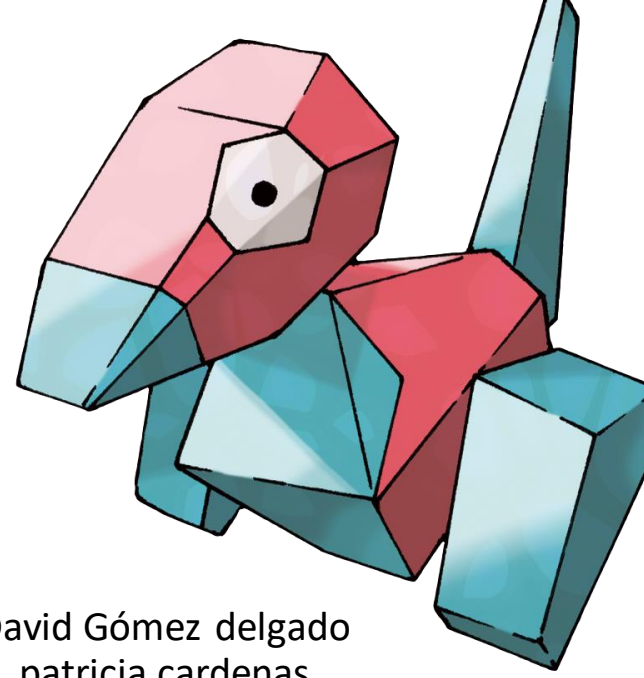
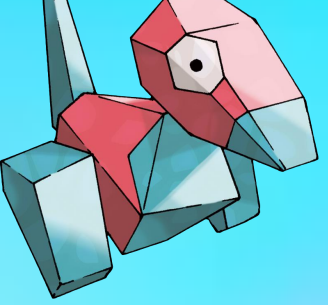
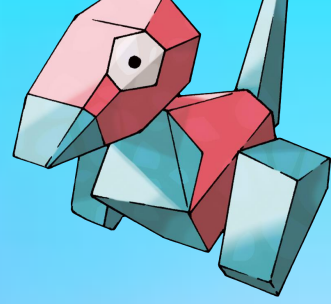


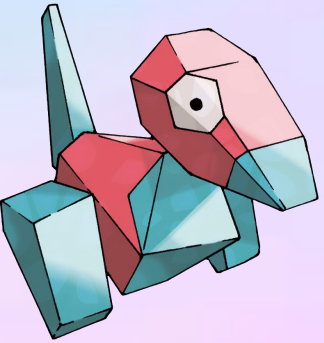
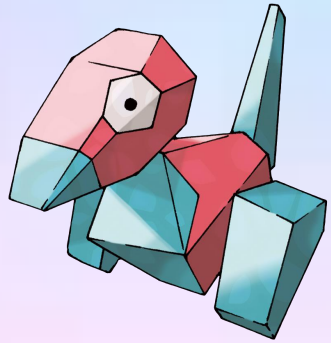
SISTEMA DE FICHEROS PORYGONEXT

David Gómez delgado
patricia cardenas Fernández
3ºComputadores
DSO UMA

UNIVERSIDAD DE MÁLAGA

Índice

Introducción

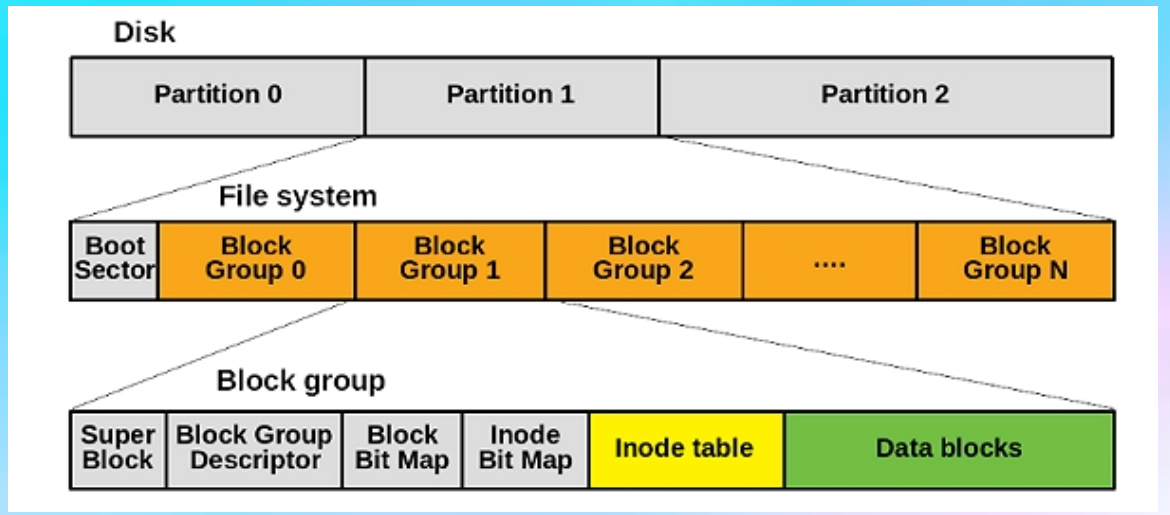
Sistema de ficheros basado en Indos básico

Característica	Ext2	Ext3	Ext4	PorygonExt
Tamaño del archivo individual	16GB-2TB	16GB-2TB	16GB-16TB	40KB
Tamaño del sistema de archivos del volumen	4TB-32TB	4TB-32TB	4TB-1EB	4KB * Número de bloques calculado
Tamaño del inodo por defecto	128 bytes	128 bytes	256 bytes	64bytes
Sello de tiempo	No hay apoyo	Segundo	Nanosegundo	No hay apoyo
Desfragmentación	No	No	Sí	No
Asignación de bloques múltiples	Básico	Básico	Avanzado	Muy Básico

Objetivos

Gestión de Bloques.
Gestión de Inodos.
Gestión de Ficheros.
Gestión de Directorios.
Estructurar la Memoria (Bitmaps)

FORMATEO DEL DISCO




```
typedef struct {
        struct superblock_fs *superblock;
        struct bitmap_t inode_bitmap;
        struct bitmap_t block_bitmap;
        struct inode_fs *inode;
        block_t *block;    //array incompleto que nos dice en cuál bloque de datos estamos
        struct timespec st_atim;                                    /* fechas del fichero */
        struct timespec st_mtim;
        struct timespec st_ctim;
        uid_t     st_uid;                                           /* El usuario y grupo */
        gid_t     st_gid;
} filesystem_t;
```

```c
struct superblock_fs {
        unsigned long magic_number;
        unsigned long bitmapb_offset;
        unsigned long bitmapi_offset;
        //unsigned long free_blocks;
        unsigned long inodes_ocupados;
    unsigned long num_blocks;
    unsigned long num_inodes;
    unsigned long offset_inodos;
    unsigned long offset_bloques;
    unsigned long block_size;
    unsigned long inode_size;
    unsigned long reserved_block;
};
```

```c
typedef unsigned char block_t[BLOCK_SIZE];
```

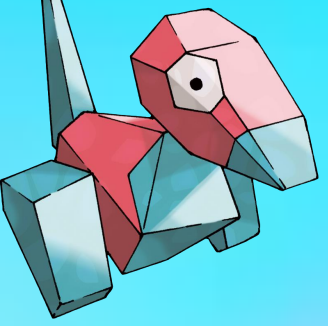
```c
struct bitmap_t{
        unsigned char *array;
        unsigned long long size;
};
```

```c
struct directory_entry{
        char name[28];
        long inode;
};
```

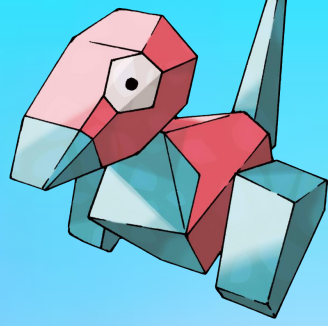
```c
struct inode_fs{
        long i_num;
        char i_type;
        int i_tam;
        int i_permisos;
        int i_links;
        int i_directos[N_DIRECTOS]; //1280 direcciones (128 bloques cada puntero)
        //offset en el fichero?
};
```

Información mmapeada

```
private_data->superblock = mmap(NULL, sizeof(struct superblock_fs), PROT_WRITE|PROT_READ, MAP_SHARED,file, 0);
private_data->block_bitmap.size = private_data->superblock->num_blocks/8;
private_data->block_bitmap.array = mmap(NULL, private_data->block_bitmap.size, PROT_WRITE|PROT_READ, MAP_SHARED,file, private_data->superblock->bitmapb_offset);
private_data->inode_bitmap.size = private_data->superblock->num_inodes/8;
private_data->inode_bitmap.array = mmap(NULL, private_data->inode_bitmap.size, PROT_WRITE|PROT_READ, MAP_SHARED,file, private_data->superblock->bitmapi_offset);

private_data->inode = mmap(NULL,private_data->superblock->bitmapi_offset - private_data->superblock->bitmapb_offset, PROT_WRITE|PROT_READ, MAP_SHARED,file, private_data->superblock->offset_inodos);
private_data->block = (block_t*) mmap(NULL,fileStat.st_size-private_data->superblock->offset_bloques, PROT_WRITE|PROT_READ, MAP_SHARED,file, private_data->superblock->offset_bloques);
//private_data->fd = file;

private_data -> st_uid = fileStat.st_uid;
private_data -> st_gid = fileStat.st_gid;

private_data -> st_atime = fileStat.st_atime;
private_data -> st_ctime = fileStat.st_ctime;
private_data -> st_mtime = fileStat.st_mtime;
```



Funciones implementadas



```c
//bitmap.c
unsigned long free_bit(struct bitmap_t *);

//create_inode.c
struct inode_fs *create_inode(char, filesystem_t *);
struct inode_fs *create_root(filesystem_t *);

// tree_manager.c
struct inode_fs *existe_inode(char *, struct directory_entry *);
struct inode_fs *inode_search (char *, struct inode_fs *, filesystem_t *);
struct inode_fs *inode_search_path(char *, filesystem_t *);

// directory_manager.c
int make_dir (char *, char *, filesystem_t *);
int rm_dir (char *, char *, filesystem_t *);

// file_manager.c
void update_entry (char *, struct inode_fs *, struct inode_fs *, filesystem_t *);
int touch (char *, char *, filesystem_t *);
void clean_inode (struct inode_fs *, filesystem_t *);
int rename_file(char *, char *, filesystem_t *);
```

```c
// file_remove.c
struct directory_entry *search_last_entry (struct inode_fs *, filesystem_t *);
void remove_dentry (char *, struct inode_fs *, filesystem_t *);
void remove_inode (struct inode_fs *, filesystem_t *);
int rm (char *, filesystem_t *);

// file_operations.c
size_t file_edit(const char *, char *, size_t, off_t,filesystem_t *);
char *read_file(char *, struct inode_fs*, filesystem_t *);

// directory_operations.c
void read_directory(char *, filesystem_t *);

// error.c
void error_parametros();
```

Formateo del fichero.

-Abrir el fichero que se pasa por parámetro.

-Calcular offsets.

-Mmapear la estructura filesystem_t *private_data

-Cerrar el fichero.

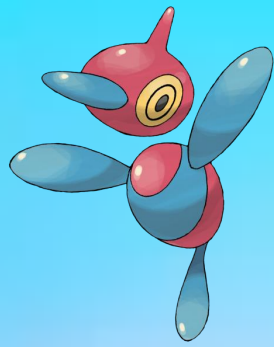
-Ocupar en el bitmap de bloques los bloques ocupados.

-Crear el Inodo raíz.

FUSE

```
static struct fuse_operations basic_oper = {
        .getattr        = fs_getattr,
        .readdir        = fs_readdir,
        .open           = fs_open,
        .read           = fs_read,
        .write          = fs_write,
        .rmdir          = fs_rmdir,
        .mkdir          = fs_mkdir,
        .create         = fs_create,
        .unlink         = fs_unlink,
        .rename         = fs_rename,
        .utimens        = fs_utimens, //no ha
        .truncate       = fs_truncate,

};
```

```
if(private_data->superblock->magic_number != MAGIC_N){
        printf("Fichero formateado erroneo\n");
        return -1;
}
```



Mejoras a futuro

-Corrección de posibles bugs.

-Implementación de Hard Links y Soft Links.

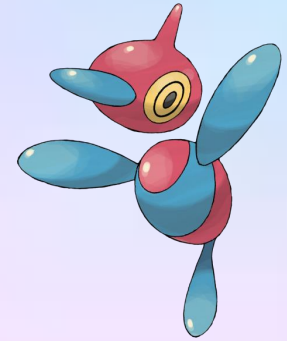
-Implementación de punteros indirectos.

-Actualizar la fecha de creación de cada fichero y directorio.

-Mejora en rendimiento.

-Desfragmentación.

-Encriptación del sistema de ficheros (mejorar la seguridad y conseguir un sistema de ficheros propio).



Conclusión

FIN