T12. Estudio de FUSE y NTFS-3G

Diseño y Estructura Interna de un Sistema Operativo

Cu...

10 de marzo de 2008

Índice

1. Introducción

Introducción

FUSE (Filesystems in Userspace)
Permite desarrollar y montar sistemas de ficheros que se ejecutan en espacio de usuario

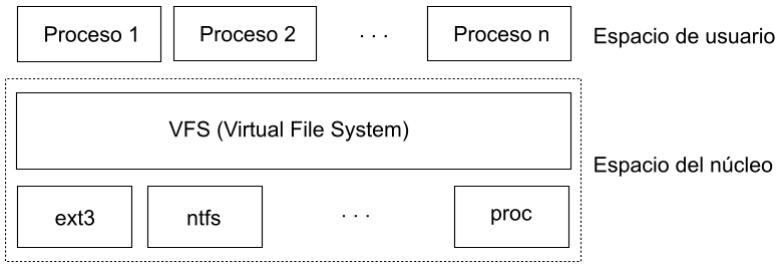
NTFS-3G
Driver para el sistema de ficheros NTFS para sistemas UNIX, implementado con FUSE. Permite lectura y escritura con seguridad

2. FUSE: Filesystems in Userspace

2.1. Introducción

Introducción

- En la mayoría de los sistemas operativos, la lógica de los sistemas de ficheros forma parte del núcleo, y suele implementarse como módulos del mismo



Introducción: ¿Qué es FUSE?

Sistemas de ficheros en espacio de usuario
Proporcionan los datos y meta datos mediante un proceso en espacio de usuario corriente

- El sistema de ficheros se accede utilizando la interfaz del kernel: necesario un módulo en el kernel realice la comunicación entre las aplicaciones y el proceso que implementa la lógica del sistema de ficheros

FUSE (Filesystems in Userspace)
Representa todo lo necesario para la implementación (*framework*) y manipulación de SSFF en espacio de usuario:

- Un módulo del kernel, comunicación entre sistema de ficheros y aplicaciones
- Una aplicación para realizar el montaje (*fusermount*)
- Una librería, *libfuse*, que permite la implementación de sistemas de ficheros FUSE

Introducción: ¿Qué es FUSE?

Características de FUSE

- Sistemas de ficheros ejecutándose en espacio de usuario
- Posibilidad de que usuarios sin privilegios pueda montar sistemas de ficheros:
  - Sin estar especificado en el fichero */etc/fstab*
  - Sin pérdida de seguridad para el sistema, aunque limitable como veremos
- Librería con API sencilla
- Instalación sencilla (no necesita parchear ni recompilar el kernel)
- La interfaz entre el espacio de usuario y el kernel es muy eficiente
- Funciona con los Kernels de Linux 2.4.X y 2.6.X
- Muy estable

Introducción: ¿Qué es FUSE?

Características de FUSE (continuación)

- Disponible para: Linux, FreeBSD, NetBSD, OpenSolaris y Mac OS X.
- Introducido en oficialmente en el kernel de Linux en la versión 2.6.14.

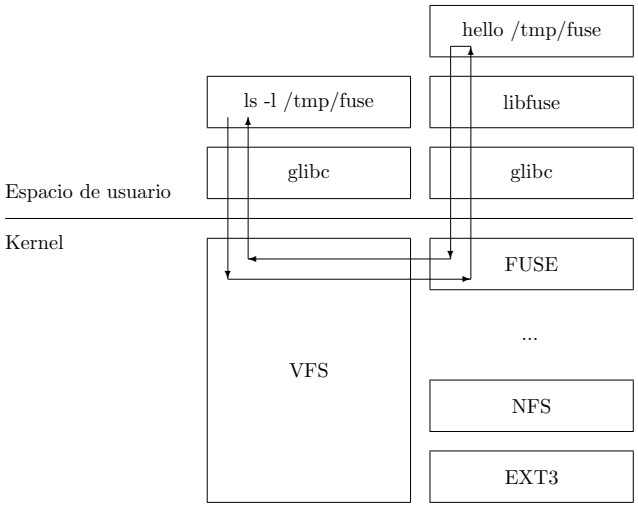
Introducción: ¿Para qué puede servir FUSE?

- Múltiples usos:
  - Sistemas de ficheros locales y remotos
  - Sistemas de ficheros que representen estado: al estilo *sysfs* o *proc*
  - Dar abstracción de sistema de ficheros: sensores

Ejemplo: *sshfs*
Posibilidad de acceder transparentemente a ficheros remotos almacenados en un servidor con acceso *ssh*, montándolo en el sistema de ficheros lógico, sin necesidad de ser el *root*

2.2. Funcionamiento

Funcionamiento



Funcionamiento

- El sistema de ficheros se ejecuta como demonio en espacio de usuario
- La comunicación entre el sistema de ficheros y el módulo se realiza mediante un descriptor de fichero especial que se obtiene abriendo el dispositivo */dev/fuse*
  - Abierta mientras el demonio está en ejecución o hasta que se desmonta
  - Este dispositivo puede ser abierto múltiples veces: múltiples montajes

Funcionamiento

Ejecución del SF

1. Obtiene el descriptor de fichero especial abriendo */dev/fuse*
2. Ejecuta *fusermount* para realizar el montaje
3. Se queda ejecutando un bucle que escucha peticiones desde el módulo del núcleo a través del descriptor de fichero

- *fusermount*:
  - Marcado con *suid* para que usuarios sin privilegios puedan realizar montajes, ya que la llamada al sistema *mount* es una llamada privilegiada
- Módulo del núcleo. Consta de 2 partes:
  1. La que responde a las peticiones al fichero de dispositivo*/dev/fuse*
  2. La que implementa las llamadas al sistema de ficheros que le remite el VFS

Funcionamiento

Tipos de sistemas de ficheros FUSE

- ***fuse*** Tipo más común. El núcleo ignora el primera argumento de la llamada *mount*. Ejemplo: *sshfs*
- ***fuseblk*** Sistemas de ficheros basados en dispositivos de bloques. El primer argumento de la llamada *mount* es interpretado por el nombre como el nombre del dispositivo (evitar múltiples montajes, ...). Ejemplo: NTFS-3G

```
asenac@asus:~$ cat /proc/filesystems
...
nodev   fuse
        fuseblk
...
asenac@asus:~$
```

2.3. Seguridad

Consideraciones de Seguridad

- Si usuarios sin privilegios tienen la capacidad de realizar montajes de sistemas de ficheros, FUSE debe asegurar que:
  - No pueda ser usada para poner en peligro el sistema
  - El autor del montaje no pueda ser capaz de obtener privilegios elevados, con la ayuda de un sistema de ficheros montado.
  - El autor del montaje no pueda tener acceso ilegítimo a información de los procesos de otros usuarios o del súper usuario.
  - El autor del montaje no pueda ser capaz de inducir un comportamiento indeseable de los procesos de otros usuarios o del súper usuario

Consideraciones de Seguridad

Ejemplo de uso mal intencionado

- Tenemos el siguiente programa:

```
int main(int argc, char * argv[])
{
        if(chmod("/sbin/shutdown", S_ISUID | 0777))
                printf("No se pudieron cambiar los permisos\n");
}
```

- Lo compilamos e introducimos en un sistema de ficheros FUSE, propiedad del usuario *root* y marcado con *suid*.
- Sin tener privilegios, lo montamos, y ejecutamos el programa anterior.

Consideraciones de Seguridad

Ejemplo de uso mal intencionado

```
asenac@toshiba:~/fuse/ejemplos$ make shutdown
gcc -Wall `pkg-config fuse --cflags --libs` shutdown.c -o shutdown
asenac@toshiba:~/fuse/ejemplos$ mkdir /tmp/fuse
asenac@toshiba:~/fuse/ejemplos$ ./shutdown /tmp/fuse/
asenac@toshiba:~/fuse/ejemplos$ ls -l /tmp/fuse/
total 0
-rwsrwxrwx 1 root root 6810 1970-01-01 01:00 hello
asenac@toshiba:~/fuse/ejemplos$ /tmp/fuse/hello
No se pudieron cambiar los permisos
asenac@toshiba:~/fuse/ejemplos$
```

- Como vemos, pese a estar marcado con *suid*, este no se hace efectivo al ejecutar el programa. Esta es una de las consideraciones de seguridad obvias que FUSE tiene en cuenta.

Consideraciones de Seguridad

Posibles usos mal intencionados a tener en cuenta

- El autor del montaje podría obtener elevados privilegios:
  1. creando un sistema de ficheros que contenga un archivo de dispositivo, y después abriendo el dispositivo,
  2. creando un sistema de ficheros que contenga una aplicación marcada con *suid* o *sgid*, y ejecutando esa aplicación.
- Si otro usuario está accediendo a archivos o directorios en el sistema de ficheros, el demonio puede grabar la secuencia exacta de las operaciones realizadas.
- Múltiples formas por las que el autor del montaje puede inducir un comportamiento indeseado en los procesos de otros usuarios.

Consideraciones de Seguridad

Soluciones adoptadas por FUSE

- Ficheros de dispositivos o con *suid* se ignoran si el autor del montaje es un usuario sin privilegios.
- No se puede montar sobre directorios sobre los que no se tengan privilegios.
- Un usuario no puede acceder a sistemas de ficheros montados por otro usuario. Esta medida se puede relajar con la opción de configuración *users_allow_other*.

Adicionalmente:

- Grupo *fuse*. Sólo usuarios del grupo pueden ejecutar *fusermount*. Ej: Ubuntu

2.4. Instalación, configuración y uso

Instalación

- Disponible como paquetes para algunas distribuciones.
  - Ej: En Ubuntu hay tres paquetes disponibles:
    - *fuse-utils*: Contiene la aplicación *fusermount*
    - *libfuse2*: Librería de FUSE
    - *libfuse-dev*: Cabeceras para el desarrollo de SSFF FUSE
    - Instalados por defecto a partir de la versión 7.10
- Bajando y compilando el código con: *configure*, *make* y *make install*

Configuración: */etc/fuse.conf*

- Actualmente, sólo dos opciones (una por línea):
  - **user_allow_other** Esta opción permite que los usuarios puedan utilizar la opción de montaje *allow_other*, que permite a cualquier usuario acceder al sistema de ficheros montado con esta opción. Recordemos que esta era una limitación de seguridad impuesta por defecto.
  - **mount_max=N** Limita el número máximo de montajes realizados mediante FUSE.

Uso

- Un sistema de ficheros FUSE es un proceso corriente $\Longrightarrow$ ejecutado como una aplicación normal:
  - Montar: ssff punto-de-montaje [-o opciones-de-montaje]
  - Desmontar: fusermount -u punto-de-montaje
- Opciones de montaje:

- **fd=N** Indica el descriptor de fichero empleado para la comunicación entre el proceso del sistema de ficheros en espacio de usuario y el kernel. Debe ser obtenido abriendo el dispositivo de FUSE (*/dev/fuse*).
- **rootmode=M** Permisos de la raíz del sistema de ficheros en octal.
- **user_id=N** El *uid* del propietario del montaje.
- **group_id=N** El *gid* del propietario del montajes.

Uso

- Opciones de montaje (continuación):
  - **default_permissions** Activa la comprobación de permisos según las restricciones de los archivos.
  - **allow_other** Esta opción relaja la medida de seguridad que restringe el acceso al sistema de ficheros montado por un usuario a otros usuarios. Esta opción, por defecto, solo puede utilizarla el *root*, pero puede eliminarse la restricción mediante la opción de configuración *user_allow_other*.
  - **max_read=N** Tamaño máximo de las operaciones de lectura.
  - **blksize=N** Tamaño de bloque del sistema de ficheros. Sólo con sistemas de ficheros de tipo *fuseblk*.

2.5. Sistemas de ficheros FUSE

Sistemas de ficheros FUSE

- **NTFS-3G** Controlador eficiente para NTFS que permite lectura y escritura. Existen otras implementaciones del sistema de ficheros NTFS basadas en FUSE.
- **EncFS** Sistema de ficheros encriptado en espacio de usuario.
- **OWFS (One Wire File System)** Permite ver sensores, botones y chips de memoria como un sistema de ficheros. Los dispositivos se incluyen en el directorio dinámicamente, y sus propiedades, como temperatura, se obtienen leyendo un archivo.
- **SshFs** Sistema de ficheros basado en el protocolo SSH File Transfer Protocol.

Sistemas de ficheros FUSE

- **FuseIso** Módulo para montar imágenes ISO9660. Permite montar imágenes de CD al estilo de las unidades virtuales de utilidades Windows.
- **FuseSmb** Sistema de ficheros cliente para el protocolo SMB.
- **FatFuse** Implementación del sistema de ficheros FAT en espacio de usuario. Actualmente solo permite lectura.
- **Gnome VFS2 Fuse** Permite montar en el sistema de ficheros virtual de Linux rutas del sistema de ficheros virtual de Gnome (Gnomo VFS2), haciendo accesibles los ficheros a cualquier aplicación, aunque no implemente un cliente de Gnome VFS. (Ej: SSH, FTP, SAMBA)

Sistemas de ficheros FUSE

- **FUSEPod** Sistema de ficheros virtual para el acceso a dispositivos IPod.
- **MysqlFs** Sistema de ficheros que almacena los archivos en bases de datos MySql.
- **GitFs** Proporciona acceso de lectura a repositorios GIT, mostrándolos como una estructura de ficheros y directorios.
- **CvsFs** Muestra el contenido de repositorios de código CVS como un sistema de ficheros.
- **XmlRpcFs** Permite la lectura y escritura de directorios remotos mediante XML-RPC.

2.6. Implementación de sistemas de ficheros

Implementación de sistemas de ficheros

- FUSE proporciona librería en C. Desarrollo principalmente en C
- También hay disponibles proyectos que permiten el desarrollo de sistemas de ficheros FUSE en *Ruby*, *Perl*, *Phyton*, *C#* o *Java*
  - libfusefs-ruby
  - libfuse-perl
  - python-fuse
  - FUSE-J
  - ...

Implementación de SSFF en C

- Un sistema de ficheros FUSE escrito en C consta básicamente de:
  - Una estructura de tipo *fuse_operations* que relaciona las distintas peticiones que se realizan sobre el sistema de ficheros con la función que implementa a cada una de ellas.
  - Una llamada al bucle de FUSE (*fuse_loop* ó *fuse_main*) que escucha las peticiones que envía el módulo del núcleo a través del descriptor de fichero especial.

Implementación de SSFF en C

```
struct fuse_operations {
        int (*getattr) (const char *, struct stat *);
        int (*readlink) (const char *, char *, size_t);
        int (*mknod) (const char *, mode_t, dev_t);
        int (*mkdir) (const char *, mode_t);
        int (*unlink) (const char *);
        int (*rmdir) (const char *);
        int (*symlink) (const char *, const char *);
        int (*rename) (const char *, const char *);
        int (*link) (const char *, const char *);
        int (*chmod) (const char *, mode_t);
        int (*chown) (const char *, uid_t, gid_t);
        int (*truncate) (const char *, off_t);
        int (*open) (const char *, struct fuse_file_info *);
        int (*read) (const char *, char *, size_t, off_t, struct fuse_file_info *);
        int (*write) (const char *, const char *, size_t, off_t,
                                        struct fuse_file_info *);
        int (*statfs) (const char *, struct statvfs *);
        int (*flush) (const char *, struct fuse_file_info *);
        int (*release) (const char *, struct fuse_file_info *);
        int (*fsync) (const char *, int, struct fuse_file_info *);
        int (*setxattr) (const char *, const char *, const char *, size_t, int);
        int (*getxattr) (const char *, const char *, char *, size_t);
```

```
        int (*listxattr) (const char *, char *, size_t);
        int (*removexattr) (const char *, const char *);
        int (*opendir) (const char *, struct fuse_file_info *);
        int (*readdir) (const char *, void *, fuse_fill_dir_t, off_t,
                        struct fuse_file_info *);
        int (*releasedir) (const char *, struct fuse_file_info *);
        int (*fsyncdir) (const char *, int, struct fuse_file_info *);
        void *(*init) (struct fuse_conn_info *conn);
        void (*destroy) (void *);
        int (*access) (const char *, int);
        int (*create) (const char *, mode_t, struct fuse_file_info *);
        int (*ftruncate) (const char *, off_t, struct fuse_file_info *);
        int (*fgetattr) (const char *, struct stat *, struct fuse_file_info *);
        int (*lock) (const char *, struct fuse_file_info *, int cmd,
                     struct flock *);
        int (*utimens) (const char *, const struct timespec tv[2]);
        int (*bmap) (const char *, size_t blocksize, uint64_t *idx);
};
```

Depurando un SSFF

- Montando el sistema de ficheros con la opción *-d*, la aplicación se ejecuta en primer plano mostrando por pantalla información sobre las peticiones realizadas al sistema de ficheros y su resultado:

```
asenac@asus:~/tmp/fusentfs3g/ejemplos$ ./hello /tmp/fuse/ -d
unique: 1, opcode: INIT (26), nodeid: 0, insize: 56
INIT: 7.8
flags=0x00000003
max_readahead=0x00020000
   INIT: 7.8
   flags=0x00000000
   max_readahead=0x00020000
   max_write=0x00020000
   unique: 1, error: 0 (Success), outsize: 40
unique: 2, opcode: GETATTR (3), nodeid: 1, insize: 40
   unique: 2, error: 0 (Success), outsize: 112
unique: 3, opcode: ACCESS (34), nodeid: 1, insize: 48
ACCESS / 04
   unique: 3, error: -38 (Function not implemented), outsize: 16
```

2.7. Alternativas

Alternativas a FUSE

Otros sistemas de ficheros en espacio de usuario

- **LUFS**: idea muy similar a FUSE
- **AVFS**: permite montar archivos empaquetas o comprimidos, y sistemas de ficheros remotos
- **Gnome VFS**: sistemas de ficheros remotos accesibles a aplicaciones de Gnome
- **UserFS**: procesos de usuario como sistemas de ficheros

2.8. Conclusiones

Conclusiones

- Sistemas de ficheros en espacio de usuario $\approx$ idea de micronúcleo:
  - Servicios se construyen sobre el micronúcleo y se ejecutan en espacio de usuario, siendo el micronúcleo el que implementa la comunicación entre cliente y servicio.

- No es necesario incrementar el tamaño del núcleo para añadir nuevos sistemas de ficheros.
- Seguridad es un requisito indispensable a tener en cuenta al permitir que usuarios sin privilegios puedan montar sistemas de ficheros con FUSE.
- FUSE reduce considerablemente la complejidad de la implementación de nuevos sistemas de ficheros. Programación a nivel de núcleo mucho más compleja.
- Rápido crecimiento $\Longrightarrow$ sistema de ficheros virtual

3. NTFS-3G

3.1. Introducción a NTFS

Introducción: NTFS

- Sistemas operativos Microsoft, a partir del NT
- Nombres largos en Unicode, ficheros largos, compresión transparente, cifrado, buen rendimiento
- Bloque lógico: direcciones de 8 bytes, tamaño entere 512 bytes y 64 Kbytes
- Fichero = conjunto de atributos. Al menos dos: nombre y flujo de datos principal o anónimo
- Varios flujos de datos por fichero, con nombre, accesibles individualmente mediante *nombre-fichero:nombre-flujo*
- MFT (Master File Table): estructura de datos principal, sucesión lineal de registros de 1 Kbyte

Master File Table

- Cada entrada representa un archivo. Si es muy grande o fragmentado múltiples registros en la tabla.
- Directorios pequeños = ficheros con lista desordenada de entradas. Directorios grandes = árboles B+

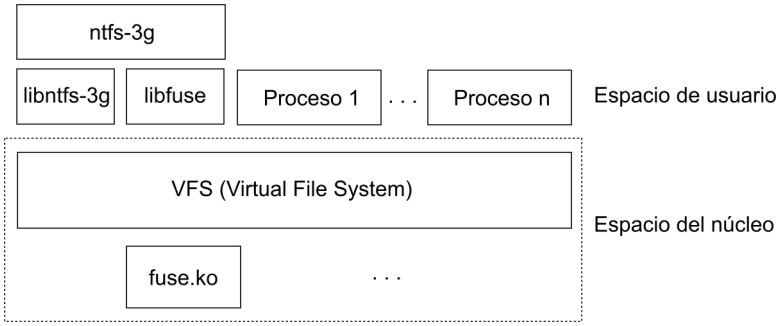
3.2. NTFS-3G

NTFS-3G

- Objetivo del proyecto: dar soporte fiable a diversos SSOO
- Driver para Linux, FreeBSD, NetBSD y Mac OS X, para NTFS
- Permite lectura y escritura con seguridad: No soporta ficheros encriptados, y sólo lectura en comprimidos
- Soporta la mayoría de operaciones POSIX, excepto listas de acceso
- Implementado mediante FUSE (tipo *fuseblk*) $\Longrightarrow$ SF en espacio de usuario

- No necesita módulo del kernel adicional, a parte del de FUSE
- Consta de:
  - Librería *libntfs-3g* con la lógica del sistema de ficheros NTFS
  - Aplicación *ntfs-3g* implementa el sistema de ficheros FUSE: estructura *fuse_operations* y llamada a *fuse_loop*

NTFS-3G



3.3. Instalación y Uso

Instalación

- Disponible como paquete para muchas distribuciones: alrededor de 100 distribuciones de Linux y FreeBSD que incluyen NTFS-3G como paquete binario para una fácil instalación
- Ubuntu 7.10 lo incluye como controlador por defecto para NTFS
- Para otras distribuciones o para actualización a la última versión descargable y compilable con *configure*, *make* y *sudo make install*.
- La instalación por defecto no permite montar a usuarios sin privilegios.

Uso

Montar

- Ejecutando: ntfs-3g /dev/sda1 /mnt/windows
- o bien: mount -t ntfs-3g /dev/sda1 /mnt/windows
- o añadiendo en */etc/fstab*: /dev/sda1 /mnt/windows ntfs-3g defaults 0 0

Desmontar

- Ejecutando: umount /mnt/windows
- o bien: fusermount -u /mnt/windows

Opciones (-o)

- Opciones típicas: *uid*, *gid*, *fmask*, *dmask*, *umask*, *ro*
- Opciones comunes de todos los SSFF FUSE.
- Otras opciones relevantes:
  - **locale**: idioma
  - **show_sys_files**: hace visibles los metadatos de NTFS
  - **streams_interface**: configura el acceso a los flujos de datos alternativos
  - **silent**: ignora operaciones *chown* y *chmod*

3.4. Calidad

Calidad

Fiabilidad es el principal objetivo del proyecto. Gran cantidad de pruebas, con las *ntfsprogs*, algunas de ellas:

- Creación de 13.000 archivos en un directorio y comprobación de la consistencia del directorio después de cada creación.
- Eliminación de 13.000 archivos de un directorio y comprobación de la consistencia del directorio después de cada eliminación.
- Eliminación de todos los archivos de una imagen real NTFS.
- Renombrado de todos los archivos de una imagen real NTFS.
- Creación, acceso, listado y eliminación de 1.000.000 de archivos de un único directorio.

Calidad

- Escritura concurrente en un archivo grande en muchas posiciones.
- Escritura de ficheros grandes en volúmenes muy fragmentados.
- Uso de archivos de intercambio (*swap*) y arranque de Linux sobre NTFS.
- Comprobación de la consistencia después de desmontar volúmenes después de una gran actividad.
- Operaciones aleatorios sobre ficheros en discos llenos.
- El *driver* y las utilidades son testeados a su vez con Valgrind.

3.5. Rendimiento

Rendimiento

- Aunque el principal objetivo sea la fiabilidad, el rendimiento es un requisito tenido en cuenta
  - A corto plazo, ofrecer rendimiento similar al resto de SSFF
  - A largo plazo, alcanzar límites teóricos de rendimiento
- NTFS-3G no optimizado aún

File system	Num Files	Random File Operation						File Size	Sequential						Random Seeks	
		Create		Lookup		Delete			Write		Rewrite		Read			
		/sec	%cpu	/sec	%cpu	/sec	%cpu		K/sec	%cpu	K/sec	%cpu	K/sec	%cpu	/sec	%cpu
ext3	16k	36316	78			65486	100	1G	36623	9	18805	6	41193	4	180.6	0
reiserfs	16k	20499	100			17113	99	1G	37768	14	19266	6	40730	6	190.8	0
reiser4	16k	17352	99			17401	99	1G	37537	5	19721	7	38991	5	197.6	0
ufs	16k	9670	100			14569	100	1G	42737	4	20203	5	42220	5	204.3	0
hfs+	16k	9557	99			14220	99	1G	42860	4	19782	5	43454	6	203.4	0
ext2	16k	3706	99			18838	99	1G	41511	4	19113	5	42623	5	192.5	0
ntfs-3g	16k	3629	4	14562	7	4057	3	1G	36483	6	16652	4	40792	2	130.6	0
jfs	16k	2851	16			954	4	1G	41003	7	20006	6	42490	5	193.9	0
xfs	16k	272	1			197	1	1G	40905	6	19905	6	42192	4	178.5	0
fat32	16k	85	99	113	99	200	99	1G	42324	9	20228	7	42102	4	187.1	0

3.6. Alternativas

Alternativas a NTFS-3G

ntfsmount

- Incluido en *ntfsprogs* (http://www.linux-ntfs.org)
- Más características que las incluidas en el módulo del kernel
- También implementado con FUSE
- Soporta sobre escritura de archivos, pero tiene limitaciones al crear y eliminar archivos y directorios
- Era el driver NTFS por defecto hasta la versión 7.04 de Ubuntu
- Proyecto del que se separó *NTFS-3G*

Captive NTFS

- También implementado con FUSE. Antes utilizaba LUFS
- Soporte completo y fiable de escritura
- Utiliza el driver original *ntfs.sys* de Windows XP
- Proyecto no mantenido

FUSE (Filesystems in Userspace)

http://fuse.sourceforge.net/

NTFS-3G

http://www.ntfs-3g.org/