Universidad Nacional de La Matanza
Florencio Varela 1903 - San Justo - Buenos Aires - Argentina

Ingeniería en Informática

Sistemas Operativos

Trabajo Investigación

Administración del sistema de archivos EXT2 en SODIUM

Equipo Docente	Nicanor Casas
	Graciela De Luca
	Waldo Valiente
	Gerardo Puyo
	Sergio Martín
	Federico Díaz
	Sebastian Deuteris

GRUPO: 06
Días de Cursada: Miércoles/*Viernes*

Alumnos		
Apellido	**Nombre**	**DNI**
Lara	Natalia	27.027.860
Quevedo	Sebastian	33.459.054
Suárez Cano	David	32.723.049

Acreditación:

Instancia	Fecha	Calificación
PRE-ENTREGA		
ENTREGA	14/06/2012	
FINAL	12/07/2012	

Índice

Desarrollo

Objetivo

Se plantea como objetivo del presente trabajo de investigación, la implementación y puesta en funcionamiento del sistema de archivos EXT2 en el sistema operativo SODIUM.
Además, se buscará que los sistemas de archivos previamente implementados en SODIUM sigan funcionando de la misma manera y que tanto el Virtual File System como los programas que se ejecutan para responder a los comandos sean modificados lo menos posible.

Alcance

La nueva funcionalidad que se pretende agregar al sistema operativo SODIUM abarca desde la creación de la imagen formateada en EXT2, el montaje de la misma, reconocimiento por parte del módulo Virtual File System (VFS), la ejecución de los comandos mkdir, ls, cd, cat y finalmente su desmontaje. Los comandos mencionados incluyen el tratamiento de las rutas absolutas y relativas.

Resumen del documento

Este documento comienza con el planteo del objetivo y alcance del trabajo de investigación en el que se basa. Se prosigue con el enunciado del conjunto de modificaciones y agregados realizados a los programas, bibliotecas y archivos fuentes incluidos en la versión 2011 del SDK de SODIUM. Se continúa con la documentación generada para describir cada función y programa creados. Luego, se incluye un lote de pruebas para verificar que se ha logrado un correcto entendimiento e implementación de EXT2. Finalmente, se proponen mejoras al proyecto actual, que bien podrían convertirse en objetivos de futuros trabajos de investigación, y conclusiones que exponen los problemas surgidos durante el desarrollo. También se incluye un apéndice con procedimientos útiles para la realización de las pruebas.

Práctica

-Introducción práctica

Se utiliza SODIUM para comprender, implementar y probar las características del sistema de archivos EXT2, adaptándolo a los requisitos del sistema de archivos virtual presente.

-Resumen de modificaciones

Se incluyen entre corchetes las líneas específicas donde encontrar las modificaciones.

Makefiles

Ruta de archivo:

- VFS2011/fuentes/kernel/Makefile

Descripción:

- Se agrega "ext2_fs.o" [36] y "ext2.o" [37] en el tag "OBJETOS_FS".

Programas

Ruta de archivo:

- VFS2011/fuentes/configurar.sh

Descripción:

- Se agrega soporte para el tipo de sistema de archivos EXT2 en la opción 8 [455-457].
- Se agrega capacidad de creación de una imagen EXT2 [547-559].

Funciones

Ruta de archivo:

- VFS2011/fuentes/include/kernel/fs/vfs.h

Descripción:

- Se agrega "#include <kernel/fs/ext2_fs.h>" dentro del conjunto de includes [14].
- Se agrega comentario y declaración de la función "pcFnIdentificarFS" [106-107].
- Se agrega comentario y declaración de la función "pcFnLeerSBEXT2" [110-111].
- Se agrega comentario y declaración de la función "vFnEXT2_CargarSBEXT2" [122-123].

Ruta de archivo:

- VFS2011/fuentes/kernel/fs/vfs.c

Descripción:

- Se agrega "#include <kernel/fs/ext2.h>" dentro del conjunto de includes [26].
- En la función "vFnIniciarVFS" [157]:
  - se agrega registro del tipo de sistema de archivos EXT2 en el VFS [176].
  - se modifica forma de detectar el tipo de sistema de archivos [182].
  - se modifica forma de montar el sistema de archivos detectado [184].
- Se agregan comentarios para doxygen y definición de la función "pcFnIdentificarFS" [196-216].
- Se agregan comentarios para doxygen y definición de la función "pcFnLeerSBEXT2" [262-298].
- En la función "vFnAgregarFileSystemType" [308]:
  - se agrega registro del tipo de sistema de archivos EXT2 en el VFS [354-357].
- Se agregan comentarios para doxygen y definición de la función "vFnEXT2_CargarSBEXT2" [779-825].

-Resumen de agregados

Bibliotecas nuevas

Ruta de archivo:

- VFS2011/fuentes/include/kernel/fs/ext2.h
- VFS2011/fuentes/include/kernel/fs/ext2_fs.h

Objetos nuevos

Ruta de archivo:

- VFS2011/fuentes/kernel/fs/ext2.c
- VFS2011/fuentes/kernel/fs/ext2_fs.c

-Documentación de las funciones y programas

Funciones en vfs.c

char * pcFnIdentificarFS (char *)
char * pcFnLeerSBEXT2 (char *)
void vFnEXT2_CargarSBEXT2 (stuSuperBloque *)

Funciones en ext2_fs.c

int iFnMontarEXT2 (stuInfoEXT2*)
int iFnDesmontarEXT2 (stuInfoEXT2 *)
int iFnMkdirEXT2 (stuInfoEXT2 *, char *)
int iFnCatEXT2(stuInfoEXT2 *, char *, void **)
int iFnCdEXT2(struct stuInfoEXT2 *, char *)
int iFnLsEXT2 (stuInfoEXT2 *, char *, stuListaArch **)
char * pcFnObtenerTipoArchivo (u16)
int iFnDiasTranscurridos (int, int)
void vFnConvertirFechaHoraUnix (u32 , stuFecha *, stuHora *)

Funciones en ext2.c

int iFnLeerSuperbloqueEXT2 (stuInfoEXT2*)
int iFnEscribirSuperBloqueEXT2 (stuInfoEXT2 *)
int iFnLeerBloqueEXT2 (stuInfoEXT2 *, byte *, int)
int iFnEscribirBloqueEXT2 (stuInfoEXT2 *, byte *, int)
int iFnObtenerCantGruposEXT2 (stuInfoEXT2 *)
int iFnLeerTablaDescGrupoEXT2 (stuInfoEXT2 *)
int iFnEscribirTablaDescGrupoEXT2 (stuInfoEXT2 *)
int iFnLeerInodoEXT2 (stuInfoEXT2 *, stuInodoEXT2 *, int)
int iFnEscribirInodoEXT2 (stuInfoEXT2 *, stuInodoEXT2 *, int)
int iFnLeerBloquesDeDatosEXT2 (stuInfoEXT2 *, stuInodoEXT2 *, byte *)
int iFnLeerIndSimpleEXT2 (stuInfoEXT2 *, u32, int *, int *, byte *)
int iFnLeerIndDobleEXT2 (stuInfoEXT2 *, u32, int *, int *, byte *)
int iFnLeerIndTripleEXT2 (stuInfoEXT2 *, u32, int *, int *, byte *)
int iFnEstablecerDirActualEXT2 (stuInfoEXT2 *pstuInfoEXT2Actual, int)
int iFnCrearDirectorioEXT2 (stuInfoEXT2 *, char *)
int iFnObtenerNroInodoLibreEXT2 (stuInfoEXT2 *, int)
int iFnObtenerNroBloqueLibreEXT2 (byte *)
int iFnCrearEntDirEXT2 (stuInfoEXT2 *, byte *, stuEntDirEXT2 *, int)
int iFnReservarInodoEXT2 (stuInfoEXT2 *, int, int)
int iFnSituarseEnRutaEXT2(stuInfoEXT2 *, char *)
int iFnObtenerCantNombresEnRutaEXT2(char *);
int iFnObtenerLongNombreEnRutaEXT2(char *, int);
void vFnObtenerNombreEnRutaEXT2 (char *, int, char *);
int iFnObtenerNroInodoEnDirEXT2(stuInfoEXT2 *, char *, u16);

Funciones en vfs.c:

pcFnIdentificarFS

- Descripción:
  Llama a pcFnLeerPBP y pcFnLeerSB_EXT2 para detectar el tipo de File System.
- Retorno:
  Tipo de File System detectado.
- Parámetros:
  pcTipoFS Almacena el tipo de File System detectado.
- Se le envía pcTipoFS (puntero a char), el cual luego tomará como parámetro la función pcFnLeerSB_EXT2 al ser invocada, esta última función al identificar el File System EXT2 guardará en pcTipoFS la cadena "EXT2".
  Ejemplo:
  pcFnIdentificarFS(pcTipoFS)
  Entonces luego del retorno: pcTipoFS="EXT2"

pcFnLeerSBEXT2

- Descripción:
  Lee el superbloque de EXT2 (offset 1024) e identifica el File System a través del "Magic Number".
- Retorno:
  Tipo de File System detectado (se espera EXT2).
- Parámetros:
  pcTipoFS Almacena el tipo de File System detectado (se espera EXT2)
- Se le envía un puntero a char para guardar el nombre del File System identificado. Esta función lee el sector 2 de la imagen de disco (el cual contiene el superbloque) y guarda en pcTipoFS el tipo de File System según el Magic Number del superbloque. Si el Magic Number es "0xef53" el Sistema de Archivo detectado es EXT2.
  Ejemplo:
  pcFnLeerSBEXT2(pcTipoFS)
  Entonces luego del retorno: pcTipoFS="EXT2"

vFnEXT2_CargarSBEXT2

- Descripción:
  Función que carga el superbloque del sistema de archivos EXT2.
- Retorno:
  void
- Parámetros:
  pstuSuperBloqueNuevo Referencia al superbloque a cargar.
- Se le envía un puntero a la estructura de Superbloque del VFS implementado anteriormente y se carga con la estructura de tipo stuInfoEXT2 que tiene información del sistema de archivo EXT2.
  Esta función es referenciada por la función vFnAgregarFileSystemType en donde se carga

la información del tipo de File System y se lo agrega en la lista de tipos de sistema de archivos reconocidos por Sodium.

Funciones en ext2_fs.c:

iFnMontarEXT2

- Descripción:
  Obtiene la información necesaria para manejar un File System EXT2.
- Retorno:
  0 si termino correctamente
- Parámetros:
  pstuInfoEXT2Actual Almacena la información del File System EXT2.
- Se le envía un puntero a la estructura stuInfoEXT2 (estructura principal para manejar el File System EXT2) para copiar en ella el superbloque y la Global Descriptor Table. Para realizar esto se hace referencia a las funciones iFnLeerSuperbloqueEXT2() y iFnLeerTablaDescGrupoEXT2(). Además establece como directorio actual el directorio raíz. Esta función es referenciada al iniciar el Virtual File System y cuando se ejecuta el comando "montar".
  Ejemplo: Cmd>montar EXT2
  Monta EXT2 en la primera unidad libre.

iFnDesmontarEXT2

- Descripción:
  Desmonta el sistema de archivos EXT2.
- Retorno:
  0 si termino correctamente
- Parámetros:
  pstuInfoEXT2AEliminar Almacena la información del File System EXT2.
- Se le envía un puntero a la estructura stuInfoEXT2 (estructura principal para manejar el File System EXT2), graba en el floppy el contenido actual de la imagen luego de haber operado sobre el File System y libera la memoria asignada a las estructuras de manejo del Filesystem. Se ejecuta cuando se ejecuta el comando
  Ejemplo: Cmd>desmontar A
  De esta manera se invoca a la función **iFnDesmontarEXT2().**
  Luego de unos segundos se muestra por pantalla el mensaje de desmontado.
  Para desmontar la unidad A debe ser previamente montada por un segundo file system.

iFnMkdirEXT2

- Descripción:
  Ejecuta el comando mkdir para EXT2.
- Retorno:
  0 si termino correctamente
- Parámetros:
  pstuInfoEXT2Actual Almacena la información del File System EXT2.

stNombreDir Nombre o path absoluto del directorio a crear.

- Se le envía un puntero a la estructura stuInfoEXT2 (estructura principal para manejar el File System EXT2) y el Nombre o path del directorio a crear, si contiene solo el nombre del directorio, el mismo se creara en el directorio actual.
  Ejemplo: Cmd>mkdir dir
  De esta manera se invoca a la función **iFnMkdirEXT2()** y se crea una nueva entrada en el directorio actual.

iFnCatEXT2

- Descripción:
  Ejecuta el comando cat para EXT2.
- Retorno:
  Tamaño del archivo en bytes o -1 en caso de error.
- Parámetros:
  pstuInfoEXT2ALeer Almacena la información del File System EXT2.
  stRuta Ruta del archivo a leer.
  pvData Contiene los datos del archivo.
- Se le envía por parámetro un puntero a la estructura stuInfoEXT2 (estructura principal para manejar el File System EXT2) y el path del archivo que se desea leer. Si el path es válido se copian los bloques de datos de dicho archivo en pvData y el Virtual File System se encarga de mostrar el contenido de pvData por pantalla.

iFnCdEXT2

- Descripción:
  Ejecuta el comando cd.
- Retorno:
  0 en caso de éxito. -1 si se trata de una ruta no valida.
- Parámetros:
  pstuInfoEXT2NuevoDir Contiene información para manejar el File System.
  stRuta Ruta del directorio que será el directorio actual.
- Se le envía por parámetro un puntero a la estructura stuInfoEXT2 (estructura principal para manejar el File System EXT2) y el path de un directorio. Si el path es válido se copian los bloques de datos de dicho directorio en el campo pbyDirActual de stuInfoEXT2 por medio de la función **iFnSituarseEnRutaEXT2().** Si el path no es válido se mantiene el directorio actual.

iFnLsEXT2

- Descripción:
  Ejecuta el comando ls para EXT2.
- Retorno:
  0 si termino correctamente
- Parámetros:

pstuInfoEXT2ALeer Almacena la información del File System EXT2.
pcRuta Path absoluto del directorio a listar.
ppstuListaArchLeidos Almacena la lista de archivos leídos.

- Se le envía un puntero a la estructura stuInfoEXT2 (estructura principal para manejar el File System EXT2), el path del directorio que se quiere listar y una estructura stuListaArch que utiliza el Virtual File System para ejecutar el comando ls.
  Ejemplo: Cmd>ls
  De esta manera se invoca a función **iFnLsEXT2** con PcRuta vacío como parámetro y se muestra por pantalla los directorios y archivos del directorio actual.

pcFnObtenerTipoArchivo

- Descripción:
  Obtiene el tipo de archivo que se muestra al ejecutar el comando ls para EXT2.
- Retorno:
  0 si termino correctamente
- Parámetros:
  usTipoDir Contiene el tipo de archivo asociado al inodo.
- Se le envía el tipo de archivo, el cual que se encuentra en los bits más significativos del campo usTipoYPermisos de la estructura de un inodo y devuelve en string el tipo de archivo. Este retorno puede ser: FILE, DIR, FIFO, LINK, SOCKET o DEVICE, y son los valores posibles de la columna "Atri" de la lista del directorio que se muestra al ejecutar el comando ls. Esta función es referenciada en la función **iFnLsEXT2.**

iFnDiasTranscurridos

- Descripción:
  Función interna para el cálculo de conversión de fecha.
- Retorno:
  Los días transcurridos desde el inicio de ese año.
- Parámetros:
  iMesesTranscurridos cantidad de meses ya pasados. Ej: Si es Enero, pasaron 0 meses. Si es Febrero, paso un mes. Si termino el año, pasaron 12 meses.
  iAnioActual El año en curso. Permite saber si es bisiesto.
- Esta función es referenciada por **vFnConvertirFechaHoraUnix().**

vFnConvertirFechaHoraUnix

- Descripción:
  Convierte el tiempo del formato Unix a estructuras de fecha stuFecha y stuHora.
- Retorno:
  void
- Parámetros:
  uiTiempoAConvertir Fecha y Hora en formato Unix
  pstuFechaConvertida Fecha convertida a estructura stuFecha.
  pstuHoraConvertida Horario convertido a estructura stuHora..

- Esta función es utilizada para convertir una fecha POSIX (la cual es definida como la cantidad de segundos transcurridos desde el 1 de enero de 1970) al siguiente formato:

  Dia/Mes/Año Hora:Min:Seg

  La fecha convertida se puede ver en la columna "fecha" de la lista de directorio resultante de ejecutar el comando ls.
  Ejemplo:
  vFnConvertirFechaHoraUnix (**1340145757**, pstuFechaConvertida, pstuHoraConvertida)
  Luego de la función la fecha convertida queda de esta manera:
  pstuFechaConvertida->iDay = 18
  pstuFechaConvertida->iMonth = 6
  pstuFechaConvertida->iYear = 2012
  pstuHoraConvertida->iHour = 22
  pstuHoraConvertida->iMinute = 47
  pstuHoraConvertida->iSecond =37
  Esta fecha es obtenida por la función que ejecuta el comando ls para EXT2 y muestra pantalla en la columna fecha: **18/6/2012 22:47:37**
  Esta función es referenciada por **iFnLsEXT2().**

Funciones en ext2.c:

iFnLeerSuperbloqueEXT2

- Descripción:
  Obtiene el superbloque del File System actual.
- Retorno:
  0 si terminó correctamente.
- Parámetros:
  pstuInfoEXT2Leida Contiene información para manejar el File System.
- Se le envía un puntero a stuInfoEXT2 (estructura principal de nuestra implementación de EXT2). Esta función lee el sector 2 de la imagen de disco (el cual contiene el superbloque) y se copia en la estructura pstuInfoEXT2Leida para poder manejar el Sistema de Archivo EXT2. Esta función es referenciada por **iFnMontarEXT2()**.

iFnEscribirSuperbloqueEXT2

- Descripción:
  Escribe el superbloque del File System actual.
- Retorno:
  0 si terminó correctamente.
- Parámetros:
  pstuInfoEXT2AEscribir Contiene información a guardar del File System.
- Se le envía un puntero a stuInfoEXT2 (estructura principal de nuestra implementación de EXT2). Esta función escribe el superbloque de la estructura pstuInfoEXT2AEscribir en el sector 2 de la imagen de disco (el cual contiene el superbloque).
  Esta función es referenciado por **iFnCrearDirectorioEXT2()**.

iFnLeerBloqueEXT2

- Descripción:
  Lee un bloque.
- Retorno:
  0 si terminó correctamente.
- Parámetros:
  pstuInfoEXT2ALeer Contiene información para manejar el File System.
  pbyBuffer Almacena el bloque leído.
  iBloque Número del bloque a leer.
- Se le envía como parámetro un puntero a stuInfoEXT2 para calcular la cantidad de sectores por bloque (utilizando la información del tamaño de bloque del Sistema de Archivo EXT2). Esta función lee en el número bloque indicado por la variable iBloque y copia su contenido en pbyBuffer.

iFnEscribirBloqueEXT2

- Descripción:
  Escribe un bloque.
- Retorno:
  0 si terminó correctamente.
- Parámetros:
  pstuInfoEXT2ALeer Contiene información para manejar el File System.
  pbyBuffer Almacena el bloque a escribir.
  iBloque Número del bloque a escribir.
- Se le envía como parámetro un puntero a stuInfoEXT2 para calcular la cantidad de sectores por bloque (utilizando la información del tamaño de bloque del File System EXT2) y el parámetro pbyBuffer contiene la información a escribir.
  Esta función escribe en el número bloque indicado por la variable iBloque todo el contenido de pbyBuffer.
  Consideraciones:
  El tamaño de la porción de memoria a la que apunta pbyBuffer debe ser por lo menos igual al tamaño de un bloque del sistema de archivo en cuestión. Ej: si trabajo con bloques de 1024 bytes, pbyBuffer debe apuntar a una porción de memoria de por lo menos 1024 bytes.
  Esta función no se tendría que usar directamente, sino siendo llamada por otras funciones. Sería muy peligroso escribir cualquier bloque directamente.
  Esta función es referenciada por **iFnCrearDirectorioEXT2()**, **iFnEscribirInodoEXT2()**, **iFnEscribirTablaDescGrupoEXT2()** y **iFnReservarInodoEXT2()**.

iFnObtenerCantGruposEXT2

- Descripción:
  Obtiene la cantidad de grupos en el File System.
- Retorno:
  Cantidad de grupos en el File System.

- Parámetros:
  pstuInfoEXT2ALeer Contiene información para manejar el File System.
- Se le envía como parámetro un puntero a la estructura stuInfoEXT2 y realiza el calculo de la cantidad de grupos del File System EXT2 utilizando la información de algunos campos de dicha estructura.

iFnLeerTablaDescGrupoEXT2

- Descripción:
  Obtiene Tabla de Descriptores de Grupo del File System actual.
- Retorno:
  0 si terminó correctamente.
- Parámetros:
  pstuInfoEXT2ALeer Contiene información para manejar el File System.
- La estructura stuInfoEXT2 tiene un puntero a la estructura stuDescGrupoEXT2Tabla, la cual representa una entrada de la Tabla de Descriptores de Grupo. Esta función asigna las posiciones de memoria para cada una de las entradas (Tabla completa) y se copia cada entrada de la Tabla de Descriptores de Grupo en la posición de memoria correspondiente reservada para ella. Esta función es referenciada por **iFnMontarEXT2()**.

iFnEscribirTablaDescGrupoEXT2

- Descripción:
  Escribe la Tabla de Descriptores de Grupo del File System actual.
- Retorno:
  0 si terminó correctamente.
- Parámetros:
  pstuInfoEXT2ALeer Contiene información para manejar el File System.
- Se copia el contenido de cada posición de memoria reservada para la Tabla de Descriptores de Grupo (apuntada por pstuDescGrupoEXT2Tabla dentro de la estructura stuInfoEXT2) desde el bloque de inicio de la Tabla (bloque siguiente al superbloque) de la imagen de disco hasta completar su tamaño.
  Esta función es referenciada por **iFnCrearDirectorioEXT2()** para actualizar el estado de la Tabla de Descriptores luego de crear un directorio nuevo.

iFnLeerInodoEXT2

- Descripción:
  Obtiene un Inodo pasado por parámetro.
- Retorno:
  0 en caso de éxito.
- Parámetros:
  pstuInfoEXT2ALeer Contiene información para manejar el File System.
  pstuInodoEXT2Leido Contiene información del Inodo leído.
  iNroInodo Contiene el numero del Inodo a leer..
- Se le envía como parámetro un puntero a stuInfoEXT2, un puntero a stuInodoEXT2 y el

número de Inodo que se quiere leer. Utilizando algunos campos de la estructura apuntada por pstuInfoEXT2ALeer se calcula en que bloque y en qué offset de ese bloque está el Inodo indicado en iNroInodo, y se copia a partir de ese offset el Inodo en la estructura apuntada por pstuInodoEXT2Leido, que contiene toda la información de un Inodo.
Esta función es referenciada por **iFnCrearDirectorioEXT2()**, **iFnEstablecerDirActualEXT2()**, **iFnLsEXT2()** y **iFnObtenerNroInodoEnDirEXT2()**.

iFnEscribirInodoEXT2

- Descripción:
  Escribe un Inodo pasado por parámetro.
- Retorno:
  0 en caso de éxito.
- Parámetros:

pstuInfoEXT2ALeer	Contiene información para manejar el File System.
pstuInodoEXT2AEscribir	Contiene información del Inodo a escribir.
iNroInodo	Contiene el número del Inodo a escribir.

- Se le envía como parámetro un puntero a stuInfoEXT2, un puntero a stuInodoEXT2 y el número de Inodo que se quiere escribir en la imagen de disco. Utilizando algunos campos de la estructura apuntada por pstuInfoEXT2ALeer se calcula en que bloque y en qué offset de ese bloque está el Inodo indicado en iNroInodo, y se copia dicho Inodo a partir de ese offset en la porción de la tabla de Inodos.
  Esta función es referenciada por **iFnCrearDirectorioEXT2()** para actualizar el estado de la Tabla de Inodos luego de crear un directorio nuevo.

iFnLeerBloquesDeDatosEXT2

- Descripción:
  Obtiene los datos de los bloques de datos referenciados por un inodo.
- Retorno:
  0 en caso de éxito.
- Parámetros:

pstuInfoEXT2ALeer	Contiene información para manejar el File System.
pstuInodoEXT2ALeer	Contiene el inodo del archivo.
pbyArchivo	Almacena los datos leídos del archivo.

- Lee todos los bloques de datos de un archivo asociado al inodo apuntado por pstuInodoEXT2ALeer.
  Para llegar a cabo esta lectura de datos, lee primero los 12 bloques directos a datos del vector **uiPtrDatos**, luego teniendo en cuenta el tamaño del archivo definido en **uiTamanio**, esta función hace referencia a las funciones que leen las indirecciones del inodo: **iFnLeerIndSimpleEXT2()**, **iFnLeerIndDobleEXT2()**, **iFnLeerIndTripleEXT2()**.
  En cada lectura se almacena el contenido de cada bloque de datos en **pbyArchivo**.
  Esta función es referenciada por la función **iFnEstablecerDirActualEXT2()** para copiar los bloques del directorio actual en pbyDirActual de la estructura stuInfoEXT2 apuntada por pstuInodoEXT2ALeer.

iFnLeerIndSimpleEXT2

- Descripción:
- Obtiene los datos de los bloques de datos de la indirección simple.
- Retorno:
  0 en caso de éxito.
- Parámetros:

pstuInfoEXT2ALeer	Contiene información para manejar el File System.
uiPtrIndSimple	Numero de bloque de indirección simple.
piBytesLeidos	Cantidad de bytes leídos del archivo.
piBytesRestantes	Cantidad de bytes restantes a ser leídos.
pbyArchivo	Almacena los datos leídos del archivo.

- Lee los bloques de indirección simple indicados en el bloque apuntado por **uiPtrIndSimple** (campo de la estructura stuInodoEXT2) y almacena el contenido de cada bloque de datos en **pbyArchivo**. Esta función es referenciada por **iFnLeerBloquesDeDatosEXT2()**.

iFnLeerIndDobleEXT2

- Descripción:
- Obtiene los datos de los bloques de datos de la indireccion doble.
- Retorno:
  0 en caso de éxito.
- Parámetros:

pstuInfoEXT2ALeer	Contiene información para manejar el File System.
uiPtrIndDoble	Numero de bloque de indirección doble.
piBytesLeidos	Cantidad de bytes leídos del archivo.
piBytesRestantes	Cantidad de bytes restantes a ser leídos.
pbyArchivo	Almacena los datos leídos del archivo.

- Lee los bloques de indirección doble a través del bloque apuntado por **uiPtrIndDoble** (campo de la estructura stuInodoEXT2) y almacena el contenido de cada bloque de datos en **pbyArchivo**. Esta función es referenciada por **iFnLeerBloquesDeDatosEXT2()**.

iFnLeerIndTripleEXT2

- Descripción:
- Obtiene los datos de los bloques de datos de la indirección triple.
- Retorno:
  0 en caso de éxito.
- Parámetros:

pstuInfoEXT2ALeer	Contiene información para manejar el File System.
uiPtrIndTriple	Numero de bloque de indirección triple.
piBytesLeidos	Cantidad de bytes leídos del archivo.
piBytesRestantes	Cantidad de bytes restantes a ser leidos.
pbyArchivo	Almacena los datos leídos del archivo.

- Lee los bloques de indirección triple a través del bloque apuntado por **uiPtrIndTriple** (campo de la estructura stuInodoEXT2) y almacena el contenido de cada bloque de datos en **pbyArchivo**. Este tipo de indirección se alcanza en archivos muy grandes.
  Esta función es referenciada por **iFnLeerBloquesDeDatosEXT2()**.

iFnEstablecerDirActualEXT2

- Descripción:

Establece el directorio actual.

- Retorno:
  0 si terminó correctamente. 1 en caso contrario.
- Parámetros:
  pstuInfoEXT2Actual Contiene información para manejar el File System.
- iNroInodo Número de inodo del directorio que sera el directorio actual.
- Se le envía por parámetro el número de inodo correspondiente a un directorio y se copian los bloques de datos de dicho directorio en el campo pbyDirActual de stuInfoEXT2, la estructura principal que maneja el File System.
  Esta función es referenciada por **iFnMontarEXT2()** y **iFnLsEXT2()**.

iFnCrearDirectorioEXT2

- Descripción:
  Crea un directorio.
- Retorno:
  0 si termino correctamente
- Parámetros:
  pstuInfoEXT2Actual Contiene información para manejar el File System.
  stNombreDir Nombre o path absoluto del directorio a crear.
- Se le envía un puntero a la estructura stuInfoEXT2 (estructura principal para manejar el Sistema de Archivo EXT2) y el nombre o path del directorio a crear, si contiene solo el nombre del directorio, el mismo se creara en el directorio actual.
  Esta función es referenciada por **iFnMkdirEXT2()**.

iFnObtenerNroInodoLibreEXT2

- Descripción:
  Obtiene el número del primer inodo libre dentro de un grupo.
- Retorno:
  Número del primer inodo libre encontrado.
- Parámetros:
  pstuInfoEXT2ALeer Contiene información para manejar el File System.
  iNroGrupo Numero del grupo donde se quiere encontrar el inodo libre.
- Recorre el mapa de bits de inodos hasta encontrar un inodo libre, es decir; el primer bit que está en cero, y devuelve el número de inodo que le corresponde a ese bit.
  Esta función es referenciada por **iFnCrearDirectorioEXT2()**.

iFnObtenerNroBloqueLibreEXT2

- Descripción:
  Obtiene el número del primer bloque de datos libre dentro de un mapa de bits de bloques

libres.

- Retorno:
  Número del primer bloque de datos libre.
- Parámetros:
  pbyMapBitBlqLib Mapa de bits de bloques libres.
- Recorre el mapa de bits de bloques hasta encontrar un bloque libre, es decir; el primer bit que está en cero, y devuelve el número de bloque que le corresponde a ese bit.
  Esta función es referenciada por **iFnCrearDirectorioEXT2()**.

iFnCrearEntDirEXT2

- Descripción:
  Crea una entrada de directorio en un directorio dado.
- Retorno:
  0 en caso de éxito o 1 en caso de no disponer de tamaño suficiente en el directorio para la nueva entrada.
- Parámetros:
  pstuInfoEXT2ALeer Contiene información para manejar el File System.
  pbyDirectorio Directorio donde crear la entrada.
  pstuEntDirEXT2Nueva Entrada de directorio a crear.
  iTipoEntDir Tipo de entrada de directorio.
- Se le envía por parámetro el puntero pstuEntDirEXT2Nueva a stuEntDirEXT2 (estructura que representa una entrada de directorio) con la entrada que se desea crear, recorre cada entrada del directorio almacenadas en **pbyDirectorio**, es decir el archivo (ya que un directorio es un tipo especial de archivo) y copia el contenido de la nueva entrada en **pbyDirectorio**.
  Esta función es referenciada por **iFnCrearDirectorioEXT2()**.

iFnReservarInodoEXT2

- Descripción:
  Reserva un inodo.
- Retorno:
  0 en caso de éxito.
- Parámetros:
  pstuInfoEXT2Actual Contiene información para manejar el File System.
  iNroGrupo Numero del grupo donde se quiere reservar el inodo.
  iNroInodo Numero del inodo a reservar en el grupo.
- Esta función
- Reserva un inodo, es decir; marca como "ocupado" el bit (en el mapa de bits de inodos) que le corresponde al número inodo de la variable **iNroInodo** y decrementa la cantidad de inodos libres del File System, tanto en el superbloque como en la tabla de descriptores de grupo.
  Esta función es referenciada por **iFnCrearDirectorioEXT2()**.

iFnSituarseEnRutaEXT2

- Descripción:
  Establece el directorio actual a partir de una ruta.
- Retorno:
  0 en caso de éxito. -1 si se trata de una ruta no valida.
- Parámetros:
  pstuInfoEXT2ALeer Contiene información para manejar el File System.
  stRuta Ruta del directorio que será el directorio actual.
- Se le envía por parámetro un puntero a la estructura stuInfoEXT2 (estructura principal para manejar el File System EXT2) y el path de un directorio. Si el path es válido se copian los bloques de datos de dicho directorio en el campo pbyDirActual de stuInfoEXT2. Si el path no es válido se mantiene el directorio actual.
  Esta función es referenciada por **iFnMkdirEXT2()**, **iFnLsEXT2()** y **iFnCdEXT2()**.

iFnObtenerCantNombresEnRutaEXT2

- Descripción:
  Obtiene la cantidad de nombres en una ruta.
- Retorno:
  Cantidad de nombres que conforman la ruta.
- Parámetros:
  stRuta Contiene la ruta a ser analizada.
- Se le envía un string correspondiente a un path, y devuelve la cantidad "nombres" del path. El último nombre puede ser de un directorio o de un archivo y si hay más de un nombre los anteriores son directorios. En caso de una ruta absoluta, se considera al primer '/' como el nombre del directorio raíz.
  Esta función es referenciada por **iFnMkdirEXT2()** y **iFnSituarseEnRutaEXT2()**.

iFnObtenerLongNombreEnRutaEXT2

- Descripción:
  Obtiene la longitud de un nombre en una ruta.
- Retorno:
  Longitud del nombre.
- Parámetros:
  stRuta Contiene la ruta a ser analizada.
  iNroNombre Numero del nombre dentro de la ruta del que se quiere obtener su longitud. La numeración comienza en 0.
- Se le envía un string correspondiente a un path, y devuelve la longitud de un nombre de directorio o archivo que forma parte del path. El nombre dentro del path analizado es el indicado por la variable iNroNombre.
  Esta función es referenciada por **iFnMkdirEXT2()** y **iFnSituarseEnRutaEXT2()**.

vFnObtenerNombreEnRutaEXT2

- Descripción:
  Obtiene un nombre en una ruta.
- Retorno:
  void
- Parámetros:
  stRuta Ruta de donde obtener el nombre.
  iNroNombre Numero del nombre a obtener, dentro de la ruta. La numeración comienza en 0.
  stNombre Almacena el nombre correspondiente a iNroNombre.
- Se le envía un string correspondiente a un path, el número dentro del path del nombre (directorio o archivo) que se quiere obtener y se copia dicho nombre en stNombre.
  Esta función es referenciada por **iFnMkdirEXT2()** y **iFnSituarseEnRutaEXT2()**.

iFnObtenerNroInodoEnDirEXT2

- Descripción:
  Obtiene el número de inodo a partir de un nombre y un tipo de archivo en el directorio actual.
- Retorno:
  Numero de inodo correspondiente al nombre y tipo de archivo. -1 si no se encontró una entrada con el nombre y tipo de archivo indicado.
- Parámetros:
  pstuInfoEXT2ALeer Almacena la información del File System EXT2.
  stNombre Nombre del archivo asociado al numero de inodo a obtener.
  usTipoDeArchivo Tipo de archivo asociado al numero de inodo a obtener.
- Se le envía por parámetro el nombre de un archivo o directorio, se busca dicho nombre en cada entrada del directorio actual y si lo encuentra devuelve el número de inodo que le corresponde a ese archivo o directorio. Si no lo encuentra devuelve -1.
  Esta función es referenciada por **iFnMkdirEXT2()** y **iFnSituarseEnRutaEXT2()**.

-Casos de Prueba

Verificar comportamiento del comando ls en SODIUM

Entrada:

- Archivo imagen de EXT2 en /build (imgFat.bin).

Proceso:

1) Ejecutar configurar.sh y elegir el sistema de archivos EXT2.
2) Ejecutar make clean.
3) Crear un archivo en imgFat.bin y obtener su nombre, tamaño, fecha de modificación y tipo (ver Apéndice).
4) Listar todo el contenido del directorio raíz de imgFat.bin (ver Apéndice), obtener los

datos desplegados para cada entrada (nombre, tamaño, fecha de modificación y tipo de archivo) y compararlos con los obtenidos en el paso anterior.

5) Ejecutar make test.
6) Ejecutar el comando ls en SODIUM y comparar los datos desplegados con aquellos obtenidos en los puntos 3 y 4.

Salida esperada:

- Los datos obtenidos desde la terminal Linux confirman los desplegados en pantalla por SODIUM.

Verificar el comportamiento del comando mkdir en SODIUM

Entrada:

- Archivo imagen de EXT2 en /build (imgFat.bin).

Proceso:

1) Ejecutar configurar.sh y elegir el sistema de archivos EXT2.
2) Ejecutar make clean.
3) Ejecutar make test.
4) Crear un directorio desde SODIUM, ejecutando mkdir <nombre del directorio>.
5) Desmontar el sistema de archivos EXT2 (ver Apéndice).
6) Listar el contenido del directorio raíz de sodium_fat12.img en /build (ver Apéndice).

Salida esperada:

- El directorio creado en el paso 4 aparece al momento de listar el directorio raíz de sodium_fat12.img en el paso 6.

Verificar cambios en el Superbloque

Entrada:

- Archivo imagen de EXT2 en /build (imgFat.bin).

Proceso:

1) Ejecutar configurar.sh y elegir el sistema de archivos EXT2.
2) Ejecutar make clean.
3) Obtener los datos del superbloque correspondientes a la cantidad de bloques libres, la cantidad de inodos libres y el tiempo de la última modificación (ver Apéndice).
4) Ejecutar make test.
5) Crear un directorio desde SODIUM, ejecutando mkdir <nombre del directorio>.
6) Desmontar el sistema de archivos EXT2 (ver Apéndice).
7) Ejecutar el paso 3, pero utilizando el archivo sodium_fat12.img de /build.

Salida esperada:

- La cantidad de inodos libres se reduce en 1.
- La cantidad de bloques libres se reduce en 1.
- El tiempo de modificación refleja el momento en que se creo el directorio en el paso 5.

Verificar cambios en el grupo

Entrada:

- Archivo imagen de EXT2 en /build (imgFat.bin).

Proceso:

1) Ejecutar configurar.sh y elegir el sistema de archivos EXT2.
2) Ejecutar make clean.
3) Obtener los datos del grupo correspondientes a la cantidad de bloques libres, la cantidad de inodos libres y la cantidad de directorios (ver Apéndice).
4) Ejecutar make test.
5) Crear un directorio desde SODIUM, ejecutando mkdir <nombre del directorio>.
6) Desmontar el sistema de archivos EXT2 (ver Apéndice).
7) Ejecutar el paso 3, pero utilizando el archivo sodium_fat12.img de /build.

Salida esperada:

- La cantidad de inodos libres en el grupo se reduce en 1.
- La cantidad de bloques libres en el grupo se reduce en 1.
- La cantidad de directorios en el grupo se incrementa en 1.

Verificar el comportamiento del comando cat en SODIUM

Entrada:

- Archivo imagen de EXT2 en /build (imgFat.bin).

Proceso:

1) Ejecutar configurar.sh y elegir el sistema de archivos EXT2.
2) Ejecutar make clean.
3) Crear un archivo vacío en imgFat.bin (ver Apéndice).
4) Crear un archivo lleno con un contenido específico en imgFat.bin (ver Apéndice).
5) Ejecutar make test.
6) Ejecutar el comando cat <ruta del archivo vacío> en SODIUM.

Ejecutar el comando cat <ruta del archivo lleno> en SODIUM

Salida esperada:

- En el caso del archivo vacío, se desplegará en la pantalla de SODIUM un mensaje indicando la situación del mismo.
- En el caso del archivo lleno, se desplegara en la pantalla de SODIUM el contenido del archivo.

Verificar el comportamiento del comando cd en SODIUM

Entrada:

- Archivo imagen de EXT2 en /build (imgFat.bin).

Proceso:

1) Ejecutar configurar.sh y elegir el sistema de archivos EXT2.
2) Ejecutar make clean.
3) Crear un directorio llamado **“carpeta”** en imgFat.bin (ver Apéndice).
4) Crear un archivo vacío llamado **“archivo”** en imgFat.bin (ver Apéndice).
5) Crear un archivo lleno llamado **“archivo”** dentro del directorio **“carpeta”** (ver Apéndice)
6) Ejecutar make test.

7) Ejecutar el comando **cat archivo** en SODIUM.
8) Ejecutar el comando **cd carpeta** en SODIUM.
9) Ejecutar el comando **cat archivo** en SODIUM.

Salida esperada:

- La ejecución del comando **cat archivo** difiere según el directorio en el que estamos situados. La primera ejecución del comando cat nos indica que no se pudo abrir el archivo ya que este está vacío. La segunda nos muestra el contenido del archivo. Aunque se esté utilizando el mismo comando, **cat archivo**, los resultados son diferentes debido a que el directorio actual es diferente. Justamente, es el correcto funcionamiento del comando **cd** en el punto 8 el que se encarga de esta última tarea.

Mejoras

Debido en parte a la falta del tiempo que se hubiera querido tener para realizar una implementación más completa, a continuación se proponen un conjunto de mejoras que bien podrían formar parte de los objetivos de futuros trabajos de investigación:

- Crear un entorno más amigable para realizar las pruebas, a través de comandos en la ayuda de SODIUM. Actualmente, para realizar cada prueba es necesario modificar el código fuente y volver a compilar todo el sistema operativo.

- Completar el desarrollo de todas las funciones requeridas por el VFS para manejar un sistema de archivos EXT2. También, en caso de considerarlo adecuado, incluir nuevas funciones.

- Acceder a un sistema de archivos EXT2 ubicado en un dispositivo externo (disquete, disco rígido, memoria usb) y dejar de hacerlo a través de ramfs. Esto implica un mayor desarrollo en el módulo de entrada–salida de SODIUM.

- Permitir el inicio de SODIUM en un sistema de archivos EXT2 al igual que actualmente ocurre con sistemas FAT.

- Implementar la administración del File System EXT3 basándose en las estructuras que utilizamos en nuestro desarrollo, ya que la diferencia sustancial que tiene con EXT2 es el mecanismo de *journaling*, el cual lleva un registro diario de la información necesaria para restablecer los datos afectados por una transacción en caso de que ésta falle. Es una capa adicional que mantiene un archivo de log de transacciones para integridad de datos. Este mecanismo de *journaling* además está incluido en otros File Systems que se pondrían implementar a futuro, como EXT4, NTFS, ReiserFS, Reiser4, XFS y JFS.

Conclusiones

Se intentó que la investigación y el desarrollo de las características de EXT2 en SODIUM fueran lo más fluida posible, pero surgieron algunos problemas que se pasarán a detallar a continuación:

- *Pérdida en el soporte al sistema de archivos FAT32 por parte del VFS*. Dado que el mecanismo para reconocer el tipo de sistema de archivos presente en imgFat.bin tomaba una aproximación "hard-coded" o "por descarte" para FAT32, al momento de incluir EXT2 (con una estructura diferente a la de los sistemas FAT) fue imposible seguir dándole soporte al primero sin retrasar considerablemente el desarrollo de la investigación.
- *Incapacidad de probar algoritmo para leer bloques de datos en la segunda y tercera indirección*. Debido a tener una implementación que utiliza ramfs para acceder al sistema de archivos EXT2 y las consiguientes limitaciones de espacio que esto implica, se decidió crear el archivo imgFat.bin con un tamaño de 60 KB (el menor que se pudo obtener). Esta configuración nos permite almacenar un archivo de hasta 51 KB (51 bloques de datos). Dado que la segunda indirección recién adquiere relevancia para tamaños de archivo mayores a 268 KB, el algoritmo correspondiente a la utilización de la misma nunca llega a ser ejecutado. Peor aún es el caso para la tercera indirección.
- *Inicio de SODIUM únicamente en floppy.* Actualmente logramos probar nuestra implementación de EXT2 utilizando el archivo imgFat.bin formateado en dicho sistema de archivos. El archivo imgFat.bin compone la imagen de SODIUM. Las mencionadas pruebas se hicieron al montar la imagen de SODIUM en floppy. No fue posible realizar la misma tarea en disco rígido. Al intentar ejecutarlo de esa manera nos muestra un error en el cual menciona que no es posible cargar imgFat.bin. Esta situación continúa ocurriendo sin importar el tipo de sistema de archivos en imgFat.bin.
- *Columnas mostradas por el comando ls*. El programa que ejecuta el listado de un directorio muestra columnas que fueron pensadas en su momento para FAT, como fue parte de nuestro objetivo modificar lo menos posible tanto el Virtual File System como los programas para los comandos, dejamos sin efecto algunos de estos campos que no aplican en nuestras estructuras de EXT2.
- *Destrucción de la imagen de SODIUM en sodium_fat12.img.* En el proceso de desmontar el sistema de archivos EXT2 desde SODIUM, el primero se graba en sodium_fat12.img. Dado que esta grabación se realiza a partir del comienzo de sodium_fat12.img, el tipo de sistema de archivos contenido en el mismo, es decir, FAT12, es reemplazado por EXT2. Al hacer esto, se pierde el acceso al resto de los archivos que componen la imagen de SODIUM.

Apéndice

Pasos para obtener datos del superbloque

1) Desde la terminal de Linux, situarnos en el directorio /build
2) Ejecutar **losetup /dev/loop0 imgFat.bin**. En caso de informarse que el dispositivo ya esta siendo usado, probar con otro dispositivo loop.
3) Ejecutar **dumpe2fs /dev/loop0** y obtener los datos del superbloque que quiera corroborar.
4) Ejecutar **losetup -d /dev/loop0**.

Pasos para obtener datos de la Tabla de Descriptores de Grupos

- Desde la terminal de Linux, situarnos en el directorio /build
- Ejecutar **losetup /dev/loop0 imgFat.bin**. En caso de informarse que el dispositivo ya esta siendo usado, probar con otro dispositivo loop.
- Ejecutar **dumpe2fs /dev/loop0** y obtener los datos de los grupos que quiera corroborar.
- Ejecutar **losetup -d /dev/loop0**.

Pasos para crear y obtener datos de un archivo

- Desde la terminal de Linux, situarnos en el directorio /build
- Ejecutar **losetup /dev/loop0 imgFat.bin**. En caso de informarse que el dispositivo ya esta siendo usado, probar con otro dispositivo loop.
- Crear un directorio en la carpeta /media, por ejemplo /media/disco.
- Montar la imagen EXT2 con el comando **mount /dev/loop0 /media/disco**.
- Crear un archivo dentro de /media/disco. Se proponen tres alternativas:
  - Crear un archivo vacío ejecutando el comando touch. Por ejemplo, **touch /media/disco/archivo**.
  - Crear un archivo lleno de un tamaño determinado pero sin un contenido específico ejecutando el comando dd. Por ejemplo, con **dd if=/dev/zero of=/media/disco/archivo bs=512 count=11** creamos un archivo de 5.5 KB (512 bytes x 11).
  - Crear un archivo lleno sin un tamaño determinado pero con un contenido específico. En este caso, una alternativa podría ser utilizar: **man cat > /media/disco/archivo**.
- Ejecutar el comando **stat /media/disco/archivo** para obtener los datos del archivo.
- Desmontar la imagen EXT2 con el comando **umount /media/disco**.
- Ejecutar **losetup -d /dev/loop0**.

Pasos para desmontar un sistema de archivos EXT2 en SODIUM

1) Montar una nueva unidad ejecutando **montar <tipo de sistema de archivos> <nombre del dispositivo>.** Por ejemplo, podemos ejecutar **montar FAT12B diskette**.
2) Ejecutar **montar (sin parametros)** para conocer el nombre de la unidad asignada en el paso anterior. La unidad asignada se identifica a través de una letra.
3) Posicionarse en la unidad asignada con **cd <nombre de unidad>.**. Por ejemplo, **cd B.**. Tener en cuenta que luego del nombre de la unidad debe escribirse un punto.
4) Desmontar el sistema de archivos EXT2 ingresando **desmontar <nombre de unidad>.**, donde el nombre de la unidad es aquel correspondiente al sistema de archivos EXT2

listado en el paso 2.

Pasos para listar el contenido de un directorio

Desde la terminal de Linux, situarnos en el directorio /build

Ejecutar **losetup /dev/loop0 imgFat.bin**. En caso de informarse que el dispositivo ya esta siendo usado, probar con otro dispositivo loop.

Crear un directorio en la carpeta /media, por ejemplo /media/disco.

Montar la imagen EXT2 con el comando **mount /dev/loop0 /media/disco**.

Ejecutar el comando **ls –la /media/disco** para listar todo el contenido del directorio. En este caso se lista el contenido del directorio raíz de nuestro sistema de archivos EXT2 en imgFat.bin.

Desmontar la imagen EXT2 con el comando **umount /media/disco**.

Ejecutar losetup –d /dev/loop0.