DCC605 File System Checker (DCC-FSSHELL).

1. Pode ser feito em dupla
2. Domingo Depois da Terceira Prova

Neste trabalho você vai implementar um shell para o sistema ext2. Para realizar seu TP recomendo um bom entendimento do Fast File System. O mesmo é a base do ext2. Além disto, a seção 42.2 do OSTEP deve ser útil, leia a mesma aqui.

Sobre o ext2

O Extended File System 2 (ext2) é um sistema de arquivos criados para sistemas Linux em meados de 1993. Visando corrigir alguns problemas da primeira versão do Extended File System (simplesmente ext), o sistema de arquivos ext2 tem um esquema de implementação mais próximo ao Berkley Fast File System (FFS).

Embora já tenha caído um pouco em desuso, o ext2 é um sistema de arquivos com bastante influência. O desenvolvimento do ext2 teve como um dos principais objetivos a extensibilidade do sistema, sendo assim o mesmo serviu como base para o ext3 e ext4 que são mais populares hoje em dia.

Layout dos inodes, grupos e blocos

Seguindo o modelo do FFS, um disco formatado com um sistema de arquivos ext2 terá um layout de blocos similar ao da figura abaixo (note que existem problemas de escala na mesma, é apenas um esquema):

```
Layout geral:
  * Bloco de Boot         --> Utilizado para iniciar o sistema, sempre ocupa
                              uma posição fixa no ínicio do disco.
  * Grupo de Blocos i     --> Cada grupo de blocos é utilizado para guardar
                              arquivos. Fazemos uso de mais de um grupo pois
                              discos tem vários cílindros. Então guardar
                              artigos relacionados em um mesmo bloco ajuda.

 1-bloco
+-------+-------+-------+-------+-------+-------+-------+-------+-------+-----
| bloco |                               |                               |
| de    |       grupo de blocos 0       |       grupo de blocos 1       | ...
| boot  |                               |                               |
+-------+-------+-------+-------+-------+-------+-------+-------+-------+-----
       /                                 \
      /                                   \
     /                                     \
    /                                       \
   /                                         \
  /                                           \
 /                                             \
/              grupo de blocos i                \
+-------+-------+-------+-------+-------+-------+-------+-------+-------+------
| super |  descritores  |       |       | tabela        |
| bloco |      do       |d-bmap |i-bmap | de inode      |  blocos de dados...
| const |     grupo     |       |       | inodes        |
+-------+-------+-------+-------+-------+-------+-------+-------+-------+------
 1-bloco    n-blocos     1-bloco 1-bloco     n-blocos          n-blocos

Layout de um grupo:
  * Super Bloco           --> Contém meta-dados do sistema de arquivos. Uma
                              cópia em cada grupo.
  * Descritores do Grupo  --> Meta-dados do grupo.
  * Data Bitmap (d-bmap)  --> Mapa de bits de dados livres
  * Inode Bitmap (i-bmap) --> Mapa de bits de inodes livres
  * Tabela de Inodes      --> Contém os inodes (metadados) do sistema de
                              arquivos. Cada arquivo tem 1 apenas 1 inode.
                              Através de links, um mesmo inode pode aparecer
                              mapear para 2 caminhos.
  * Bloco de Dados        --> Os dados do arquivos e diretórios em si.
```

Structs úteis

Para entender melhor o ext2, vamos dar uma olhada no cabeçalho do Linux que descreve o sistema de arquivos. O mesmo pode ser encontrado aqui.

Antes de iniciar, temos que entender os tipos `__le32` e `__le16`. Como o Linux é cross-platform, tipos genéricos para qualquer arquitetura são necessários. Esses dois em particular são *unsigned ints* de 32 e 16 bits. Os mesmos sempre vão ser representados em *little endian*.

Como o PC que vai corrigir o TP é little endian (o seu tamnbém deve ser), pode usar um atalho como:

```
typedef int __le32;
```

Super bloco

```
struct ext2_super_block {
  __le32 s_inodes_count;          /* Inodes count */
  __le32 s_blocks_count;          /* Blocks count */
  __le32 s_r_blocks_count;        /* Reserved blocks count */
  __le32 s_free_blocks_count;     /* Free blocks count */
  __le32 s_free_inodes_count;     /* Free inodes count */
  __le32 s_first_data_block;      /* First Data Block */
  __le32 s_log_block_size;        /* Block size */
  // . . .
  __le32 s_blocks_per_group;      /* # Blocks per group */
  // . . .
  __le32 s_inodes_per_group;      /* # Inodes per group */
  // . . .
  __le16 s_magic;                 /* Magic signature */
  __le32 s_first_ino;             /* First non-reserved inode */
  __le16 s_inode_size;            /* size of inode structure */
  // . . .
}
```

Descritores de Grupo

```
struct ext2_group_desc
{
  __le32 bg_block_bitmap;         /* Blocks bitmap block */
  __le32 bg_inode_bitmap;         /* Inodes bitmap block */
  __le32 bg_inode_table;          /* Inodes table block */
  __le16 bg_free_blocks_count;    /* Free blocks count */
  __le16 bg_free_inodes_count;    /* Free inodes count */
  __le16 bg_used_dirs_count;      /* Directories count */
  __le16 bg_pad;
  __le32 bg_reserved[3];
};
```

Para saber o número de grupos no ext2 usamos a seguinte abordagem. A mesma faz uso dos campos do superbloco.

```
/* calculate number of block groups on the disk */
unsigned int group_count = 1 + (super.s_blocks_count-1) / super.s_blocks_per_group;

/* calculate size of the group descriptor list in bytes */
unsigned int descr_list_size = group_count * sizeof(struct ext2_group_descr);
```

Para ler os descritores do grupo, primeiramente você deve calcular o offset do inicio do disco. Como o disco tem 1024 bytes reservados no inicio e o primeiro bloco é um superbloco, o código é para ler o descrito é tal como:

```
struct ext2_group_descr group_descr;
/* position head above the group descriptor block */
/* sd --> storage device, no nosso caso um arquivo */
lseek(sd, 1024 + block_size, SEEK_SET);
read(sd, &group_descr, sizeof(group_descr));
```

O descritor do grupo vai conter meta-dados para identificar o data e inode bitmap daquele grupo. Uma macro boa de se ter indica qual o local do disco de um dado bloco:

```
/* location of the super-block in the first group */
#define BASE_OFFSET 1024
#define BLOCK_OFFSET(block) (BASE_OFFSET + (block-1)*block_size)
```

INodes

```
/*
 * Structure of an inode on the disk
 */
struct ext2_inode {
  __le16 i_mode;                /* File mode */
  __le16 i_uid;                 /* Low 16 bits of Owner Uid */
  __le32 i_size;                /* Size in bytes */
  __le32 i_atime;               /* Access time */
  __le32 i_ctime;               /* Creation time */
  __le32 i_mtime;               /* Modification time */
  __le32 i_dtime;               /* Deletion Time */
  __le16 i_gid;                 /* Low 16 bits of Group Id */
  __le16 i_links_count;         /* Links count */
  __le32 i_blocks;              /* Blocks count */
  __le32 i_flags;               /* File flags */
  __le32 i_block[EXT2_N_BLOCKS]; /* Pointers to blocks */
  // . . .
};
```

Diretórios

```
struct ext2_dir_entry {
  __le32  inode;         /* Inode number */
  __le16  rec_len;       /* Directory entry length */
  __le16  name_len;      /* Name length */
  char    name[];        /* File name, up to EXT2_NAME_LEN */
};
```

Criando imagens

O script de teste já cria as imagens que você deve trabalhar em cima. O mesmo faz uso dos comandos `dd` e `mkfs` discutidos em sala. Segue alguns exemplos:

Comando dd - criando imagem zerada

```
$ filename=fs-0x00dcc605-ext2-10240.img
$ dd if=/dev/zero of=$filename bs=1024 count=10240
```

```
1024+0 records in
1024+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.0242941 s, 43.2 MB/s
```

Comando mkfs.ext2 - criando um disco de 1mb sem superblock backup

```
$ mkfs.ext2 fs-0x00dcc605-ext2-10240.img
```

```
mke2fs 1.42.13 (17-May-2015)
Creating filesystem with 10240 1k blocks and 2560 inodes
Filesystem UUID: 24c464b5-2e6c-4b6f-8309-d3454d683858
Superblock backups stored on blocks:
        8193

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
```

Comandos que você deve realizar

Todos os comandos abaixo devem ser re-implementados utilizando os inodes. Isto é, não utilize os programas do Unix.

1. **cd** Caminhar para um diretório. Inicie o programa na raiz.
2. **ls** Seu shell deve executar um ls e listar todos os arquivos do diretório atual.
3. **stat** Pega os metadados de um arquivo/diretório.
4. **find** Imprime toda a árvore de pastas/arquivos iniciando do diretório atual.

5. **sb** Lê os dados do super-bloco.

Entrega

Um .c e um .h (caso precise) que roda o shell corrigindo. Chame seu programa de `dcc_fs_shell`.

A entrega será pelo moodle. Desta vez como é um único arquivo faz menos sentido um repositório no git. Porém, caso deseje utilizar, pode fazer a entrega pelo git.

Rodando seu programa

```
$ ./dcc_fs_shell uma_imagem.img
```