;
        struct timespec        i_atime;
        struct timespec        i_mtime;
        struct timespec        i_ctime;
        unsigned int           i_blkbits;
        unsigned long          i_blksize;
        unsigned long          i_version;
        blkcnt_t               i_blocks;
        unsigned short          i_bytes;
        spinlock_t             i_lock; /*
i_blocks, i_bytes, maybe i_size */
        struct mutex           i_mutex;
        struct rw_semaphore    i_alloc_sem;
        struct inode_operations       *i_op;
```

Figura 6. Segmento de código de la librería include/linux/fs.h en la cual se define la estructura del inodo.

Las estructuras de los sistemas de archivos son frecuentemente utilizadas en las funciones para controlar los procesos de lectura/escritura. Por ejemplo, el archivo fs/ext2/inode.c contiene las funciones que gestionan los inodos en el disco, es decir que en él se encuentran las funciones que permiten a Linux leer y escribir en el medio físico. Por ejemplo a continuación se muestra la función EXT2_DISCARD_PREALLOC la cual es llamada al cerrar un archivo, liberando los bloques asignados por otra función.

```
void ext2_discard_prealloc (struct inode * inode)
{
#ifdef EXT2_PREALLOCATE
  struct ext2_inode_info *ei = EXT2_I(inode);
  write_lock(&ei->i_meta_lock);
  if (ei->i_prealloc_count) {
    unsigned short total = ei->i_prealloc_count;
    unsigned long block = ei->i_prealloc_block;
    ei->i_prealloc_count = 0;
    ei->i_prealloc_block = 0;
    write_unlock(&ei->i_meta_lock);
    ext2_free_blocks (inode, block, total);
    return;
   } else
    write_unlock(&ei->i_meta_lock);
  #endif
}
```

Figura 7. Función EXT2_DISCARD_PREALLOC que se encuentra en el archivo fs/ext2/inode.c.

Esta función se asegura si la opción de poder reasignar bloques cumple la condición #ifdef EXT2_PREALLOCATE.

Luego se crea un apuntador al inodo del bloque, luego se cuentan los bloques asignados, y se evita que el bloque sea sobrescrito por otro proceso con la función write_lock(&ei->i_meta_lock);.

La función verifica entonces que tenga bloques asignados y si se cumple la condición if (ei->i_prealloc_count) {, la función entonces cuenta los bloques asignados y el bloque más recientemente reservado con las sentencias:

```
unsigned short total = ei->i_prealloc_count;
unsigned long block = ei->i_prealloc_block;.
```

Hecho esto el sistema coloca estos contadores a cero y libera el inodo para que pueda ser sobrescrito:

```
ei->i_prealloc_count = 0;
ei->i_prealloc_block = 0;
write_unlock(&ei->i_meta_lock);
```

La siguiente función es la que libera los bloques asignados para que puedan ser utilizados por otros procesos, a esta función se le envían tres parámetros: el inodo del archivo, el último bloque asignado y el total de bloques.

ext2_free_blocks (inode, block, total);.

MySQL como Sistema de Archivos de Linux

Para que MySQL pueda controlar el sistema de archivos de Linux, debe de modificarse el código de ambos software. La ventaja que ofrece un sistema modular como Linux es que permite que dicho cambio se lleve a cabo, aunque no sin un gran esfuerzo.

Este cambio obligaría a pensar en una base de datos que controle sus propias peticiones de lectura/escritura a disco; dicho cambio podría sugerir un retardo en los tiempos de respuesta, y obligaría a una base de datos tradicionalmente rápida a ser más lenta; sin embargo, también es factible que el tiempo de respuesta disminuya si la base de datos está instalada en una partición independiente de cualquier otra aplicación. Como parte de este cambio se sugiere que se maneje como un nuevo sistema de archivos basado en el ext2, pero independiente de este.

Como se ha mencionado antes, ext2 es de los primeros sistemas de archivos y el más difundido, lo que facilita encontrar información sobre su codificación, información sobre su evolución y otros datos que podrían facilitar el desarrollo

del proyecto. Otras de las consideraciones es que ext3 al igual que xfs soporta *registro por diario*[12] , lo que no se considera necesario para este proyecto, ya que solamente supone fines académicos.

En la figura 8 se plantea una de las posibles soluciones para esta propuesta, en la que MySQL toma un segmento del sistema de archivos existente, haciendo parte de su propia codificación.

Como se puede observar el la figura 8, el segmento de código del núcleo que controla el sistema de archivos formaría parte de la capa de base de datos, lo que permitiría al gestor controlar todos sus procesos de lectura/escritura. Esta modificación obligaría a crear un nuevo módulo de sistemas de archivos que incluya las funciones de MySQL, como parte de su codificación.

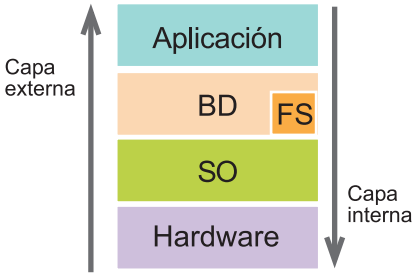

Figura 8. Esquema primera propuesta.

Otra solución permitiría a MySQL hacer uso directamente de las librerías del sistema de archivos ext2, esto evitaría la duplicidad de código y permitiría que las mejoras hechas al sistema de archivos, pasen a MySQL con cambios menores. La desventaja de esto es que conllevaría a una falla de seguridad del sistema, ya que le permitiría a una aplicación acceder directamente al núcleo del SO. En la figura 9 se representa el esquema planteado.

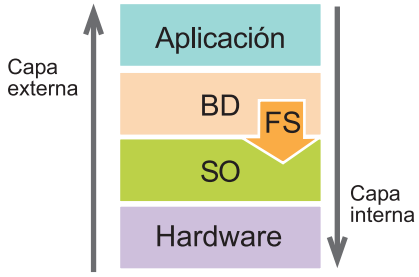

Figura 9. Esquema segunda propuesta.

La tercera opción consiste en modificar el kernel de Linux funcionándolo con MySQL, el cambio que se presenta en esta propuesta sería tan profundo que estaríamos hablando de una nueva distribución de Linux.

Este cambio obligaría además a que la única función del sistema en el que se monte sea solamente de servidor de bases de datos; además requeriría mucho más tiempo de desarrollo.

La solución óptima es una mezcla de las dos primeras propuestas mencionadas; consistiría en la creación de un nuevo módulo de sistema de archivos, restringiendo el acceso directo al núcleo de MySQL. Esto último solo requeriría una pequeña modificación a MySQL, en sentido del manejo de credenciales para poder acceder al núcleo. En la figura 11 se representa la propuesta.

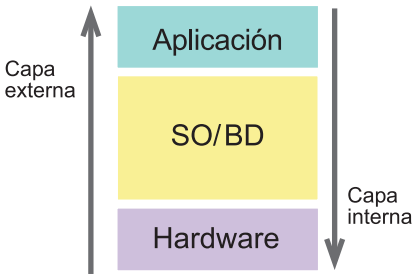

Figura 10. Esquema tercera propuesta.

Los principales puntos a tomar en cuenta para realizar esta última propuesta serían los siguientes:

1. Tomar como base ext2 como sistema de archivo.
2. Seleccionar la versión más reciente tanto del sistema operativo como de la base de datos.
3. Establecer el tipo de credenciales que se requerirá para que MySQL pueda acceder al núcleo.

12. El registro por diario es un mecanismo que permite a un sistema informático implementar transacciones, registrándolas en un *journal* o registro de diario con el fin de restablecer los datos si la transacción falla.

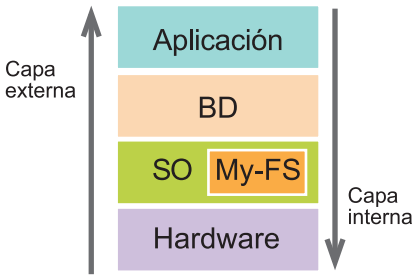

Figura 11. Esquema cuarta propuesta.

Una cosa importante será implementar los estándares POSIX[13] sobre el manejo de archivos en el módulo a desarrollar. Esto es importante ya que estos estándares establecen lo mínimo que un sistema debe tener para ser portable.

Otra de las consideraciones que se deben tomar en cuenta es el sistema de almacenamiento que se utilizará en MySQL. Aunque lo recomendable es que tanto InnoDB[14] como MyISAM[15] funcionen correctamente bajo el nuevo sistema de archivos, por cuestiones de tiempo de desarrollo puede optarse solamente por activar uno de los dos; esto con la idea de poder manejar los archivos de datos cuando se creen.

Conclusión

Linux es un sistema operativo versátil, que ha evolucionado con el paso de los años hasta tener una participación importante en el mercado de Sistemas Operativos. A juicio de la autora sería una muy buena oportunidad didáctica en las materias relacionadas a la carrera, ya que este proyecto abarca la mayoría de las competencias del profesional de informática, puesto que incluye programación, teoría de sistemas operativos y bases de datos, y en cierta medida, hardware.

Además de permitir el escenario idóneo para "Aprender haciendo", permite a la universidad Don Bosco integrarse con un papel investigador a los esfuerzos de una gran comunidad de profesionales y de sus centros de estudios, como una precursora visionaria que busca innovar en un área tan fértil como lo es la computación. Este primer paso al que le seguirán muchos, establece el inicio de un proyecto amplio y de mucho trabajo, pero a la vez satisfactorio; al que a la vez pueden integrarse cada vez más elementos tanto en el área investigativa como en el desarrollo del producto final, que es obtener una base de datos integrada al sistema operativo para lograr una mayor eficiencia y rapidez de procesos.

13. POSIX es el acrónimo de Portable Operating System Interface, viniendo la X de UNIX con el significado de la herencia de la API (Se traduciría como Sistema Operativo Portable basado en UNIX).
14. Tecnología de almacenamiento de MySQL que permite transacciones el tipo ACID, bloqueo de registro e integridad de referencia.
15. Tecnología de almacenamiento de MySQL basado en ISAM.

Referencias

Dean, Jeff, Bruno Gomes Pessanha, Nicolai Langfeldt, Steven Pritchard y James Stanger (2006).

Linux Kernel Organization (2006), *The Linux Kernel Archives* Consultada el 4 de diciembre 2006.

López Camacho, Vicente, (2001) *Linux. Guía de Instalación y administración*. McGraw-Hill.

LPI Linux Certification in a Nutshell, 2nd Edition.

Martinez Castaño, Juan Antonio, (1999) *El sistema de fichero virtual (VFS) de Linux* Consultada el 4 de diciembre 2006.

Martínez. Rafael (1998-2006), *Kerne/Núcleo* Consultada el 4 de diciembre 2006.

Mysql AB (1995-2006) *15.1Panoramica de InnoDB del manual de referencia de Mysql* en Consultada el 4 de diciembre 2006.

Prades González, Alfredo, Daniel Pecos Martínez y David Díez Muñoz (1999/2000), *El Kernel* Consultada el 4 de diciembre 2006.

Prime Media Press (2003),*"Oracle y HP establecen récord mundial en prueba TPC-C con Linux"* Consultada el 4 de diciembre 2006.

Universidad de las Palmas de Gran Canaria, S.Candela, R.García, G.Padrón (1994-2006) *Anatomía del núcleo de Linux 2.4. Lecciones de Linux* Consultada el 4 de diciembre 2006.

Universidad de las Palmas de Gran Canaria, José María Rodríguez Rodríguez, Xerach Sarda Morales, (2003) *Inode* Consultada el 4 de diciembre 2006.

Universidad Nacional de Rosario Argentina, Maximiliano Cristiá, Gisela Giusti, Felipe Manzano, *A MLS Linux Prototype Called Lisex* Consultada el 4 de diciembre 2006

Wikipedia (2006) Consultada el 4 de diciembre 2006.

Zabaljáuregui, Matías, *Concurrencia y mecanismos de sincronización en el kernel Linux* Consultada el 4 de diciembre 2006.

Zator Systems(2000-2006)*, Sistema Unix/Linux* Consultada el 4 de diciembre 2006.