UNIVERSIDADE DO SAGRADO CORAÇÃO

LUCAS ALVARES GOMES

PROTÓTIPO DE CLONAGEM DE DISCO

BAURU

2010

UNIVERSIDADE DO SAGRADO CORAÇÃO

PROTÓTIPO DE CLONAGEM DE DISCO

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Sociais Aplicadas como parte dos requisitos para obtenção do Título de Bacharel em Ciência da Computação, sob orientação do Prof. Esp. André Luiz Ferraz Castro.

BAURU

2010

LUCAS ALVARES GOMES

PROTÓTIPO DE CLONAGEM DE DISCO

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Sociais Aplicadas como parte dos requisitos para obtenção do Título de Bacharel em Ciência da Computação, sob orientação do Prof. Esp. André Luiz Ferraz Castro.

Banca examinadora:

_____
Esp. André Luiz Ferraz de Castro
Universidade do Sagrado Coração

_____
Ms. Larissa Pavarini da Luz
Universidade do Sagrado Coração

_____
Prof. Esp. Henrique Pachioni Martins
Universidade do Sagrado Coração

Bauru, 26 de dezembro de 2010

“42”. (Douglas Adams)

RESUMO

Diversos mecanismos voltados para backup e restauração de dados têm sido propostos para assegurar a integridade dos dados de computadores, visto que a perda de dados importantes pode ter profundas implicações sobre o futuro de uma empresa.
Um método muito popular para realizar cópias de segurança do sistema operacional, softwares instalados, arquivos de configuração e dados de uma organização com o mínimo de esforço é a clonagem de disco, uma vez que esta pode fazer a cópia de um disco rígido inteiro e replicá-lo para um ou mais discos.
Essa abordagem também pode ser aplicada em diversos cenários, tais como provisionamento de novos computadores em ambiente de trabalho, atualizações de disco rígido, por fabricantes de computador que desejam instalar em seus produtos um sistema operacional pré-configurado, com ferramentas e determinadas aplicações.
Contudo, este trabalho de pesquisa é apenas um protótipo de um sistema de clonagem de discos direcionado apenas para alguns sistemas de arquivos.
No modelo proposto serão avaliadas características como integridade dos dados após o processo de restauração e compactação da imagem resultante.

Palavras – Chave: Clonagem de disco. Backup. Integridade dos dados. Sistema de arquivos.

ABSTRACT

Several mechanisms aimed at backing up and restoring data have been proposed to ensure the integrity of computer data, since the loss of important data can have profound implications on the future of a company.
A very popular method for backing up the operating system, installed software, configuration files and data from an organization with minimal effort is disk cloning, since it can make a copy of an entire hard disk and replicate it to one or more disks.
This approach can also be applied in various scenarios, such as provisioning of new desktop computers, hard disks upgrades, for computer manufacturers wishing to install in their products an operating system pre-configured with certain tools and applications.
However, this research is only a prototype system for disk cloning focused only on certain filesystems.
In the proposed model will be evaluated characteristics such as data integrity after the restore process and compression of the resulting image.

Key words: Disk cloning. Backup. Data integrity. Filesystem.

Lista de Figuras

Figura 1 – Valores de cada símbolo em número um decimal.....11
Figura 2 – Conversão de binário para decimal.....12
Figura 3 – Conversão de hexadecimal para decimal 12
Figura 4 - (a) Dados no registrador. (b) Memória big endian. (c) Memória little endian. 14
Figura 5 – Tabela ASCII.....15
Figura 6 – Estrutura da MBR (Master Boot Record).....16
Figura 7 – Disco rígido e seus componentes.....17
Figura 8 – Placa controladora.....18
Figura 9 – Motor de rotação 19
Figura 10 – Uma foto mostrando o quão fino os pratos são 19
Figura 11 – Braço com uma cabeça presa em cada ponta.....20
Figura 12 – Atuador 21
Figura 13 – Trilhas, setores e cilindros 21
Figura 14 – Superfície de um prato dividido por trilhas e setores.....22
Figura 15 – Uma aplicação pede para ler um arquivo do disco rígido, o sistema operacional pede para o sistema de arquivo encontrá-lo.....23
Figura 16 – Drive de disco com uma Partição Primária 24
Figura 17 – Drive de disco com Partição Estendida. 25
Figura 18 – O aplicativo fdisk listando os tipos de partição. 28
Figura 19 – Aplicação exibindo o menu de ajuda.....31
Figura 20 – Aplicação clonando um disco 32
Figura 21 – Exemplo de um XML gerado pela aplicação.....33
Figura 22 – Diagrama de criação da imagem.....33
Figura 23 – Diagrama de restauração da imagem 35
Figura 24 – Bootloader com erro após aplicação da imagem 36

Lista de Tabelas

Tabela 1 – Tabela de conversão entre decimal, binário e hexadecimal 13
Tabela 2 – Estrutura da MBR 26
Tabela 3 – O padrão de 64 bytes da tabela de partições da MBR 26
Tabela 4 – Estrutura de 16 bytes de uma entrada de partição 27

Sumário

1 INTRODUÇÃO 7
1.1 MOTIVAÇÃO 9
1.2 OBJETIVOS 10
1.2.1 Objetivo Geral 10
1.2.2 Objetivos Específicos 10
2 ORGANIZAÇÃO DOS DADOS 11
2.1 SISTEMAS DE NUMERAÇÃO: DECIMAL, BINÁRIO E HEXADECIMAL 11
2.2 ENDEREÇAMENTO E ORDENAÇÃO DE BYTES 13
2.3 CODIFICAÇÃO DE CARACTERES 14
2.4 ESTRUTURA DE ORGANIZAÇÃO DE DADOS 16
3 O DISCO RÍGIDO: COMPONENTES E GEOMETRIA 17
3.1 PRINCIPAIS COMPONENTES 17
3.1.1 Placa controladora 17
3.1.2 Motor 18
3.1.3 Pratos 19
3.1.4 Braço e Cabeça 20
3.1.5 Atuador 20
3.2 GEOMETRIA DE DISCO 21
4 SISTEMAS DE ARQUIVOS 23
4.1 PARTICIONAMENTO 24
4.1.1 Partição Primária 24
4.1.2 Partição Estendida 24
4.2 MBR 25
4.2.1 Área do código 26
4.2.2 Tabela de partições 26
4.2.3 Assinatura da MBR 28
5 METODOLOGIA 29
6 RESULTADOS OBTIDOS 37
7 CONCLUSÃO 38
8 REFERÊNCIAS BIBLIOGRÁFICAS 39

1 INTRODUÇÃO

Hoje em dia os computadores são uma grande parte da vida das pessoas e empresas, eles são usados para as mais diversas tarefas como: compras, trabalho, educação e entretenimento. Eles estão substituindo diários, álbuns de fotos, televisão, enciclopédias e até mesmo o carteiro.

O ensino auxiliado pela internet tem o potencial de melhorar o ensino e a aprendizagem mais do que qualquer outra inovação recente (PRITCHARD, 2007).

Com a evolução dos dispositivos de armazenamento de dados tem-se a capacidade de armazenar milhares de *gigabytes* de dados, o que a algumas décadas átras era uma tarefa impossível. Esses dados são fundamentais na vida de pessoas e empresas, a retirada de informações a partir da análise desses dados tem feito com que as empresas tenham maior eficácia e rapidez nas tomadas decisões, devido a este fato, muitas empresas tem como uma das principais responsabilidades assegurar a integridade dos dados de computadores, para que se possa garantir a sobrevivência e competitividade da organização.

No entanto, garantir a segurança desses dados não é uma tarefa simples, todos estão sujeitos a cortes de energia, inundações, incêndios, roubos, ataque de vírus e danos acidentais, portanto, não existe segurança absoluta, os efeitos podem ser desastrosos e a recuperação, quando possível, muito cara.

Para Tyagi (2004):

> No mundo da computação atual, onde os dados de uma organização são a parte mais importante na maioria dos casos, e qualquer tipo de desastre nos dados pode causar uma grande perda financeira e de negócios, e a organização pode arruinar a sua reputação também.

Complementando Patterson (2010):

> A maioria das empresas dependem de informações de missão crítica que pode ter um impacto sobre o sucesso de suas operação. Como consequência, essas empresas fazem uso de medidas especiais para proteger essa informação vital. Sistemas de *Backup* ajudam essas empresas a prevenir o risco de perder o controle de seus dados.

De acordo com o autor acima, medidas de segurança e ferramentas de *backup* ajudam a amenizar o risco de perda de dados. Simples tarefas como manter o sistema operacional atualizado e o uso de *firewalls* para gerenciar a vulnerabilidade do sistema são

exemplos de medidas simples que podem ser aplicadas, quanto aos sistemas de *backup*, hoje em dia existem diversas ferramentas disponíveis no mercado que usam diferentes técnicas para preencher as necessidades de *backup*.

Dentre as técnicas, existe uma chamada clonagem de disco, que é uma maneira avançada de se copiar dados, onde a maior vantagem é a criação de uma imagem de um disco rígido inteiro, contendo o sistema operacional, softwares instalados, configurações e os outros dados (MIKE D. , 2008)

Nessa técnica as informações contidas em um disco rígido é escrito para um arquivo de imagem, que age como um armazenamento intermediário na operação de clonagem e, em seguida, diversos outros discos podem ser carregado a partir da imagem gerada, nesse contexto, uma possível aplicação de uma ferramenta de clonagem de disco como sistema de *backup* seria, substituir um disco rígido defeituoso por outro previamente gravado que contenha os mesmos dados, já configurado e pronto para o uso (GARVRISH, 2008), (MIKE D. , 2008).

Além de sistemas de *backup*s, uma ferramenta de clonagem de discos pode ter diversas outras aplicações, muitas empresas utilizam essas ferramentas quando precisam providenciar novos computadores para os membros do seu pessoal, fabricantes de computadores utilizam aplicando uma imagem contendo uma instalação padrão nas máquinas antes de saírem da fabrica, instituições de ensino podem utilizar quando é preciso atualizar os aplicativos contidos nos computadores utilizados em salas de aula, e, para o uso pessoal, um indivíduo pode utilizar a fim de atualizar o seu disco rígido, copiando os dados contidos no seu antigo disco rígido para um novo com maior capacidade de armazenamento.

Ferramentas de clonagem de disco podem trabalhar com os mais diversos sistemas de arquivos disponíveis, que são estruturas lógicas e de rotinas, que permitem ao sistema operacional controlar o acesso ao disco rígido. Devem também ter a opção de gerar uma imagem resultante compactada e prover recursos para melhorar a velocidade e precisão no processo de criação e aplicação de uma imagem (GARVRISH, 2008).

Sendo assim as principais contribuições deste trabalho abordam a apresentação dos problemas relacionados a clonagem do disco, explicando quais dados devem ser preservados no processo, a compactação dos dados e como se beneficiar de certos sistemas de arquivos.

Como resultado pretendido para este trabalho é esperado a implementação de um protótipo de um sistema de clonagem de disco, que trabalhe com alguns dos principais sistemas de arquivos hoje existentes, visando uma boa performance e diminuição do tamanho da imagem resultante.

1.1 MOTIVAÇÃO

Assegurar a integridade dos dados de computadores é uma tarefa crítica para qualquer organização, visto que a perda de dados importantes pode ter profundas implicações sobre o futuro de uma empresa. Há diferentes estratégias que podem ser utilizadas para garantir que os dados sejam preservados uma delas é a clonagem de disco, um método muito popular para realizar cópias de segurança do sistema operacional, softwares instalados e dados de uma organização com o mínimo de esforço.

No entanto, softwares de clonagem de disco não são utilizados somente para *backups* do sistema, alguns exemplos de outros usos são:

- Provisionamento de novos computadores: Aplicação de uma imagem contendo um conjunto padrão de softwares e configurações, que pode ser utilizadas para que os novos usuários não percam tempo com a instalação de aplicativos individuais.
- Atualização do disco rígido: O usuário pode copiar todo o conteúdo do seu disco rígido antigo, incluindo o sistema operacional e todos os softwares instalados e configurados, para um novo disco rígido com maior capacidade de armazenamento.
- Sistema de recuperação: Uma imagem que quando aplicada restaura um computador para sua configuração original de fábrica.

Diante deste contexto ressalta-se a necessidade do estudo dessa tecnologia que é um benefício real para quem precisa manter um sistema de computador.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

O objetivo deste trabalho é desenvolver um protótipo de um sistema de clonagem de discos que trabalhe com alguns dos principais sistemas de arquivos hoje existentes.

1.2.2 Objetivos Específicos

- Desenvolver um protótipo de um sistema de clonagem de discos.
- Utilizar algorítimo de compactação de dados para reduzir o tamanho da imagem resultante.

2 ORGANIZAÇÃO DOS DADOS

2.1 SISTEMAS DE NUMERAÇÃO: DECIMAL, BINÁRIO E HEXADECIMAL

Um sistema de numeração é um sistema de escrita para expressão dos números, pessoas estão acostumadas a trabalhar com números decimais, mas, computadores usam binário, onde só existe 0 e 1, cada 0 ou 1 são chamados de *bit*, e *bits* são organizados em grupos de 8 chamados *byte.*

Números decimais são um conjunto numérico que representa os números em base 10 símbolos, portanto empregando 10 símbolos. Assim como pode ser visualizado na figura 1, o número mais a direita da coluna tem o valor 1, cada coluna seguinte tem 10 vezes o valor da coluna anterior, por exemplo, a segunda coluna terá o valor 10 e na terceira 100.

Número: 3,258

1,000	100	10	1
3	2	5	8

(3 x 1,000) + (2 x 100) + (5 x 10) + (1 x 8) = **3,258**

Figura 1 – Valores de cada símbolo em número um decimal

Fonte: PRÓPRIA.

No sistema binário os valores numéricos são representados usando apenas dois valores: 0 e 1, ou seja um sistema de base 2. Os computadores utilizam o sistema binário pois isso torna-os muito mais fáceis de serem implementados utilizando a tecnologia eletrônica atual (BRAIN, [200-]). Assim os dados serão interpretados usando-se o estado da corrente elétrica que possuem dois níveis de tensão: baixo e alto (GUIMARÃES, [19--?]). De acordo com a figura 2, em um sistema de base 2 a coluna mais a direita tem o valor 1 e as colunas seguintes tem 2 vezes o valor da coluna anterior.

Número: 10111100

128	64	32	16	8	4	2	1
1	0	1	1	1	1	0	0

(1 x 128) + (0 x 64) + (1 x 32) + (1 x 16) + (1 x 8) + (1 x 4) + (0 x 2) + (0 x 1) = **188**

Figura 2 – Conversão de binário para decimal

Fonte: PRÓPRIA.

O sistema hexadecimal é um sistema de base 16, ou seja, existem 16 símbolos: 0 a 9 para representar valores de zero a nove, e letras que vão de A a F e representam valores de dez a quinze. Na computação os números hexadecimais são uma forma mais amigável de representar os valores binários, uma vez que cada dígito do sistema hexadecimal pode representar 4 bits, para se representar 1 *byte* em hexadecimal é preciso usar apenas dois dígitos. De acordo com a figura 3, em um sistema de base 16 a coluna mais a direita tem o valor 1 e as colunas seguintes tem 16 vezes o valor da coluna anterior.

Número: 0x1FA4

4,096	256	16	1
1	15	10	4

(1 x 4,096) + (15 x 256) + (10 x 16) + (4 x 1) = **8,100**

Figura 3 – Conversão de hexadecimal para decimal

Fonte: PRÓPRIA.

Para referência a tabela 1 mostra os valores decimais dos primeiros 16 números em representação binária e hexadecimal.

Tabela 1 – Tabela de conversão entre decimal, binário e hexadecimal

Fonte: PRÓPRIA.

Decimal	Binário	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

2.2 ENDEREÇAMENTO E ORDENAÇÃO DE BYTES

Para armazenar dados em um dispositivo é preciso alocar um espaço na memória do mesmo, este espaço é chamado de endereço de memória. O endereço de memória é um identificador único que aponta para um local de memória onde um programa de computador ou outros dispositivos possam armazenar pedaços de dados e depois recuperá-los (UMD, [19--?]).

Um byte é a menor quantidade de espaço que geralmente é alocado para dados, como um byte possui 8 bits este pode representar no máximo 256 valores diferentes (2^8 = 256), porém os bytes podem ser agrupados para que se possa armazenar valores maiores. Em computadores modernos com endereçamento por byte, cada endereço de memória

corresponde a um byte, dados maiores que um byte ocupam um sequencia de bytes consecutivos. Alguns microprocessadores foram projetados para trabalhar com endereçamento por *word*, que é um grupo de tamanho fixo de bits que são tratadas em conjunto pelo sistema, de modo que a unidade de armazenamento seja maior que um *byte* (UMD, [19--?]).

Computadores diferem na maneira que organizam esses dados, alguns usam a ordem *big-endian* e colocam o *byte* mais significante do número no primeiro byte do espaço alocado. Outros usam a ordem *little-endian* onde colocam o byte menos significante do número no primeiro *byte* do espaço alocado (INTEL, 2004).

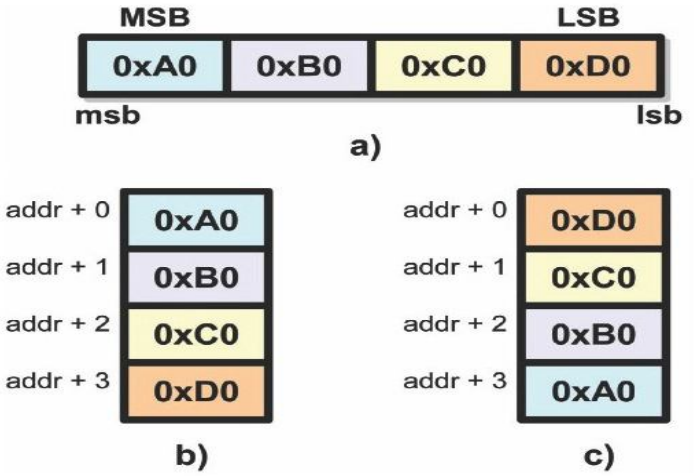

Figura 4 - (a) Dados no registrador. (b) Memória big endian. (c) Memória little endian.
Fonte: LLOPIS, 2009.

2.3 CODIFICAÇÃO DE CARACTERES

Além de números é preciso que os computadores armazenem também letras, a maneira mais comum de fazer isso é codificando os caracteres usando ASCII ou *Unicode*.

ASCII significa Código Padrão Americano para o Intercâmbio de Informação (*American Standard Code for Information Interchange*). Um código ASCII é uma representação

numérica de um caractere do Inglês Americano. Por exemplo, o valor hexadecimal 0x61 representa o caractere 'a'.

Existem 128 definições de caracteres em ASCII, onde 33 desses não são imprimíveis e são usados para controle, afetando na forma de como um texto é processado, 94 caracteres são imprimíveis, estes são letras, pontuação e símbolos técnicos e, por fim, o espaço que é considerado invisível. A figura 5 exibe a tabela de conversão ASCII.

Dec	Hex		Dec	Hex		Dec	Hex		Dec	Hex		Dec	Hex		Dec	Hex		Dec	Hex	
0	00	NUL	16	10	DLE	32	20		48	30	0	64	40	@	80	50	P	96	60	`	112	70	p
1	01	SOH	17	11	DC1	33	21	!	49	31	1	65	41	A	81	51	Q	97	61	a	113	71	q
2	02	STX	18	12	DC2	34	22	"	50	32	2	66	42	B	82	52	R	98	62	b	114	72	r
3	03	ETX	19	13	DC3	35	23	#	51	33	3	67	43	C	83	53	S	99	63	c	115	73	s
4	04	EOT	20	14	DC4	36	24	$	52	34	4	68	44	D	84	54	T	100	64	d	116	74	t
5	05	ENQ	21	15	NAK	37	25	%	53	35	5	69	45	E	85	55	U	101	65	e	117	75	u
6	06	ACK	22	16	SYN	38	26	&	54	36	6	70	46	F	86	56	V	102	66	f	118	76	v
7	07	BEL	23	17	ETB	39	27	'	55	37	7	71	47	G	87	57	W	103	67	g	119	77	w
8	08	BS	24	18	CAN	40	28	(	56	38	8	72	48	H	88	58	X	104	68	h	120	78	x
9	09	HT	25	19	EM	41	29	)	57	39	9	73	49	I	89	59	Y	105	69	i	121	79	y
10	0A	LF	26	1A	SUB	42	2A	*	58	3A	:	74	4A	J	90	5A	Z	106	6A	j	122	7A	z
11	0B	VT	27	1B	ESC	43	2B	+	59	3B	;	75	4B	K	91	5B	[	107	6B	k	123	7B	{
12	0C	FF	28	1C	FS	44	2C	,	60	3C	<	76	4C	L	92	5C	\	108	6C	l	124	7C	|
13	0D	CR	29	1D	GS	45	2D	-	61	3D	=	77	4D	M	93	5D	]	109	6D	m	125	7D	}
14	0E	SO	30	1E	RS	46	2E	.	62	3E	>	78	4E	N	94	5E	^	110	6E	n	126	7E	~
15	0F	SI	31	1F	US	47	2F	/	63	3F	?	79	4F	O	95	5F	_	111	6F	o	127	7F	DEL

Figura 5 – Tabela ASCII

Fonte: PRÓPRIA.

Apesar de toda simplicidade e poder do ASCII este é limitado ao inglês americano, símbolos de outros idiomas não podem ser representados. Para resolver esse problema o *Unicode* foi criado.

O padrão *Unicode* define códigos para as principais línguas escritas hoje, atualmente na versão 5.2, este padrão provê códigos para 107,361 caracteres (UNICODE, 2010).

O *Unicode* apenas define os *codepoints* (a associação de um número único para cada caractere), para se converter o *codepoint* em *bytes* é preciso codificá-los. O padrão Unicode e a ISO/IEC 10646 suportam três formas de codificação: UTF-8, UTF-16 e UTF-32, que são suficientes para atender todos os requisitos de codificação de caracteres conhecidos (UNICODE, 2009).

O UTF-8 é popularmente usado em HTML e outros protocolos. Este tipo de codificação tem a vantagem de ser compatível com o ASCII, sendo necessário apenas um *byte* para codificar os 128 caracteres da tabela ASCII, para os demais caracteres o UTF-8 pode usar de 2 a 4 *bytes* para codificá-los, dependendo do símbolo a ser representado, o

segundo método, UTF-16, utiliza 2 *bytes* para codificar os caracteres mais usados e 4 *bytes* para os menos usados e, por último o UTF-32 que é utilizado em ambientes onde o tamanho dos caracteres codificados precisa ser fixo, este tipo de codificação utiliza 4 *bytes* para armazenar cada caractere (UNICODE, 2009).

2.4 ESTRUTURA DE ORGANIZAÇÃO DE DADOS

Antes de entender como um sistema de arquivo trabalho é preciso saber os conceitos gerais de organização dos dados (CARRIER, 2005).

Para definir como os dados serão armazenados no dispositivo é preciso criar um estrutura de dados para que se possa organizá-los de modo a facilitar o acesso e modificação dos mesmos.

Essa estrutura de dados funciona como um formulário onde cada campo do tem um nome e um tamanho. Um exemplo de estrutura de dados desse tipo seria a MBR (*Master Boot Record*), um modelo que armazena informação da estrutura organizacional do disco.

A MBR utiliza 512 *bytes* divididos da seguinte forma: os primeiros 446 primeiros *bytes* serão usados para armazenar o código de *boot*, 64 *bytes* usados para armazenar a tabela de partição do disco e 2 *bytes* finais para a assinatura de registro. (HOWARD, 1996)

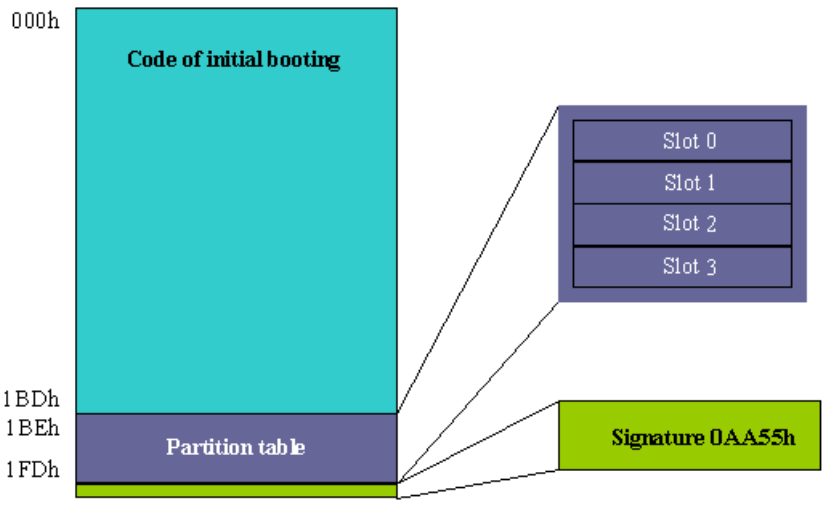

Figura 6 – Estrutura da MBR (Master Boot Record)

Fonte: VORONIN, [19--?].

3 O DISCO RÍGIDO: COMPONENTES E GEOMETRIA

3.1 PRINCIPAIS COMPONENTES

Os componentes de um disco rígido ficam selados em uma caixa de metal para evitar a entrada de materiais externos, esses componentes são muito sensíveis e mesmo pequenas partículas de poeira podem danificar o dispositivo (ALECRIM, 2007a).

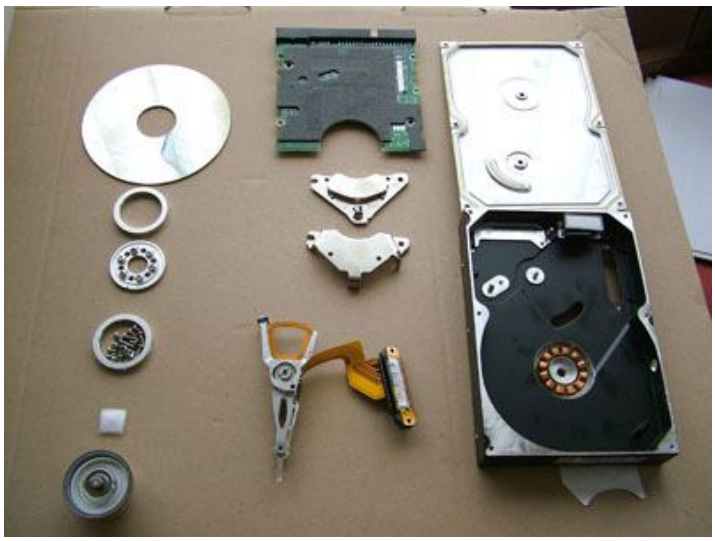

Figura 7 – Disco rígido e seus componentes

Fonte: KINGSLEY-HUGHES, 2005.

3.1.1 Placa controladora

O disco rígido é ligado a placa-mãe através da placa controladora, este circuito integrado que tem como função controlar a rotação do motor e o movimento das cabeças de leitura, transmitir receber e interpretar os dados entre os discos e o computador e efetuar rotinas de segurança.

> A placa lógica, ou placa controladora, é a parte "pensante" do HD. Com exceção dela, o HD é um dispositivo relativamente simples, composto por uma série de dispositivos mecânicos. É a controladora que faz a interface com a placa-mãe, controla a rotação do motor e o movimento das cabeças de leitura, de forma que elas leiam os setores corretos, faz a

verificação das leituras, de forma a identificar erros (e se possível corrigi-los, usando os bits de ECC disponíveis em cada setor), atualiza e usa sempre que possível os dados armazenados no cache de disco (já que acessá-lo é muito mais rápido do que fazer uma leitura nas mídias magnéticas) e assim por diante. (MORIMOTO, 2007a)

Figura 8 – Placa controladora

Fonte: KINGSLEY-HUGHES, 2005.

3.1.2 Motor

É o responsável por manter uma rotação constante, atualmente os discos rígidos utilizam motores de 5,600 RPM (Rotações por minuto), 7,200 RPM e 10,000 RPM, essa velocidade de rotação é um principais fatores que determinam a performance (MORIMOTO, 2002b).

Os motores também são um dos maiores responsáveis por manter a durabilidade do disco, pois a maioria das falhas graves provêm justamente do motor (MORIMOTO, 2002b).

Figura 9 – Motor de rotação

Fonte: KINGSLEY-HUGHES, 2005.

3.1.3 Pratos

São nos pratos que os dados magnéticos são armazenados, os discos rígidos normalmente tem vários pratos que são montados no mesmo eixo, cada prato pode armazenar informações em ambos os lados.

Os pratos são normalmente feitos de alumínio, vidro ou de materiais híbridos de vidro e cerâmica. (KOZIEROK, [entre 1997 e 2004])

Eles são recobertos por um material magnético e por uma camada de material protetor, quanto mais denso for o material magnético, maior é a capacidade de armazenamento do disco (ALECRIM, 2007a).

Figura 10 – Uma foto mostrando o quão fino os pratos são

Fonte: KINGSLEY-HUGHES, 2005.

3.1.4 Braço e Cabeça

A cabeça é um dispositivo que contém uma bobina que utiliza impulsos magnéticos para manipular as moléculas da superfície do disco. Existe uma cabeça para cada lado dos pratos e ficam presas são presas ao braço. O braço tem a função de posicionar as cabeças sob a superfície dos pratos, permitindo assim o acesso a qualquer parte do prato. (ALECRIM, 2007a).

Figura 11 – Braço com uma cabeça presa em cada ponta
Fonte: KINGSLEY-HUGHES, 2005.

3.1.5 Atuador

É o responsável por coordenar o movimento das cabeças de leitura. Para que a movimentação ocorra, o atuador contém em seu interior uma bobina que é "induzida" por imãs. O atuador precisa executar o seu trabalho com bastante precisão, o simples fato da cabeça encostar na superfície de um prato é suficiente para causar danos a ambos (ALECRIM, 2007a).

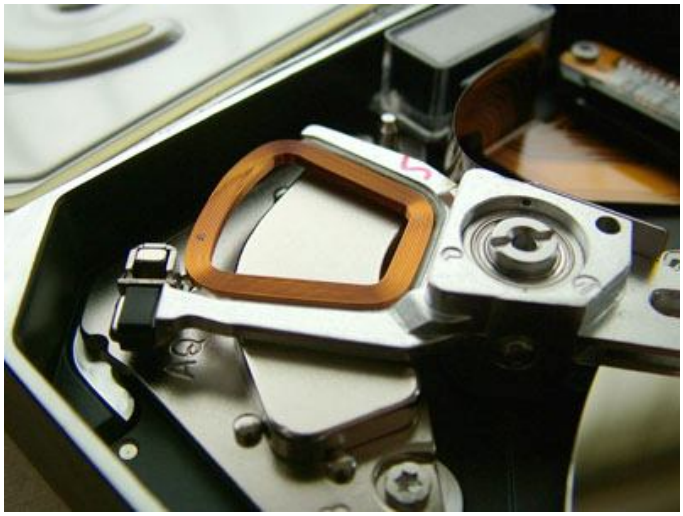

Figura 12 – Atuador

Fonte: KINGSLEY-HUGHES, 2005.

3.2 GEOMETRIA DE DISCO

Cada superfície dos pratos são divididas em anéis concêntricos chamados de trilhas. Uma trilha é dividida em setores, onde cada setor tem possui normalmente 512 bytes. Um conjunto de trilhas com o mesmo número nos vários pratos formam um cilindro. Esse conjunto de divisões é denominado geometria de disco e é um procedimento feito pelo fabricante do disco rígido (ALECRIM, 2007a).

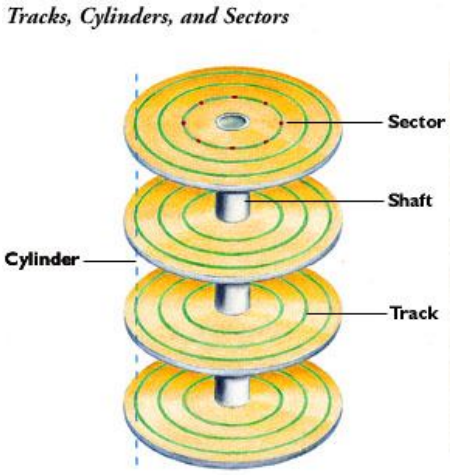

Figura 13 – Trilhas, setores e cilindros

Fonte: MORIMOTO, 2002b.

A geometria de disco é usada para organizar o processo de gravação e leitura dos dados gravados no disco rígido. (MORIMOTO, 2002b).

As trilhas são contadas a partir da borda do disco e vão se tornando menores conforme se aproximam do centro, sendo assim, a trilha mais externa recebe o número 0 e as seguintes recebem os números 1, 2, 3, e assim por diante (ALECRIM, 2007a).

Cada trilha é dividida em setores, que são pequenos trechos onde dados serão armazenados. Para definir o limite entre uma trilha e outra, assim como determinar onde um setor termina e o outro começa, são usados marcas de endereçamento, essas marcas são pequenas áreas com um sinal magnético especial para que a cabeça do disco seja orientada e permita que a controladora de disco possa localizar os dados desejados. (MORIMOTO, 2002b).

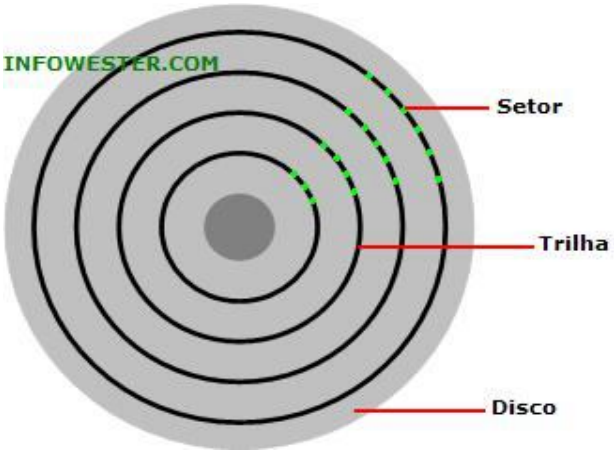

Figura 14 – Superfície de um prato dividido por trilhas e setores.

Fonte: ALECRIM, 2007a.

Um disco rígido pode conter vários pratos, porém apenas um braço, assim quando o disco precisa ler alguma trilha que está na superfície de algum dos pratos o braço posicionará a cabeça até essa trilha e fará com que as demais cabeças fiquem posicionadas sob o mesmo número de trilha. Para esse evento é dado o nome de cilindro, ou seja, o cilindro é um conjunto de trilhas com o mesmo número nos vários pratos (ALECRIM, 2007a).

4 SISTEMAS DE ARQUIVOS

Um sistema de arquivos é um conjunto de estruturas lógicas e de rotinas, que permitem ao sistema operacional controlar o acesso ao disco rígido (MORIMOTO, [entre 1999 e 2010]c).

O sistema de arquivo nada mais é que um mecanismo feito para organizar os dados em um dispositivo de armazenamento, dessa maneira o sistema operacional pode interpretá-lo e então gravar e recuperar os dados de dentro do dispositivo (CARRIER, 2005).

Além das funções básicas para manipular os dados do dispositivo, diferentes sistemas de arquivos oferecem diferentes recursos. Por exemplo, o recurso de *journaling,* que mantem um *log* sobre todas as operações feitas em arquivos do disco, assim, quando algum erro inesperado surge, ou o sistema é desligado incorretamente é possível localizar todas as operações que não haviam sido completadas e restaurar a consistência do sistema de arquivos (MORIMOTO, [entre 1999 e 2010]d).

Diferentes sistemas operacionais usam diferentes sistemas de arquivos. Atualmente o NTFS é o sistema de arquivos mais utilizados e recomendando para utilização com o sistema operacional Windows (MICROSOFT, 2010).

O Sistema operacional Linux tem suporte a diversos sistemas de arquivos, tanto nativos ou provenientes de outros sistemas operacionais. Porém os mais utilizados atualmente ficam por conta da família EXT2, EXT3 e EXT4.

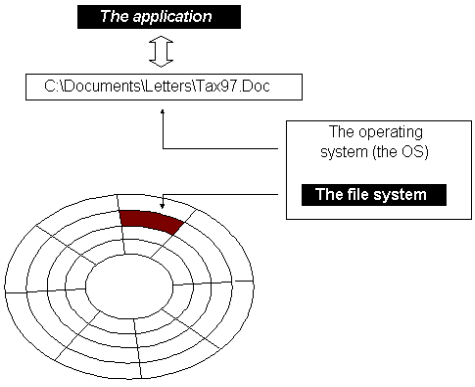

Figura 15 – Uma aplicação pede para ler um arquivo do disco rígido, o sistema operacional pede para o sistema de arquivo encontrá-lo.

Fonte: KARBO, [entre 1996 e 2005].

4.1 PARTICIONAMENTO

O Particionamento é o processo de dividir o disco rígido em várias partes, conhecidas como partições. Cada partição é independente da outra, ou seja, cada uma pode ter o seu próprio sistema de arquivos, assim, é possível por exemplo, instalar o sistema operacional GNU/Linux em uma partição e o sistema operacional Windows em outra partição. (DINIZ, [19--?])

Uma partição pode ser do tipo primária ou estendida, e, o disco rígido pode ter no máximo quatro partições, onde, pelo menos uma dessas partições tem que ser primária, e, apenas uma pode estendida (KIOSKEA, 2008).

4.1.1 Partição Primária

Uma partição primária não pode ser subdividida, assim só é possível conter um sistema de arquivos. Alguns sistemas operacionais precisam ser instalados em uma partição primária para que funcionem corretamente, porém, uma partição primária não precisa necessariamente ter um sistema operacional. (ALECRIM, 2005b)

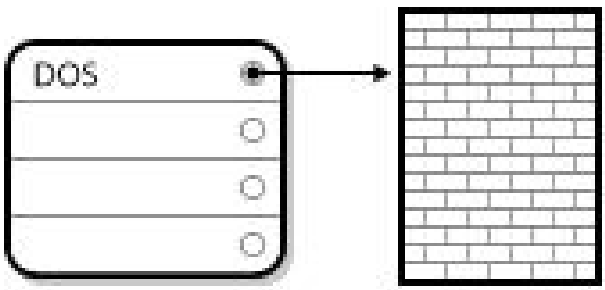

Figura 16 – Drive de disco com uma Partição Primária

Fonte: RED HAT, 2005.

4.1.2 Partição Estendida

Como a tabela de partições da MBR só suporta 4 entradas, foi preciso criar uma maneira para que um disco possa ser particionado mais do que quatro vezes, assim surgiu o conceito de partição estendida. Uma partição estendida é uma espécie de contêiner, dentro dela é possível endereçar até 255 partições, denominadas partições lógicas. Cada partição lógica pode ter o seu próprio sistema de arquivos (MORIMOTO, [entre 1999 e 2010]e).

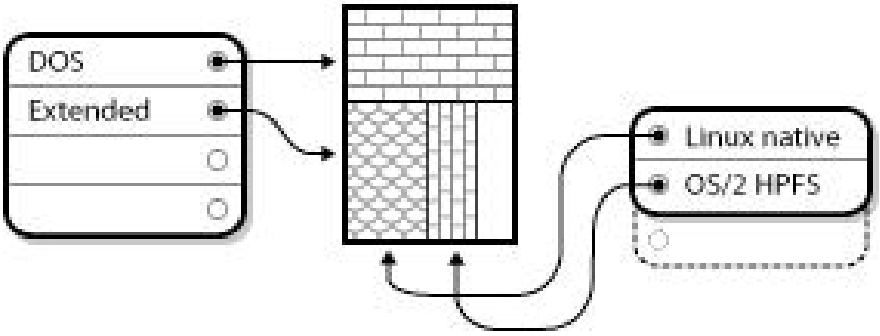

Figura 17 – Drive de disco com Partição Estendida.
Fonte: RED HAT, 2005.

4.2 MBR

Quando o computador é ligado o processador ativa o processo de processamento de dados, mas, uma vez que a memória está vazia o processador não tem nada para executar. Para lidar com isso os fabricantes de *chip* e *bios* desenvolveram o seu código para que o processador, uma vez ativado, sempre inicia a execução no mesmo local, ffff0h. (DEW, [entre 1995 e 2002])

Da mesma forma cada disco rígido tem o seu ponto de partida, onde as informações sobre o disco são armazenadas. Também deve haver um lugar onde a bios possa carregar o programa de inicialização que inicia o processo de *boot* do sistema operacional. O lugar onde essas informações são armazenadas é chamado de MBR (DEW, [entre 1995 e 2002]).

A MBR em um disco está sempre localizada no cilindro 0, cabeça 0 e setor 1, ou seja, no primeiro setor do disco. Assim quando o computador é ligado ele sempre irá olhar para esse primeiro setor de informações para saber como proceder com o processo de *boot* e carregar o sistema operacional (DEW, [entre 1995 e 2002]).

A MBR tem a seguinte estrutura:

Tabela 2 – Estrutura da MBR

Fonte: PRÓPRIA.

Endereço		Descrição		Tamanho em octetos
Hex	Dec			
0000	0	Código do *bootloader*		446
01BE	446	Tabela de partições		64
01FE	510	55h	Assinatura da MBR 0xAA55	2
01FF	511	AAh		
Tamanho total da MBR: 446 + 64 + 2 =				**5**

4.2.1 Área do código

Os primeiros 446 *bytes* da MBR são reservados para o código que é responsável por fazer o sistema operacional iniciar. Logo após o computador ser iniciado a *bios* irá fazer suas checagens de rotina e então irá carregar os primeiros 446 *bytes* da MBR, que ficará com o controle do computador e então possa iniciar o sistema operacional (KIOSKEA, 2008).

4.2.2 Tabela de partições

A tabela de partições armazenada da MBR contém 64 *bytes* que são divididos em 4 partes de 16 *bytes,* um para cada entrada. Conforme a tabela abaixo:

Tabela 3 – O padrão de 64 *bytes* da tabela de partições da MBR

Fonte: PRÓPRIA.

Endereço		Tamanho (em *bytes*)	Descrição
Dec.	Hex.		
446 - 461	1BE - 1CD	16	Entrada para a partição 1
462 - 477	1CE - 1DD	16	Entrada para a partição 2
478 - 493	1DE - 1ED	16	Entrada para a partição 3
494 - 509	1EE - 1FD	16	Entrada para a partição 4

Cada entrada de partição possui a seguinte estrutura:

Tabela 4 – Estrutura de 16 *bytes* de uma entrada de partição

Fonte: PRÓPRIA.

Endereço	Tamanho (em *bytes*)	Descrição
0	1	Indicador de *boot*
1 - 3	3	Endereço CHS inicial
4	1	Tipo da partição
5 - 7	3	Endereço CHS final
8 - 11	4	Setor inicial
12 - 15	4	Tamanho da partição (em setores)

1. **Indicador de *boot*:** Indica se aquela é a partição “ativa” (bootável) para ser iniciada, gerenciadores de *boot* não utilizam essa informação (SEDORY, [entre 2004 e 2007]).
2. **Endereço CHS inicial:** CHS em inglês é uma sigla para cylinder, head, sector (cilindro, cabeça, setor). Essa informação é utilizada para identificar a localização do primeiro setor da partição . Com o número do cilindro junto com o número da cabeça, é possível identificar a trilha. Uma vez que a trilha é identificada, pode-se usar a informação do setor e assim saber qual o primeiro setor da partição (SEDORY, [entre 2004 e 2007]).
3. **Tipo da partição:** O tipo da partição é um número que indica o uso antecipado da partição, essa marcação pode ser utilizada para denotar um tipo específico de sistema de arquivo, ou para indicar que a partição está associada a um determinado sistema operacional (RED HAT, 2005).

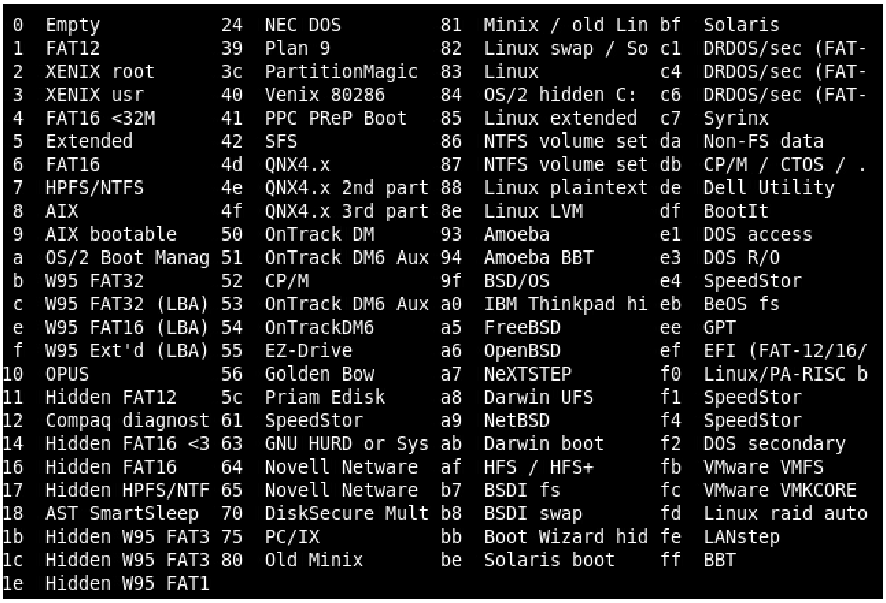

Figura 18 – O aplicativo fdisk listando os tipos de partição.

Fonte: PRÓPRIA.

4. **Endereço CHS final:** Endereço CHS final.
5. **Setor Inicial:** Este valor identifica o primeiro setor da partição exatamente como valores iniciais do CHS faz, porém o faz utilizando LBA (*Logical Block Addressing)* ou Endereçamento de Bloco Lógico, que é uma maneira comum usada para especificar a localização dos blocos de dados armazenados em dispositivos de armazenamento (SEDORY, [entre 2004 e 2007]).
6. **Tamanho da partição:** Armazena o tamanho da partição em número setores (SEDORY, [entre 2004 e 2007]).

4.2.3 Assinatura da MBR

O MBR termina com dois *bytes*, que são definidos como o número mágico (0xAA55). O número mágico funciona como verificação de validação do MBR. (JONES, 2006)

5 METODOLOGIA

Para realização do presente estudo foi desenvolvido inicialmente uma pesquisa bibliográfica na qual foi possível familiarizar-se com as características e peculiaridades do tema a ser explorado, segundo a autora STUMPF (2008), esta pesquisa é o planejamento inicial tipo qualquer trabalho que envolva pesquisas.

A pesquisa aplicada também foi utilizada neste projeto, essa pequisa fez com que todo o levantamento de dados dos principais problemas encontrados na pesquisa bibliográfica fossem analisados, e a aplicação fosse desenvolvida transferindo estes resultados para a realidade.

Para o desenvolvimento do protótipo de clonagem de disco foi utilizada a linguagem de programação Python, uma linguagem de alto nível, interpretada e orientada a objetos, projetada para priorizar a importância do esforço do programador sobre o esforço computacional, possui uma sintaxe clara e concisa priorizando a legibilidade do código, conta com recursos poderosos de sua biblioteca padrão e por módulos e frameworks desenvolvidos por terceiros. Python também pode ser facilmente estendido utilizando linguagem C, muitas das bibliotecas desenvolvidas em C ou C++ são utilizadas através de *bindings*, que é uma a ligação entre bibliotecas para o uso direto no interpretador. A linguagem foi criada por Guido van Rossum em 1991. A linguagem vem crescendo em rítimo acelerado nos últimos anos, e hoje já é possível encontrar Python nas maiores empresas de desenvolvimento técnologico do mundo, alguns exemplos são: Google, Nasa, Nokia, MIT, Mandriva, Serpro.

Como o estudo trata-se de um protótipo, apenas três sistemas de arquivos são suportados pela aplicação: NTFS, Ext3 e Linux Swap. NTFS é o sistema de arquivos padrão do Windos NT e seus derivados: Windows 2000, Windows XP, Windows Vista e Windows 7. O Ext3 é utilizado por diversas distribuições GNU/Linux como sistema de arquivos padrão, porém, hoje em dia vem sendo substituído pelo seu sucessor Ext4. Partições que utilizam um sistema de arquivos para *swap* no Linux não terão seus dados copiados pela aplicação, uma vez que este é simplesmente utilizada pelo sistema operacional como área de troca (uma forma de armazenar o conteúdo que está na memória primária do computador para a memória secundária) para restaurar uma partição de *swap* para linux, a aplicação precisa

copiar apenas o UUID desse sistema de arquivos. Outros sistemas de arquivos devem continuar funcionando, uma vez que aplicação tratará os demais de forma genérica, extraindo e aplicando os dados por meio de *byte copy,* ou seja, todos os bytes contidos nestas partições serão copiados e depois restaurados.

Para a aplicação funcionar corretamente o disco de origem ou destino deve estar ocioso, ou seja, não é possível criar ou aplicar a clonagem em um disco que está sendo utilizado por outros processos, para resolver esse problema foi necessário desenvolver uma base sólida e extensível para o projeto, pois este não poderiam depender dos sistemas operacionais instalados na máquina. O sistema baseia-se em GNU/Linux Ubuntu 10.10, para sua construção foi utilizando a ferramenta *deboostrap* na qual permite instalar uma base de um sistema operacional em um subdiretório de outro, e, os scripts de apoio foram escritos em linguagem Python. O sistema tem foco em tamanho mínimo e velocidade no *boot* e pode ser carregado a partir de um CD-ROM ou Pen-drive. A escolha do sistema operacional GNU/Linux foi feita devido a quantidade de sistemas de arquivos suportados pelo *kernel* e pela possibilidade de carregar o sistema operacional a partir de múltiplas mídias como: *CD-ROM, Hard disk, Pen-drive* e *PXE*.

Para a escolha do algorítimo de compactação de dados foi levado em conta a velocidade que este executa sua tarefa, o algorítimo zip foi escolhido devido a performance apresentada frente aos outros algorítimos testados BZ2 e LZMA, convém ressaltar que ambos os algorítimos se mostraram eficazes no processo de compactação de dados.

Um ambiente virtualizado foi utilizado para o desenvolvimento do estudo, este conta um hardware hospedeiro com as seguintes configurações: Processador AMD Athlon(tm) 64 X2 Dual-Core Processor TK-53, 2048MB DDR2 SDRAM, 128MB ATI Mobility Radeon.

A aplicação desenvolvida possui funcionalidades: criação de uma imagem a partir de um dispositivo de disco e restauração dos arquivos de clonagem em um ou mais dispositivo(s) de disco. A figura 19 mostra um menu com todas as opções disponíveis pela aplicação.

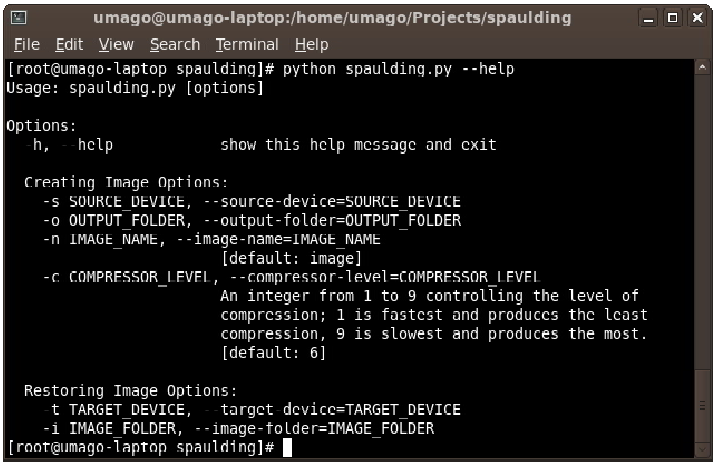

Figura 19 – Aplicação exibindo o menu de ajuda

Fonte: PRÓPRIA.

No processo de criação da imagem, o usuário deve entrar com as informações sobre o dispositivo de origem e a pasta de destino onde serão gravados os arquivos da imagem. A partir dessa entrada de dados do usuário é verificado se o dispositivo escolhido como origem é um dispositivo de disco ou uma partição de um disco, essa informação é obtida utilizando as convenções de nomes de um sistema Linux. Por exemplo um dispositivo SCSI será nomeado como sda, sdb, sdc, sdd... etc dentro do diretótio /dev. Da mesma forma, as partições deste discos serão nomeadas acrescentando um valor inteiro na frente do nome para indicar o número da partição como: sda1, sda2, sda5, sdb1, sdb2. Assim para descobrir se o dispositivo é uma partição de um disco verifica-se se o dispositivo contém um inteiro como sufixo.

No caso da criação de uma imagem onde o dispositivo de entrada é um dispositivo de disco, será preciso copiar alguns dados adicionais para que o processo de restauração funcione corretamente, neste caso é copiado para um arquivo os primeiros 446 *bytes* da MBR que incluem o código do *bootloader*, e, utilizando a biblioteca *pyparted* será gravado em um arquivo uma estrutura de dados que representa a tabela de partição desse disco.

O próximo passo é gerar uma lista com as partições desse disco que contém sistemas de arquivos válidos, esses sistemas de arquivos não precisam ser suportados pela nossa aplicação. Partições que não contém sistemas de arquivos válidos serão descartadas, assim o tamanho da imagem resultante será menor. No caso do dispositivo de entrada ser uma partição de um disco essa lista irá conter o próprio dispositivo.

Após a geração da lista das partições que serão feitas a clonagem é possível obter algumas informações importantes para o processo de criação da imagem, primeiro será a soma do espaço utilizado em cada uma dessas partições, partições que não são suportadas pela nossa aplicação esse espaço utilizado será o tamanho em *bytes* total dessa partição. A partir dessa some é feita uma divisão do total de *bytes* utilizados de todas as partições pelo tamanho do bloco de dados que a aplicação irá trabalhar, no caso 1024 *bytes* por bloco, assim vamos saber quantos blocos de dados a imagem resultante irá conter.

Através da figura 20 pode-se observar uma porcentagem que contabiliza o processo de criação da imagem, esses valores são obtidos utilizando a quantidade de total de blocos de dados para se gerar a imagem.

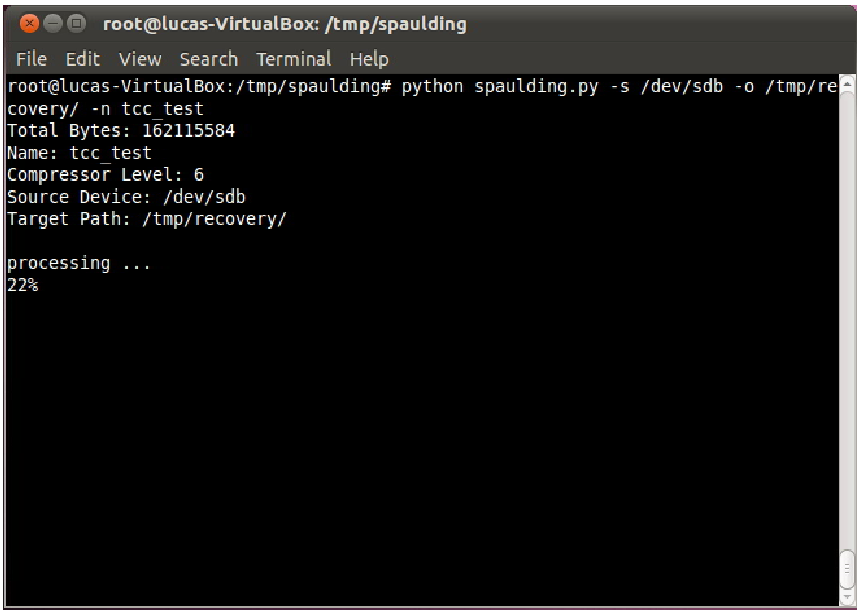

Figura 20 – Aplicação clonando um disco

Fonte: PRÓPRIA.

Agora é preciso percorrer cada partição, retirando informações específicas de cada uma como: número da partição, o tipo de sistema de arquivo e em alguns casos, o UUID. Em seguida a partição será colocada em modo de leitura. Cada bloco lido dessa partição será compactado utilizando o algorítimo de compactação de dados zip e o dado resultante dessa compactação será gravado em um arquivo.

Na última etapa o sistema grava no disco um arquivo em formato XML contendo as informações obtidas em todo o processo de criação, esse arquivo será utilizado posteriormente no processo de restauração dessa imagem. Através da figura 21 é possível ver um exemplo desse arquivo.

```
−<image is_disk="True" name="tcc_test" total_bytes="162115584" compressor_level="6">
   <partition number="1" uuid="69ef7f84-9787-4e41-a665-df934e0950db" type="ext3"/>
   <partition number="5" type="ntfs"/>
   <partition number="2" uuid="bb312563-abcb-48e8-b2b6-dc6e927ab291" type="linux-swap(v1)"/>
 </image>
```

Figura 21 – Exemplo de um XML gerado pela aplicação

Fonte: PRÓPRIA.

Na figura 22 é possível observar um diagrama do processo de criação.

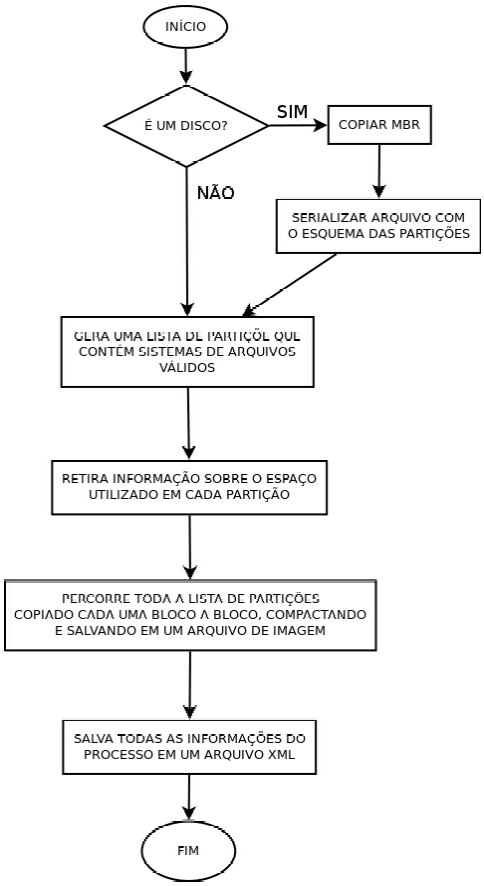

Figura 22 – Diagrama de criação da imagem

Fonte: PRÓPRIA.

Um problema identificado no processo de criação de uma imagem a partir de uma partição NTFS, foi a consistência do sistema de arquivos, para ter sucesso em criar uma imagem é preciso que o sistema de arquivos NTFS esteja livre de erros. Uma possível correção para o problema consiste em utilizar o aplicativo *chkdsk* na partição a partir de um sistema operacional Windows.

No processo de restauração da imagem, o usuário deve entrar com as informações sobre o dispositivo de destino e a pasta de onde se encontram a imagem a ser restaurada.

A primeira coisa que a aplicação irá fazer é carregar as informações armazenadas no XML descritor da imagem e verificar se o dispositivo de destino é do mesmo tipo que o de origem. Uma clonagem feita de um partição só poderá ser aplicada em outra partição da mesma forma que uma clonagem feita de um disco só poderá ser aplicada em um disco. No caso da imagem pertencer a um disco, a aplicação irá restaurar o código do *bootloader* e o esquema de partições nesse dispositivo de destino.

A seguir a aplicação irá iterar por cada partição do disco recém criado, e, consultando o XML descritor da imagem irá restaurar as informações específicas de cada uma delas. Nessa iteração também será feita a restauração dos dados, a aplicação irá ler bloco a bloco o arquivo que contém os dados das partições, irá descompactá-los e popular a nova partição.

Alguns cuidados devem ser tomados no processo de restauração, as partições onde a imagem será restaurada deve ser maior ou ter exatamente o mesmo tamanho da partição de origem, independente do espaço utilizado que havia na origem. Na figura 23 é possível observar um diagrama do processo de restauração.

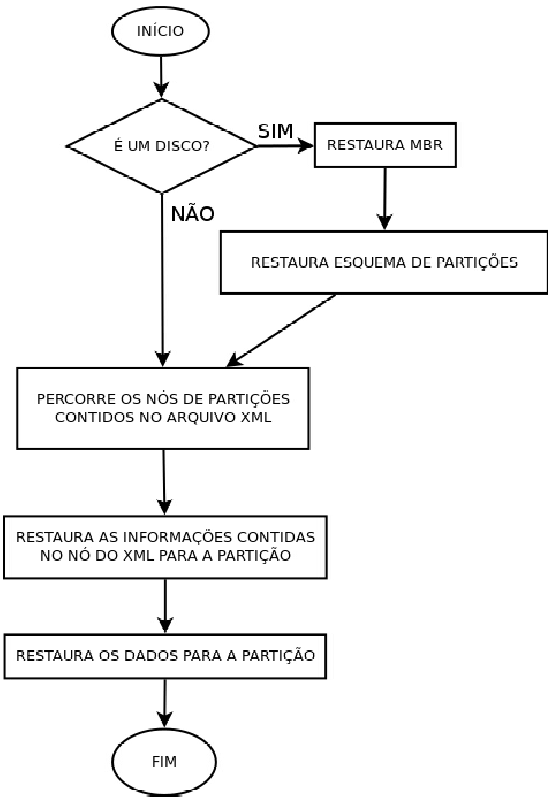

Figura 23 – Diagrama de restauração da imagem

Fonte: PRÓPRIA.

Um problema foi identificado no processo de restauração de um disco que continha um sistema operacional GNU/Linux Ubuntu 8.10, após a restauração da imagem, o *bootloader* GRUB 0.97 não conseguia mais iniciar o sistema operacional contido no disco, para resolver esse problema foi preciso reinstalar o *bootloader* a partir de um Live-CD. A figura 24 mostra o erro exibido após a aplicação da imagem.

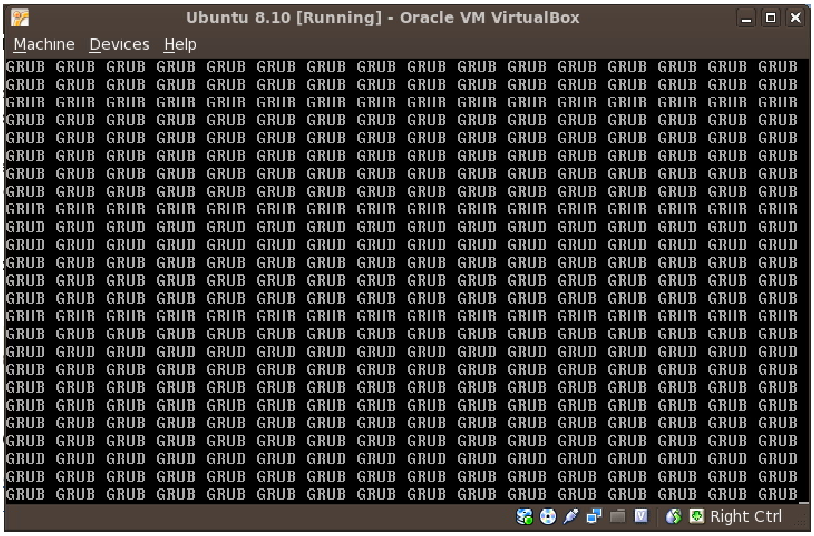

Figura 24 – Bootloader com erro após aplicação da imagem

Fonte: PRÓPRIA.

6 RESULTADOS OBTIDOS

Dos resultados pode-se concluir que a técnica utilizada para se fazer a clonagem de disco obteve sucesso, todas as operações foram realizadas sem grandes problemas, não houve perca de dados e o tamanho da imagem resultante do processo de clonagem obteve uma boa taxa de compactação, chegando a mais de 70% em alguns casos. Todos os problemas encontrados nos processos de criação e restauração das imagens foram facilmente corrigidos utilizando ferramentas básicas encontradas nos próprios sistemas operacionais que estavam sendo clonados.

A escolha do algorítimo zip para fazer a compactação dos dados mostrou-se viável devido a sua velocidade em relação ao BZ2 e ao LZMA, principalmente quando um nível baixo de compactação era utilizado.

A escolha da linguagem de programação Python para este tipo de aplicação obteve sucesso, a simplicidade do código junto a técnica de orientação a objetos gerou um código fácil de ser mantido, otimizado e estendido para suportar mais sistemas de arquivos.

7 CONCLUSÃO

Verificou-se nesta pesquisa que a técnica utilizada neste trabalho para se fazer clonagem de discos é viável, uma vez que esta mostrou uma redução de tamanho da imagem resultante e, principalmente, garantiu a consistência dos dados após o processo de restauração. O estudo também permitiu concluir que a utilização de linguagens de programação de alto nível como Python para este tipo de aplicação é totalmente aceitável, deixando o programador mais livre para cuidar da lógica da aplicação e não tendo que se preocupar com características específicas da arquitetura do computador, o código final impressiona pela sua simplicidade, facilidade de manutenção e extensão.

Convém evidenciar que não foi propósito desta pesquisa medir desempenho da geração e restauração das imagens, entende-se que este objetivo poderá ser alcançado através de outra investigação.

8 REFERÊNCIAS BIBLIOGRÁFICAS

ALECRIM, E. **Conhecendo o disco rígido.** 2007a Disponível em: <http://www.infowester.com/hds1.php> Acesso em: 16 abr 2010.

ALECRIM, E. **Como usar o fdisk da Microsoft.** 2005b Disponível em: <http://www.infowester.com/tutfdisk.php> Acesso em: 30 mai 2010.

BRAIN, M. **How Bits and Bytes Work.** [200-] Disponível em: <http://www.howstuffworks.com/bytes.htm> Acesso em: 11 abr 2010.

CARRIER, B. **File System Forensic Analysis.** Massachusetts: Addison Wesley, 2005. 600 p.

D, M. **Beginners Guides: Cloning WindowsXP.** 2008 Disponível em: <http://www.pcstats.com/ArtVNL.cfm?articleID=418> Acesso em: 4 abr 2010.

DEW. **The Master Boot Record (MBR) and Why it is Necessary?.** [entre 1995 e 2002] Disponível em: <http://www.dewassoc.com/kbase/hard_drives/master_boot_record.htm> Acesso em: 30 mai 2010.

DINIZ, M. **Sistemas de arquivos.** [19--?] Disponível em: <http://www.uniriotec.br/~morganna/guia/sistemas_de_arquivos.html> Acesso em: 9 mai 2010.

GARVRISH, B. **How Does Hard Drive Cloning Work?.** 2008 Disponível em: <http://www.ehow.com/how-does_4605913_hard-drive-cloning-work.html> Acesso em: 4 abr 2010.

GUIMARÃES, D. P. **Sistemas Numéricos.** [19--?] Disponível em: <http://www.tecnobyte.com.br/sisnum1.htm> Acesso em: 11 abr 2010.

HOWARD, G. **Partitions and Volumes.** 1996 Disponível em: <http://www.yale.edu/pclt/BOOT/PARTITIO.HTM > Acesso em: 11 abr 2010.

INTEL. **Endianness White Paper.** 2004 Disponível em: <http://www.intel.com/design/intarch/papers/endian.pdf> Acesso: 11 abr 2010.

JONES M. T. **Por Dentro do Processo de Inicialização do Linux.** 2006 Disponível em: <http://www.ibm.com/developerworks/br/library/l-linuxboot/> Acesso em: 9 mai 2010.

KARBO. **About file systems: DOS formatting, FAT, etc.** [entre 1996 e 2005] Disponível em: <http://www.karbosguide.com/hardware/module6a1.htm> Acesso em: 30 mai 2010.

KINGSLEY-HUGHES, A. W. **Take a peek inside a hard drive!.** 2005 Disponível em: <http://www.pcdoctor-guide.com/wordpress/?p=595> Acesso em: 16 abr 2010.

KIOSKEA. **Partition - Partitioning a hard drive**. 2008 Disponível em: <http://en.kioskea.net/contents/repar/partitio.php3> Acesso em: 30 mai 2010.

KOZIEROK, C. M. Hard Disk Drives. [entre 1997 e 2004] Disponível em: <http://www.pcguide.com/ref/hdd/op/mediaMaterials-c.html> Acesso em: 16 abr 2010.

LLOPIS, N. **Secrets of Multiplatform Data Baking.** 2009 Disponível em: <http://www.gamasutra.com/view/feature/3998/secrets_of_multiplatform_data_.php> Acesso em: 03 jun 2010.

MICROSOFT. **Escolhendo um sistema de arquivos**: **NTFS, FAT ou FAT32.** 2010 Disponível em: <http://technet.microsoft.com/pt-br/library/cc787653(WS.10).aspx> Acesso em: 9 mai 2010.

MORIMOTO, C. E. **Hardware, o Guia Definitivo.** 2007a Disponível em: <http://www.gdhpress.com.br/hardware/leia/index.php?p=cap5-3> Acesso em: 16 abr 2010.

MORIMOTO, C. E. **Hardware, Manual Completo.** 2002b Disponível em: <http://www.gdhpress.com.br/hmc/leia/index.php?p=cap5-2> Acesso em: 16 abr 2010.

MORIMOTO, C. E. **Sistema de arquivos.** [entre 1999 e 2010]c Disponível em: <http://www.guiadohardware.net/termos/sistema-de-arquivos> Acesso em: 16 mai 2010.

MORIMOTO, C. E. **Journaling.** [entre 1999 e 2010]d Disponível em: <http://www.guiadohardware.net/termos/journaling> Acesso em: 2 mai 2010.

MORIMOTO, C. E. **Formatação.** [entre 1999 e 2010]e Disponível em: <http://www.guiadohardware.net/termos/formatacao> Acesso em: 30 mai 2010.

PATTERSON, J. M. **Data Backup Reporting in a Nutshell.** 2010 Disponível em: <http://ezinearticles.com/?Data-Backup-Reporting-in-a-Nutshell&id=3894334> Acesso em: 4 abr 2010.

PRITCHARD, A. **Effective Teaching With Internet Technologie.** 1. ed. Thousand Oaks:

Sage Publications, 2007. 144 p.

RED HAT. **Guia de Instalação para Arquitetura POWER da IBM®.** 2005 Disponível em: <http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-ig-ppc-multi-pt_br-4/> Acesso em: 30 mai 2010.

SEDORY, D. B. **MBR/EBR Partition Tables.** [entre 2004 e 2007] Disponível em: <http://thestarman.narod.ru/asm/mbr/PartTables.htm> Acesso em: 30 mai 2010.

STUMPF, I. R. **Métodos e técnicas de pesquisa em comunicação.** São Paulo: Atlas, 2008. 51 p.

TYAGI, T. **Data Recovery with & without Programming.** 2004 Disponível em: <http://www.p-dd.com/data-recovery-programming-book.html> Acesso em: 4 abr 2010.

UMD. **Big and Little Endian.** [19--?] Disponível em: <http://www.cs.umd.edu/class/sum2003/cmsc311/Notes/Data/endian.html> Acesso em: 11 abr 2010.

UNICODE. **The Unicode® Standard: A Technical Introduction.** 2010 Disponível em: <http://www.unicode.org/standard/principles.html> Acesso em: 11 abr 2010.

VORONIN A. **Data storage on hard discs.** [19--?] Disponível em: <http://ixbtlabs.com/articles/bootman/> Acesso em: 11 abr 2010.