http://w3.ualg.pt/~hdaniel/pin/avaliacao/tp2.pdf

2º Trabalho Prático PIn 2005/06
Módulo do kernel para leitura de disco IDE

1 Descrição

Desenvolver um módulo do kernel, segundo a metodologia dos drivers para dispositivos de caracter, que permita ler informação do disco rígido master ligado ao controlador IDE primário, de acordo com as especificações seguintes:

1.1

O módulo deverá chamar-se **dsk**, de modo que deverá ser identificado em **/proc/devices** por:

```
Block devices
(...)
<Major>     dsk
```

1.2

Terá um parâmetro de linha de comando que permita que o utilizador indique o número Major do dispositivo:

```
insmod dsk.ko [Major = ??]
```

No caso do parâmetro ser omitido o Major deverá ser automaticamente atribuído.
No caso do Major especificado pelo utilizador não estar livre o módulo não deverá ser inserido, sendo enviada uma mensagem para o registo do sistema com **printk( )**.

1.3

Quando o módulo for inserido ou removido deverá também ser registada a operação com **printk ()**.

1.4

Deverá ser escrito um script, chamado **dsk**, que receba como primeiro parâmetro **start** ou **stop**, e que respectivamente insira ou remova o módulo no kernel. Como segundo parâmetro (opcional), este script poderá receber o Major do módulo especificado pelo utilizador:

```
dsk  <start | stop>  [Major = ??]
```

Que deverá ser passado para o módulo no momento da inserção.

1.5

Os serviços disponibilizados pelo módulo deverão ser:

a) **open** e **release**, de modo que actualizem um contador com o número de utilizadores correntes.

b) Leitura de sectores. Assim com o comando:

```
hexedit –s /dsk/
```

deve ser possível visualizar os sectores.

A escrita de sectores, no entanto, deve ser impedida. Se se tentar escrever, p.ex. com **hexedit,** ou com **cat /dev/dsk**, deve ser escrita a mensagem no registo do **kernel:**

```
DSK device is Read only
```

c) **ioctl** que permita saber o número de utilizadores correntes, a versão ATA/ATAPI do dispositivo, o número de sectores total do disco (em modo LBA) e o tamnho em bytes de cada bloco ou sector. O ficheiro header a utilizar, com nome **"dskioctl.h"** deverá ser:

```
/* IOCTLs para dsk
*/

#ifndef DSKIOCTL
#define DSKIOCTL

#include <linux/ioctl.h>

#define IOCID 0x17

// return no. users
#define HDIO_NUSERS       _IO   (IOCID, 0)

// return ATA/ATAPI version
#define HDIO_ATAVER       _IO   (IOCID, 1)

// return no. LBA sectors in drive
#define HDIO_LBASEC   _IO   (IOCID, 2)

// return sectors (block) size
#define HDIO_SECSIZE  _IO   (IOCID, 3)

#endif
```

Sugestão: Identificar o disco IDE0 na inicialização do módulo, guardando os valores necessários para a versão ATA/ATAPI e nº de sectores no disco em variáveis, que serão retornadas pelos comandos de IOCTL. Assim evita-se efectuar a identificação do disco sempre que é necessário executar um comando IOCTLm como HDIO_ATAVER e HDIO_LBASEC.

1.6
O módulo não deverá suportar remoção de disco.

1.7
Deverá ser desenvolvido também um programa em C, que possa ser executado por qualquer utilizador, e que aceda ao módulo. O programa deverá chamar-se **dskm.c** e suportar os seguintes parâmetros na linha de comandos:

```
users       Retorna o número de utilizadores correntes do módulo dsk:

                  Users: 1

sectors     Retorna o número de sectores total do disco IDE0

                  LBA maximum sectors: 32905939205

secsize     Retorna o número de bytes em cada sector

                  Sector (block) size: 512
```

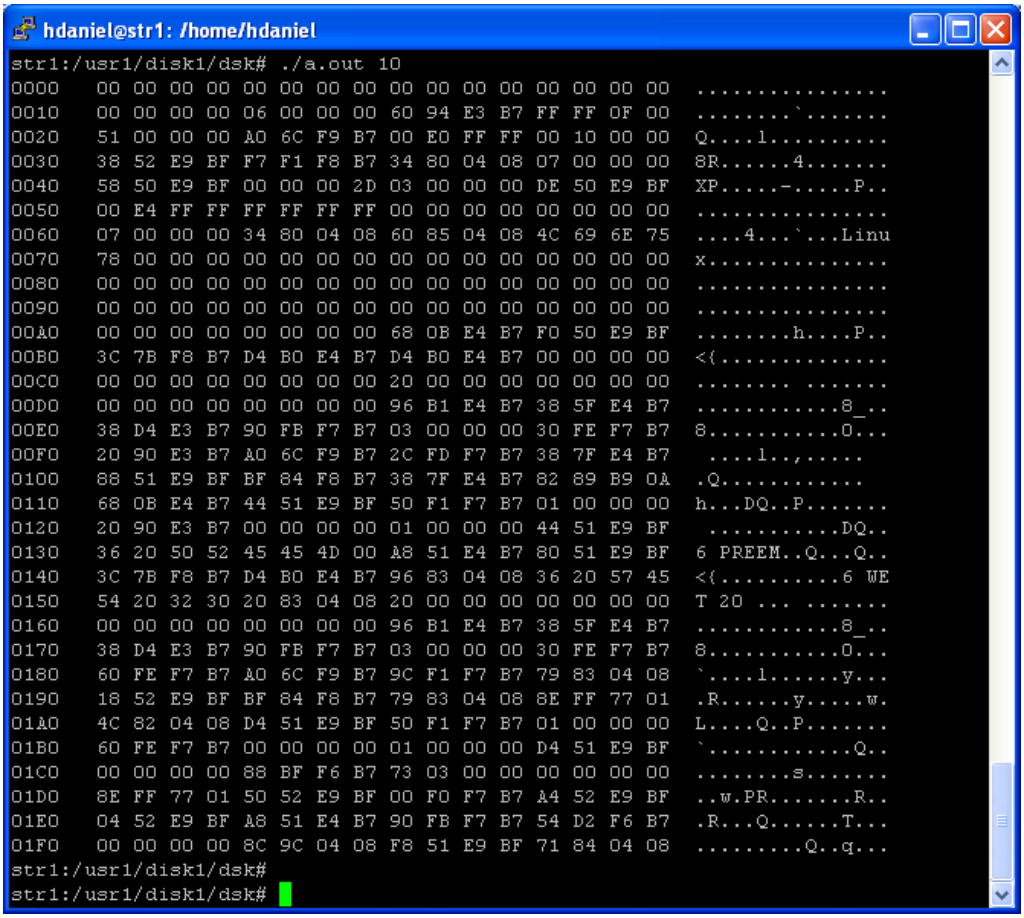
`version` Retorna a versão ATA/ATAPI

ATA/ATAPI version: 6

`geo` Retorna a geometria CHS lógica do disco IDE0, onde se assumem 255 cabeças e 63 sectores por cabeça

C: 2048 H: 255 S: 63

`<nº sector>` Apresenta o sector passado como argumento na forma:



Isto é apresenta o valor de cada byte em hexdecimal e em ASCII. Se o código ASCII for menro que 32 ou superior a 127 mostra apenas um ponto.

Todas estas operações deverão ser efectuadas usando IOCTL, de acordo com os comandos especificados em 1.5 c).
Se o número de parâmetros não for o correcto, ou não for conhecido deverá ser impressa a mensagem:

```
Usar com:
dskm [users | sectors | secsize | version | geo | <LBA sec. no. to show ]
```

http://w3.ualg.pt/~hdaniel/pin/avaliacao/tp2.pdf

2 Entrega do trabalho

2.1 Data

A data limite de entrega é terça-feira 13 de Junho de 2006. A entrega para além do prazo sofre uma penalização de 0.5 valores por dia.

2.2 Código fonte

O ficheiro com o código fonte do módulo deverá ter o nome **dsk.c** e deve ser acompanhado de um *makefile* para gerar o módulo com o nome **dsk.ko**. O módulo deverá ser compilado de modo que com o comando **modinfo** obtenha-se informação pelo menos sobre a licença, a descrição dos parâmetros (indicando o valor por defeito), uma descrição breve do módulo, a versão, e os Nomes, N$^{os.}$ e curso dos autores:

```
modinfo dsk.ko

filename:       dsk.ko
license:        GPL
author:         Helder Daniel
version:        1.0
description:    Leitor do disco rígido IDE0
vermagic:       2.6.14-ipipe preempt 386 gcc-3.3
depends:
srcversion:     B865AF6B744DFDB7321CE85
```

De qualquer forma, todos os ficheiros devem ter como comentário os Nomes, N$^{os.}$ e curso dos elementos do grupo.
Deve ser entregue também o script **dsk** que insere e remove o módulo do kernel, referido em 1.4.

Pode ser entregue por e-mail ou em disquete ao docente do turno prático em que está inscrito o grupo.

2.3 Relatório

Com o código fonte terá de ser entregue um relatório, que deverá incluir um pequeno manual do utilizador e um manual de implementação. Um *template* para este relatório está disponível na página da disciplina:

http://w3.ualg.pt/~hdaniel/pin/avaliacao/relat_template.pdf

No manual do utilizador deverá ser descrito tanto o módulo **dsk.ko** como a aplicação **dskm**, referida no ponto 1.7, incluindo os parâmetros da linha de comandos de ambos.

O relatório deverá ser entregue em papel, ao docente do turno prático em que está inscrito o grupo.

3 Avaliação

De acordo com o indicado na aula teórica 1. Será tomada especial atenção à correcção do programa, à eficiência da implementação e à estrutura e clareza do código.

Nota: Funcionalidades não pedidas nos enunciados não serão avaliadas.

http://w3.ualg.pt/~hdaniel/pin/avaliacao/tp2.pdf

4 Bibliografia recomendada

Corbet, Jonathan, Alessandro Rubini e Kroah-Hartman, Greg (2005). *"Linux Device Drivers 3rd edition"*, O'Reilly, http://lwn.net/Kernel/LDD3/

Salzman, Peter Jay, Michael Burian and Ori Pomerantz (2005). *"The Linux Kernel Module Programming Guide"*, *http://www.tldp.org/LDP/lkmpg/2.6/lkmpg.pdf*

Identificação de dispositivos IDE: *http://www.t13.org/technical/e00159r2.pdf*

Aulas teóricas e práticas disponíveis em http://w3.ualg.pt/~hdaniel/Ed/, especialmente:

Capítulo 5 e 7 das aulas teóricas.
Ficha prática 4.2 e 6.

Nota:
podem ser utilizados ou adaptados funções e scripts desenvolvidas nas aulas teóricas, práticas ou dadas como código auxiliar para as fichas práticas, em particular poderá ser usado o código auxiliar da RAMDSK como base de desenvolvimento do módulo, mas todas as funções e timers que simulão a remoção do disco terão de ser removidas, e o nome das função deverá começar por DSK e não RAMDSK.