Kernel TP3 - Rapport système de fichiers - TexFS

Maxime Lovino Loic Willy

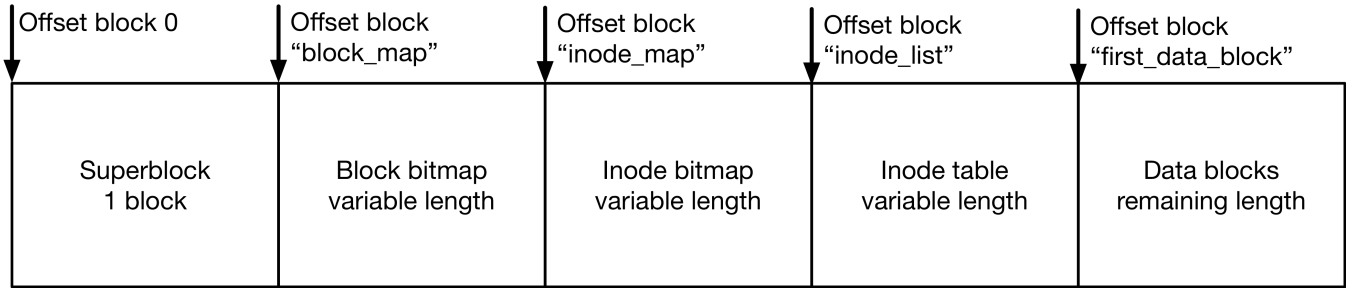
23 décembre 2017

1 Introduction

Pour réaliser le système de fichiers utilisé par notre Kernel, nous nous sommes inspirés du système de fichiers Ext2 sur lequel nous avons travaillé pendant un semestre en deuxième année. Nous avons adapté Ext2 par rapport aux besoins qui étaient spécifiés pour ce projet. C'est-à-dire que nous avons principalement simplifié Ext2 en supprimant les doubles et triples indirections, qui ne sont pas nécessaires compte tenu de la taille des fichier que nous devons stocker. Nous n'avons également pas tenu compte de la notion de groupes.

De ce fait, lors de la création d'une image TexFS, il est nécessaire de spécifier la taille de bloc, le nombre de bloc à allouer et le nombre de fichiers maximum pour l'image. La taille de bloc et le nombre de blocs devaient être spécifiés de toute façon, mais compte tenu de la structure utilisée, il est nécessaire d'entrer également le nombre de fichiers maximum que contiendra l'image. (Si nous avions des groupes, nous aurions pu allouer un nouveau groupe en cas de besoins de nouveaux fichiers)

2 Structure du système de fichiers



2.1 Blocs de métadonnées

2.1.1 Superblock

```
typedef struct tex_fs_superblock_st {
    uint16_t magic;
    uint8_t version;
    char label[MAX_LABEL_LENGTH];
    uint16_t block_size;
    uint32_t block_map;
    uint32_t block_count;
    uint32_t inode_bitmap;
    uint32_t inode_list;
    uint32_t inode_count;
    uint32_t first_data_block;
} __attribute__((packed)) tex_fs_superblock_t;
```

2.1.2 Block bitmap

2.1.3 Inode bitmap

2.1.4 Inode Table

```
typedef struct tex_fs_inode_st {
    char name[MAX_FILENAME_LENGTH];
    uint32_t size;
    uint32_t direct_blocks[DIRECT_BLOCKS];
    uint32_t indirect_blocks[INDIRECT_BLOCKS];
} __attribute__((packed)) tex_fs_inode_t;
```

3 Exemples

4 Implémentation

5 Avantages du système choisi

6 Inconvénients du système choisi