




TP3 - Filesystem tipo FUSE

Integración de esqueleto



TP Filesystem - Introducción

- El esqueleto para este TP está en la rama *main* del repo [fisopfs](#)
- Para integrarlo deben:
 - desde su branch *entrega_sched* crear una nueva rama **base_filesystem**
 - mergear los cambios de *fisopfs*
 - pushear *base_filesystem* y crear *entrega_filesystem*

```
// Pararse en la rama entrega_sched
$ git checkout entrega_sched

// Crear una nueva rama base_filesystem (y pararse en ella)
$ git checkout -b base_filesystem

// Agregar el repositorio del esqueleto como remoto
$ git remote add fisopfs git@github.com:fisop/fisopfs.git

// Integración del esqueleto del tp sched
$ git fetch --all
$ git merge fisopfs/main --allow-unrelated-histories

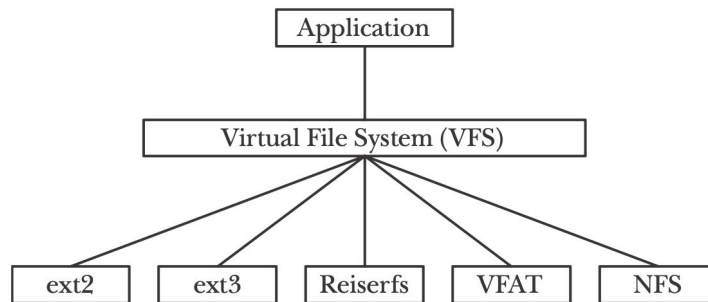
// Pushear la rama base_filesystem
$ git push origin base_filesystem

// Creación de la rama de entrega
$ git checkout -b entrega_filesystem
$ git push -u origin entrega_filesystem
```

Introducción a filesystem tipo FUSE

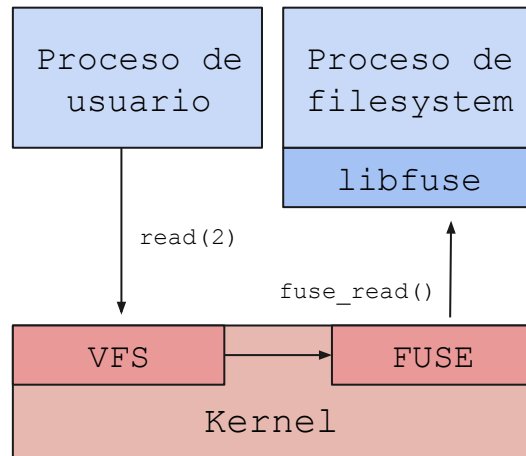
Virtual File System de Unix

- Una **interfaz** entre cualquier implementación de filesystem y userspace
 - Conjunto de syscalls: **read**, **write**, **open**, etc.
- El kernel puede incluir múltiples implementaciones de filesystem (incorporadas o como módulos) y elegir cual usar
- Incluso se puede **delegar al usuario** (FUSE)
 - El kernel intercepta las syscalls pero las “redirige” a un proceso de usuario



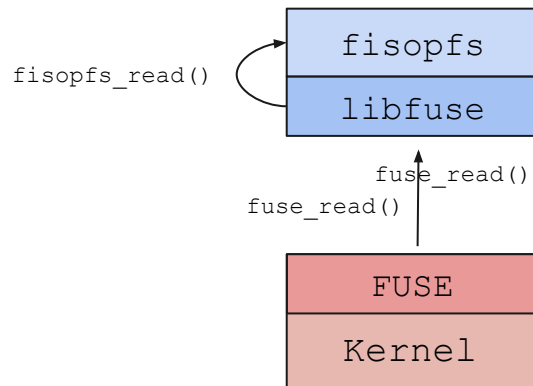
Filesystem in USErspace (FUSE)

- Para quien usa el filesystem la interfaz es VFS
- FUSE incluye:
 - Una librería de usuario
 - Un módulo del kernel
- El kernel redirecciona las **syscalls** a un proceso determinado, convirtiendolas en **operaciones** de FUSE
- Es el proceso que recibe las operaciones quien decide cómo manejarlas



libfuse

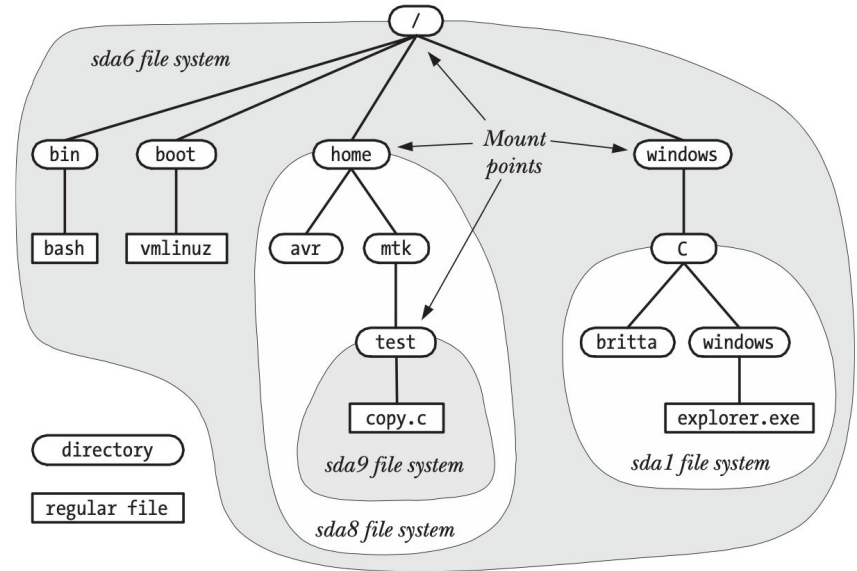
- **Importante:** Utilizaremos la librería libfuse v2
- Incluir el header **<fuse.h>**
 - Requiere de flags especiales de compilación! (ver pkg-config)
- Implementar un arreglo de **fuse_operations**
 - Arreglo de punteros a función
 - Prefijo en común para mantener el orden
- Entrada en **fuse_main**
- Documentación en el [repo de libfuse](#) (v2.9.9)



```
static struct fuse_operations operations = {  
    .getattr = fisopfs_getattr,  
    .readdir = fisopfs_readdir,  
    .read = fisopfs_read,  
};
```

mountpoints

- Directorio a partir del cual se hace accesible un **filesystem**
- Se pueden montar y desmontar filesystems usando las utilidades **mount** y **umount**
 - Conviene hacerlo sobre directorios vacíos
- Cuando una syscall interactúa con un directorio u archivo, el **kernel** traduce la syscall a la operación en la implementación del filesystem



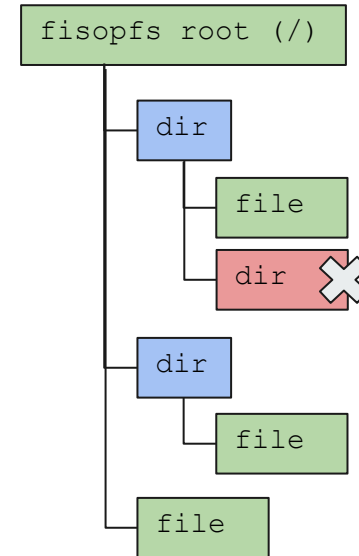
Requisitos de la implementación

Requisitos del filesystem

Es *requisito* que el sistema de archivos soporte las siguientes funcionalidades:

- Creación de archivos
- Creación de directorios
- Lectura de directorios, incluyendo los pseudo-directorios . y ..
- Lectura de archivos
- Escritura de archivos
- Acceder a las estadísticas de los archivos
 - Incluir y mantener fecha de modificación y creación
 - Asumir que todos los archivos son creados por el usuario y grupo actual
- Borrado de un archivo
- Borrado de un directorio

La creación de directorios debe soportar al menos **un nivel de recursión**, es decir, directorio raíz y sub-directorio





Recomendaciones

- Manejar los errores con constantes de **errno.h**
 - ENOENT, ENOTDIR, EIO, EINVAL, EBIG, ENOMEM, etc
- Utilizar estructuras estáticas para representar archivos y directorios
 - Se recomienda segmentar la memoria de alguna forma (e.g. usando bloques)
- Definir límites sensibles para:
 - Cantidad de archivos/directorios totales (i.e. similar a la cantidad de i-nodos o de archivos por directorio)
 - Tamaño máximo por archivo/directorio
 - Tamaño máximo del filesystem

Persistencia en disco

- Persistencia en disco serializando el filesystem entero en un único archivo
 - Se carga al momento de **montar**
 - Se guarda al momento de **desmontar**
 - Se guarda al momento de ejecutar un **flush**
- El resto del tiempo, el filesystem vive en memoria
- Se recomienda **primero** tener el filesystem funcionando de forma **efímera** en memoria; y **luego** encarar la **persistencia**

