Mini Teste

Sistemas de Operação

Ano Lectivo de 2016/2017

–/11/2016

NºMec: _____ **Nome:** _____

Na resolução deste exame, tenha em consideração o seguinte:

- **As questões são independentes entre si.** Assim, a resposta a qualquer questão deve considerar o estado do disco tal como apresentado e não aquele que resultaria da execução do excerto de código apresentado numa outra qualquer questão.
- **As questões devem ser respondidas no contexto concreto do disco apresentado.**
- Pode responder às questões pela ordem que quiser, muito embora se responder primeiro à primeira questão ganhará uma compreensão do sistema de ficheiros que o ajudará na resposta às restantes.
- As questões 1, 7 e as três mais bem cotadas das restantes têm cotação de 3 valores cada; as outras têm cotação de 2,5 valores cada.
- A duração do exame é de 75+15mn.
- À saída, **deve entregar** tudo o que recebeu.

Considere que se criou um disco virtual sobre um ficheiro *linux*. Esse disco foi formatado usando o programa `mksofs` e montado no diretório `/tmp/mnt/`, usando o programa `sofsmount`. Diversas operações de manipulação de ficheiros (ficheiros regulares, diretórios e atalhos) foram a seguir efetuadas sobre esse diretório (ponto de montagem).

As listagens das páginas 5 a 8 representam o estado interno de alguns blocos/clusters do disco após as operações anteriores, mostrados usando a ferramenta `showblock`. Alguns campos do superbloco e do nó-i número 1 da tabela de nós-i foram intencionalmente ocultados /* /*. A tabela de nós-i é apenas parcialmente mostrada; todos os nós-i não mostrados estão livres e limpos e não foram alterados desde a formatação. Os campos `atime`, `mtime` e `ctime` nunca são mostrados. Para facilitar a leitura, nos campos `name` das entradas de directório o carácter `'\0'` foi substituído por um espaço.

---

1. Complete o preenchimento da tabela seguinte com a informação referente a todos os ficheiros residentes no disco.

caminho absoluto (***path***)	**nº do nó-i** (***nInode***)	**tipo** (**dir/file/symlink**)	**refcount**
/	0		

2. Nos dados apresentados abaixo sobre o estado das estruturas de dados internas do sistema de ficheiros, alguns campos foram intencionalmente substituídos por `???`.

   (a) Apresente os valores dos seguintes campos do superbloco.

   `ntotal`: ..... `itsize`: .....

   `ctotal`: ..... `cfree`: .....

   (b) Apresente o valor do seguinte campo do nó-i (*inode*) número 1.

   `inode[1].next`: .....

   ⋮
   ⋮

---

3. Considere que o excerto de código seguinte é executado, não tendo sido gerada nenhuma exceção.

```
uint32_t n;
soAllocInode(S_IFREG, &n);
```

   (a) Apresente os valores, após a execução do excerto de código, dos campos do superbloco indicados abaixo, assim como o valor da variável `n`.

   `itotal`: ..... `ifree`: .....

   `ihead`: ..... `itail`: .....

   `n`: .....

   (b) Há outros nós-i (*inodes*), além daquele cujo número foi armazenado na variável `n`, alterados em consequência da execução? Se sim, quais os nós-i alterados e que alterações sofreram? Se não, indique porquê?

   ⋮
   ⋮

---

4. Considere que o excerto de código seguinte é executado, não tendo sido gerada nenhuma exceção.

```
int ih = iOpen(3);
soFreeFileClusters(ih, 10);
soFreeFileClusters(ih, 3);
```

   (a) Apresente os valores, após a execução do excerto de código, dos campos do superbloco seguintes, sendo que `ref[*]` representa todo o array útil. Pode usar notação compactada, quando aplicável. Se não respondeu à questão 2, considere que antes da execução `cfree = 1000`.

   `cfree`: ..... `tbfreeclust_head`: ..... `tbfreeclust_tail`: .....

   chead.cache: `in`: ..... `out`: ..... `ref[*]`: .....

   ctail.cache: `in`: ..... `out`: ..... `ref[*]`: .....

(b) Apresente os valores dos seguintes campos do nó-i número 3, sendo que `d[*]` e `i1[*]` representam os arrays na totalidade.

`size`: .... `csize`: ....

`d[*]`: ....

`i1[*]`: .... `i2`: ....

(c) Há um cluster da zona de dados alterado em consequência da execução do excerto de código anterior. Qual? Que alterações sofre?

⋮

⋮

⋮

---

5. Considere que o excerto de código seguinte é executado, não tendo sido gerada nenhuma exceção.

```
uint32_t n1, n2;
int ih0 = iOpen(0);
soGetDirEntry(ih0, "aaaa", &n1);
int ih1 = iOpen(n1);
soDeleteDirEntry(ih1, "gggg", &n2);
```

(a) Que valores são armazenados nas variáveis `n1` e `n2`?

`n1`: .... `n2`: ....

(b) Apresente os valores, após a execução do excerto de código, dos campos do superbloco seguintes. Se não respondeu à questão 2, considere que antes da execução `cfree = 1000`.

`ifree`: .... `cfree`: ....

(c) Dos 3 clusters de dados com entradas de diretório apresentados neste exame, qual ou quais sofrem alterações em consequência da execução do excerto de código? Que alterações sofrem?

⋮

⋮

⋮

⋮

⋮

---

6. Considere que o excerto de código seguinte é executado, e que após a sua execução `ret` tem o valor 0.

```
#define PERM 0755
int ret = soMkdir("/zzzz", PERM);
```

(a) Que campos dos nós-i em uso ou que ficaram em uso após a execução sofrem alterações em consequência da execução do excerto de código? Que alterações sofrem?

(b) Que clusters da zona de dados sofrem alterações em consequência da execução do excerto de código? Que alterações sofrem?

---

7. O comando `rm -rf dir`, sendo `dir` um directório, apaga-o, incluindo todos os ficheiros, atalhos e outros diretórios nele contidos. Considere que, sendo o diretório corrente o ponto de montagem do disco, executa o comando seguinte

```
rm -rf "dddd"
```

À frente de cada uma das seguintes chamadas de sistema do sofs16 coloque um `sim` ou um `não` para respetivamente indicar se ela é invocada no contexto de execução do comando anterior. Se não sabe, deixe em branco, porque uma resposta errada desconta um valor igual ao que conta uma resposta correta.

`soLink`:	`soUnlink`:	`soMknod`: .....
`soRename`:	`soRead`:	`soWrite`: .....
`soTruncate`:	`soReaddir`:	`soMkdir`: .....
`soRmdir`:	`soSymlink`:	`soReadlink`: .....

Estado da estrutura de dados interna do disco

Block 0 as superblock data

```
   Magic number: 0x50F5
   Version number: 0x2016
   Volume name: mini-teste-modelo
   Total number of blocks in the device: ???
   Properly unmounted: yes
Inode table metadata:
   Total number of inodes: 24
   First block of the inode table: 1
   Number of blocks of the inode table: ???
   Number of free inodes: 16
   Head of list of free inodes: 10
   Tail of list of free inodes: 1
Data zone:
   Number of blocks per cluster: 2
   First block of the cluster zone: 4
   Total number of clusters: ???
   Number of free clusters: ???
   Number of clusters used by list of free clusters: 1
   FCT head cache of references to free data clusters:
      Index of the first filled cache element: 26
      Index of the first free cache element: 0
      Cache contents:
         (nil) (nil) (nil) (nil) (nil) (nil) (nil) (nil) (nil) (nil)
         (nil) (nil) (nil) (nil) (nil) (nil) (nil) (nil) (nil) (nil)
         (nil) (nil) (nil) (nil) (nil) (nil) 28 29 30 31
         32 33 34 35 36 37 38 39 40 41
         42 43 44 45 46 47 48 49 50 51

   FCT tail cache of references to free data clusters:
      Index of the first free cache element: 6
      Index of the first filled cache element: 0
      Cache contents:
         19 20 6 27 7 8 (nil) (nil) (nil) (nil)
         (nil) (nil) (nil) (nil) (nil) (nil) (nil) (nil) (nil) (nil)
         (nil) (nil) (nil) (nil) (nil) (nil) (nil) (nil) (nil) (nil)
         (nil) (nil) (nil) (nil) (nil) (nil) (nil) (nil) (nil) (nil)
         (nil) (nil) (nil) (nil) (nil) (nil) (nil) (nil) (nil) (nil)

   FCT head cluster of references to free clusters:
     Cluster number: 1
     Index of first filled entry: 50
   FCT tail cluster of references to free clusters:
     Cluster number: 1
     Index of first empty entry: 216
```

Block 1 as inode entries

```
Inode #0
type =  directory, permissions = rwxrwxr-x, refcount = 4, owner = 1000, group = 1000
size in bytes = 320, size in clusters = 1
d[] = {0 (nil) (nil) (nil) (nil)}, i1[] = {(nil) (nil)}, i2 = (nil)
----------------
Inode #1
type = free regular file, permissions = rw-rw-r--, refcount = 0, owner = 1000, group = 1000
size in bytes = 2000, size in clusters = 0
next = ???
d[] = {(nil) (nil) (nil) (nil) (nil)}, i1[] = {(nil) (nil)}, i2 = (nil)
----------------
Inode #2
type =  directory, permissions = rwxrwxr-x, refcount = 2, owner = 1000, group = 1000
size in bytes = 320, size in clusters = 1
d[] = {2 (nil) (nil) (nil) (nil)}, i1[] = {(nil) (nil)}, i2 = (nil)
----------------
Inode #3
type =  regular file, permissions = rw-rw-r--, refcount = 1, owner = 1000, group = 1000
```

```
size in bytes = 22000, size in clusters = 9
d[] = {13 (nil) (nil) (nil) 14}, i1[] = {15 (nil)}, i2 = (nil)
----------------
Inode #4
type =  regular file, permissions = rw-rw-r--, refcount = 1, owner = 1000, group = 1000
size in bytes = 5000, size in clusters = 5
d[] = {3 4 5 25 26}, i1[] = {(nil) (nil)}, i2 = (nil)
----------------
Inode #5
type =  directory, permissions = rwxrwxr-x, refcount = 2, owner = 1000, group = 1000
size in bytes = 256, size in clusters = 1
d[] = {9 (nil) (nil) (nil) (nil)}, i1[] = {(nil) (nil)}, i2 = (nil)
----------------
Inode #6
type =  regular file, permissions = rw-rw-r--, refcount = 2, owner = 1000, group = 1000
size in bytes = 2000, size in clusters = 2
d[] = {10 11 (nil) (nil) (nil)}, i1[] = {(nil) (nil)}, i2 = (nil)
----------------
Inode #7
type =  regular file, permissions = rw-rw-r--, refcount = 1, owner = 1000, group = 1000
size in bytes = 13, size in clusters = 1
d[] = {21 (nil) (nil) (nil) (nil)}, i1[] = {(nil) (nil)}, i2 = (nil)
----------------
```

Block 2 as inode entries

```
Inode #0
type =  symlink, permissions = rwxrwxrwx, refcount = 1, owner = 1000, group = 1000
size in bytes = 12, size in clusters = 1
d[] = {12 (nil) (nil) (nil) (nil)}, i1[] = {(nil) (nil)}, i2 = (nil)
----------------
Inode #1
type = free regular file, permissions = rw-rw-r--, refcount = 0, owner = 1000, group = 1000
size in bytes = 4, size in clusters = 0
next = 1
d[] = {(nil) (nil) (nil) (nil) (nil)}, i1[] = {(nil) (nil)}, i2 = (nil)
----------------
Inode #2
type = free clean, permissions = ----------, refcount = 0, owner = 0, group = 0
size in bytes = 0, size in clusters = 0
next = 11
d[] = {(nil) (nil) (nil) (nil) (nil)}, i1[] = {(nil) (nil)}, i2 = (nil)
----------------
    ...
----------------
Inode #7
type = free clean, permissions = ----------, refcount = 0, owner = 0, group = 0
size in bytes = 0, size in clusters = 0
next = 16
d[] = {(nil) (nil) (nil) (nil) (nil)}, i1[] = {(nil) (nil)}, i2 = (nil)
----------------
```

Block 3 as inode entries

```
Inode #0
type = free clean, permissions = ----------, refcount = 0, owner = 0, group = 0
size in bytes = 0, size in clusters = 0
next = 17
d[] = {(nil) (nil) (nil) (nil) (nil)}, i1[] = {(nil) (nil)}, i2 = (nil)
----------------
    ...
----------------
Inode #6
type = free clean, permissions = ----------, refcount = 0, owner = 0, group = 0
size in bytes = 0, size in clusters = 0
```

```
next = 23
d[] = {(nil) (nil) (nil) (nil) (nil)}, i1[] = {(nil) (nil)}, i2 = (nil)
----------------
Inode #7
type = free clean, permissions = ---------, refcount = 0, owner = 0, group = 0
size in bytes = 0, size in clusters = 0
next = 9
d[] = {(nil) (nil) (nil) (nil) (nil)}, i1[] = {(nil) (nil)}, i2 = (nil)
----------------
```

Blocks 6-7 as references

```
0008:   (nil)      (nil)      (nil)      (nil)      (nil)      (nil)      (nil)      (nil)
0016:   (nil)      (nil)      (nil)      (nil)      (nil)      (nil)      (nil)      (nil)
0024:   (nil)      (nil)      (nil)      (nil)      (nil)      (nil)      (nil)      (nil)
0032:   (nil)      (nil)      (nil)      (nil)      (nil)      (nil)      (nil)      (nil)
0040:   (nil)      (nil)      (nil)      (nil)      (nil)      (nil)      (nil)      (nil)
0048:   (nil)      (nil)    0000000052 0000000053 0000000054 0000000055 0000000056 0000000057
0056: 0000000058 0000000059 0000000060 0000000061 0000000062 0000000063 0000000064 0000000065
0064: 0000000066 0000000067 0000000068 0000000069 0000000070 0000000071 0000000072 0000000073
0072: 0000000074 0000000075 0000000076 0000000077 0000000078 0000000079 0000000080 0000000081
0080: 0000000082 0000000083 0000000084 0000000085 0000000086 0000000087 0000000088 0000000089
0088: 0000000090 0000000091 0000000092 0000000093 0000000094 0000000095 0000000096 0000000097
0096: 0000000098 0000000099 0000000100 0000000101 0000000102 0000000103 0000000104 0000000105
0104: 0000000106 0000000107 0000000108 0000000109 0000000110 0000000111 0000000112 0000000113
0112: 0000000114 0000000115 0000000116 0000000117 0000000118 0000000119 0000000120 0000000121
0120: 0000000122 0000000123 0000000124 0000000125 0000000126 0000000127 0000000128 0000000129
0128: 0000000130 0000000131 0000000132 0000000133 0000000134 0000000135 0000000136 0000000137
0136: 0000000138 0000000139 0000000140 0000000141 0000000142 0000000143 0000000144 0000000145
0144: 0000000146 0000000147 0000000148 0000000149 0000000150 0000000151 0000000152 0000000153
0152: 0000000154 0000000155 0000000156 0000000157 0000000158 0000000159 0000000160 0000000161
0160: 0000000162 0000000163 0000000164 0000000165 0000000166 0000000167 0000000168 0000000169
0168: 0000000170 0000000171 0000000172 0000000173 0000000174 0000000175 0000000176 0000000177
0176: 0000000178 0000000179 0000000180 0000000181 0000000182 0000000183 0000000184 0000000185
0184: 0000000186 0000000187 0000000188 0000000189 0000000190 0000000191 0000000192 0000000193
0192: 0000000194 0000000195 0000000196 0000000197 0000000198 0000000199 0000000200 0000000201
0200: 0000000202 0000000203 0000000204 0000000205 0000000206 0000000207 0000000208 0000000209
0208: 0000000210 0000000211 0000000212 0000000213 0000000214 0000000215 0000000216 0000000217
0216:   (nil)      (nil)      (nil)      (nil)      (nil)      (nil)      (nil)      (nil)
0224:   (nil)      (nil)      (nil)      (nil)      (nil)      (nil)      (nil)      (nil)
0232:   (nil)      (nil)      (nil)      (nil)      (nil)      (nil)      (nil)      (nil)
0240:   (nil)      (nil)      (nil)      (nil)      (nil)      (nil)      (nil)      (nil)
0248:   (nil)      (nil)      (nil)      (nil)      (nil)      (nil)      (nil)      (nil)
```

Blocks 34-35 as references

```
0000: 0000000016 0000000017 0000000018   (nil)      (nil)      (nil)      (nil)      (nil)
0008:   (nil)      (nil)      (nil)      (nil)      (nil)      (nil)    0000000022 0000000023
0016: 0000000024   (nil)      (nil)      (nil)      (nil)      (nil)      (nil)      (nil)
0024:   (nil)      (nil)      (nil)      (nil)      (nil)      (nil)      (nil)      (nil)
0032:   (nil)      (nil)      (nil)      (nil)      (nil)      (nil)      (nil)      (nil)
          ...
0048:   (nil)      (nil)      (nil)      (nil)      (nil)      (nil)      (nil)      (nil)
```

Blocks 4-5 as direntries

```
.                                                  0000000000
..                                                 0000000000
bbbb                                               0000000006
dddd                                               0000000002
aaaa                                               0000000005
                                                   (nil)
          ...
                                                   (nil)
```

Blocks 8-9 as direntries

.	0000000002
..	0000000000
gggg	0000000003
ffff	0000000004
aaaa	0000000008
	(nil)
...	
	(nil)

Blocks 22-23 as direntries

.	0000000005
..	0000000000
gggg	0000000007
ffff	0000000006
	(nil)
...	
	(nil)

Blocks 28-29 as text

```
0000: .  .  /  a  a  a  a  /  g  g  g  g  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0032: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0064: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0096: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
          ...
```

Blocks 46-47 as text

```
0000: .  .  /  d  d  d  d  /  f  f  f  f  \n 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0032: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0064: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0096: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
          ...
```

Assinatura das funções referenciadas neste exame

```
int iOpen(uint32_t in);
void soAllocInode(uint32_t type, uint32_t* p_nInode);
void soFreeFileClusters(int ih, uint32_t ffcn);
void soGetDirEntry(int pih, const char *name, uint32_t * cinp);
void soDeleteDirEntry(int pih, const char *name, uint32_t * cinp);
int soMkdir(const char *path, mode_t mode);
```

Declaração das estruturas de dados internas do SOFS15

```
#define NULL_REFERENCE 0xFFFFFFFF   /* represents the absence of a reference */
```

```
#define FCT_CACHE_SIZE  50 /* size of cache */

struct FCTRecord /* data structure representing a cache and a reference cluster */
{
    struct
    {
        uint32_t ref[FCT_CACHE_SIZE];   /* the cache */
        uint32_t in;                    /* insertion point */
        uint32_t out;                   /* retrieval point */
    } cache;
    uint32_t cluster_number; /* number of chead/ctail cluster */
    uint32_t cluster_idx;    /* index of first filled/free position in previous cluster */
};

struct SOSuperBlock
{
    uint16_t magic;    /* file system identification number */
    uint16_t version;  /* version number */
    char name[PARTITION_NAME_SIZE + 1]; /* volume name */
    uint8_t mstat;     /* mount status */

    uint8_t csize;     /* number of blocks per cluster */
    uint32_t ntotal;   /* total number of blocks in the device */

    uint32_t itstart;  /* physical number of the block where the table of inodes starts */
    uint32_t itsize;   /* number of blocks that the table of inodes comprises */
    uint32_t itotal;   /* total number of inodes */
    uint32_t ifree;    /* number of free inodes */
    uint32_t ihead;    /* head of linked list of free inodes */
    uint32_t itail;    /* tail of linked list of free inodes */

    uint32_t czstart;  /* number of the block where the cluster zone starts */
    uint32_t ctotal;   /* total number of data clusters */
    uint32_t cfree;    /* number of free clusters */
    uint32_t crefs;    /* number of clusters used by the list of free clusters */
    FCTRecord chead;   /* head cache and head cluster of references */
    FCTRecord ctail;   /* tail cache and tail cluster of references */
};
```

```
#define DPB     (BLOCK_SIZE / sizeof(SODirEntry)) /* number of direntries per block */

#define SOFS16_MAX_NAME 59 /* maximum length of a file name (in characters) */

/** \brief Definition of the directory entry data type. */
struct SODirEntry
{
    char name[SOFS16_MAX_NAME + 1];  /* the name of a file (NULL-terminated string) */
    uint32_t in;                     /* the associated inode number */
};
```

```
#define IPB (BLOCK_SIZE / sizeof(SOInode)) /* number of inodes per block */

#define INODE_FREE (0001000) /* flag signaling inode is free (it uses the sticky bit) */

#define N_DIRECT 5 /* number of direct references in the inode */

#define N_INDIRECT 2 /* number of indirect references in the inode */

/** \brief Definition of the inode data type. */
struct SOInode
{
    uint16_t mode;      /* file type and permissions */
    uint16_t refcount;  /* number of hard links (directory entries) associated to the inode */
    uint32_t owner;     /* user ID of the file owner */
    uint32_t group;     /* group ID of the file owner */
    uint32_t size;      /* file size in bytes: */
    uint32_t csize;     /* cluster count: total number of clusters used by the file */

    union /* usage depends on state */
    {
        uint32_t atime;      /* time of last access to file information (if inode is in use) */
        uint32_t next;       /* next free inode (if inode is free) */\
    };
    uint32_t ctime;          /* time of last change to inode information */
    uint32_t mtime;          /* time of last change to file information */

    uint32_t d[N_DIRECT];    /* direct references */
    uint32_t i1[N_INDIRECT]; /* references to clusters that extend the d array */
    uint32_t i2;             /* reference to a cluster that extends the i1 array */
};
```

```
#define RPB (BLOCK_SIZE / sizeof (uint32_t))     /* number of references per block */
```
