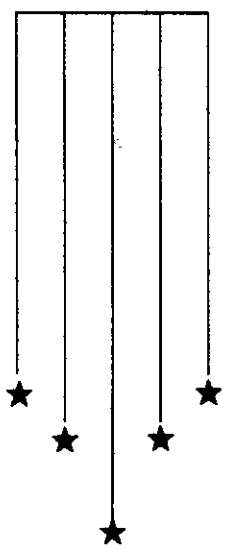
DISK UTILITIES

PROJECT WORK



Submitted by

B.KarthiKeyan(0028q0133)
R.Venkatasubramanian(0028q0165)

Under the guidance of

Mr. C.RajanKrupa M.C.A
Computer Technology Department

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF SCIENCE

(Applied Science -Computer Technology)

of the BHARATHIAR UNIVERSITY, Coimbatore.

DEPARTMENT OF COMPUTER TECHNOLOGY

KUMARAGURU COLLEGE OF TECHNOLOGY

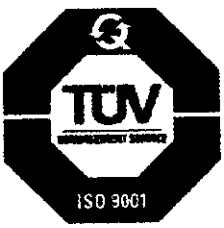
COIMBATORE-641 006.

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE:641 006

Department of Computer Technology

Certificate



This is to certify that this project entitled

DISK UTILITIES

has been submitted by Mr. B.KARTHIKEYAN, R.VENKATASUBRAMANIAN

In partial fulfillment of the requirements for the award of degree of Bachelor of Science Applied Science Computer Technology of Bharathiar University, Coimbatore:641 046 during the academic year 2002-2003.

C. Rajamanickam
(Guide)

(Head of Department)

Certified that the Candidate was Examined by us in the Project Work Viva-Voce Examination held on 25.03.2003

University Register Number 0028Q0133, 0028Q0165

25/3
(Internal Examiner)

(External Examiner)

ACKNOWLEDGMENT

"Great things are not performed with strength but with prayer and perseverance". I thank the Almighty for His abundant showers of blessings.Any project big or small cannot be successful without the support of many people. Here I deem fit to express my gratitude to them.

We are bound to express our gratitude to **Dr. K.KPadmanaban, B.Sc(Engg), M.Tech, Ph.D**., Principal, Kumaraguru College of Technology, for his constant encouragement throughout our course.

We wish to thank **Dr.V. Sundaram, M.Sc, Ph.D.,** Head of the Department, Computer Technology, Kumaraguru College of Technology, for constantly encouraging us to pursue new goals and ideas.

We wish to thank **Mrs.V.Geetha M.C.A,** Course Co-ordinator, for her valuable efforts and help throughout our project work.

We are very grateful to **Mr. S.Hameed Ibrahim M.C.A.** Class Advisor for his valuable suggestions and support throughout our project work.

We wish to express our gratitude and sincere thanks to our project guide **Mr. C.RajanKrupa M.C.A,** Department of Computer

Technology, Kumaraguru College of Technology, for being supportive through out the tenure of the project.

We wish to thank all our friends and our family members who were showing this contributions in many subtle ways and indeed instrumental in achieving the result.

SYNOPSIS

A floppy will be unusable if there is an error in reading the first track. Using our tool we are trying to make these floppies usable by transferring the MBR to the next successive track in the floppy. It makes the floppy to be usable through our tool. The Scanning option for the floppy is provided which gives a report about free space, sectors, clusters, sector per cluster and bad sectors.

The partitioning of the harddisk is dividing the harddisk into separate logical drives. Since large capacity, harddisk is not advisable to use as it is. So it is partitioned into different logical drives to increase the performance of the disk. An option for displaying the partition information along with the tool is provided.

A Formatter is also provided with our tool to ensure the floppy to be clean before using it after recovery. The option for saving and restoring the Mbr is provided as an additional option.

Dedicated to Our Beloved

Parents

Table of contents

Chapter 1

Introduction

Floppies are generally used as a portable storage device for carrying small amount of data from one place to the other with ease. The Floppies are working on the principle of magnetism. The head of the Floppy Disk Drive records the data through Electromagnetic principle. The transfer of bits between the disk surface and read/write head of the floppy disk drive is done only through Magnetic fields.

In our tool we are trying to provide the user all possible options in the floppy drive for the sake of convenience.

Need for Project

The Project is mainly focused in the field of system side utilities. The project entitled "Disk utilities" gives the user variety of options for the sake of convenience. The main objective of this project is to satisfy the basic requirements of the user. Such as scanner, formatter etc.,

The tool can be applied anywhere, when there is a personal computer. Since, it is wide spread; there is no specific area of application for this project.

The floppy disks are normally fragile in nature. They should handle with great care since it is made of plastic a very light material. The floppies normally consist of tracks & sectors. So when the first track of floppy normally contains any errors. The floppy will be normally unusable.

So in order to make these floppies usable we are trying to recover the MBR of the floppy. Normally the system recognizes a FDD only by accessing the MBR. When the MBR gets affected it will make the floppy unusable.

So we are rewriting the MBR of the floppy and making it reusable. We are also providing the formatting option for the floppy to ensure the reusable floppy clean before using it.

Chapter 2

System Analysis

Existing System:

The Existing tool for the recovery of the Floppy disk is not effective. Normally the floppies are thrown away when it gives some error. Since the floppies are very cheap now. There is no single tool, which can perform all the available options to the floppy such as the scanner, formatter and the recovery unit.

The Windows Operating system contains the two options given but not the floppy Recovery. The existing system normally recovers the files in the floppy disk when it is unusable while we are trying to make the whole floppy reusable. The Partition information is not available in any tool along with the floppy tool. This option along with the option for saving and restoring the Master boot record is very useful. The existing tool is the utility available such as Norton Disk Doctor.

Proposed System:

The proposed tool for the recovery of the Floppy disk is very effective. Since it rewrites the Master Boot Record directly it is very effective and efficient in its work. But it is very important to note that the tool and recover the floppy only when the floppy is unusable only through the change in Mbr. The physically affected floppies cannot be recovered through our tool. Since the amount of money is very important. We are trying our level best to recover a floppy ot ensure its amount affordable. Our tool is the one, which can perform all the available options to the floppy such as the scanner, formatter and the recovery unit.

The Windows Operating system contains the two options given but not the floppy Recovery. The existing system normally recovers the files in the floppy disk when it is unusable while we are trying to make the whole floppy reusable. The Partition information is not available in our tool along with the floppy tool. This option along with the option for saving and restoring the Master boot record is very useful.

Since every boot virus affects the Master boot record ,it is always very safe to keep a hard copy of the Master Boot Record .when any virus attacks the harddisk, it is very easy to rewrite the Existing one with the backup. The Partition information is very useful in determining
The partition information in detail. Since it clearly specifies the starting and ending locations with reference to tracks, sectors and the cylinders

2.1 System Requirements

2.1.1 Hardware Specification

Processor : Pentium II

Memory : 64 MB Ram

Display : Color Monitor

Hardware : 4 GB HDD & 1.5 " FDD

2.1.2 Software Specification

Operating System : Windows 9x and Dos 5.0

2.1.3 Developed in:

- Turbo C 3.0

2.1.4. Reasons for choice of Software

Turbo C 3.0

Turbo C was developed at AT & T's Bell laboratories of USA in 1972. C became popular due to its reliability, simplicity and easiness. C is often regarded as problem oriented language and a machine-oriented language.

It is called as a middle level language since it possesses a good programming efficiency (as compared to machine-oriented languages) and good machine efficiency (as compared to problem oriented languages).

C is the best choice for programmers who which to find a development solution for projects which especially deals with involving access of low level resources of the system like hard disk etc..,and involves traversing of interrupts. The communication of hardware devices through c is very quick and effective when compared with the other languages. Since it is a low level language it can communicate with the peripherals through the interrupts, which are the effective way to interact .C offers faster execution and better result.

It is a very powerful language that it can easily handle even a very complicated problem with ease. The effective pointers and interrupts makes this possible. So we have selected C as our developin tool to make system side interactions in a better way.

SYSTEM STUDY

The system study will give the, detailed study of all the peripheral devices that we are using. The important things handled here are the detailed geometry of the Floppy and the Harddisk. After this chapter, there will be a clear idea about he structure and the functions of both the Floppy and the harddisk.

Bios

Bios is the heart of the computer hardware. the bios acts as a Interface between the operating system and the hardware devices. The bios communicates with the hardware only through Interrupt. There is interrupt assigned for each device in the Bios.

Interrupt

Each and every computer peripheral has its own IRQ number. The communication of devices is possible only through the interrupts. Each device contains a specific IRQ number assigned by the Bios. The Interrupt Vector table holds all the details about address of each interrupt code in the memory. when any interrupt is called the respective address is passed to the bios which executes the request as per the priority. There is priority available for each interrupt .

Floppy Disk Geometry

The Floppies are normally used as portable storage devices for carrying small amount of data from one place to the other. The architecture of the floppy is somewhat easier in understanding the normal functions of the floppy. It works according to the principle of Electromagnetism. The floppy contains a circular plastic disk coated with magnetic material.

The Disk is placed safely within the soft materials so that it cannot be affected by external mediums.

When we are formatting a floppy disk, it will create Sectors and tracks. If we are using a 1.44 mb floppy it will create 18 tracks and 80 sectors. We can use the Format option given in our tool to format the floppy. The floppies are divided into tracks, sectors, clusters and so on.

These factors are very useful in calculating and accessing the exact data present in the Floppy. The data present in the floppy is normally specified in reference with the tracks and the sectors only. Because it is easy in tracking, the data present in the floppy exactly.

The floppy contains the following:

1. Track
2. Sector
3. Cluster

Track

The track in the floppy is the concentric circles present in the circular disk. The number of tracks in the floppy disk varies according to the total capacity of the floppy disk. Since there are various sizes available in the floppy there are also various sizes available for the tracks in the floppy disk.

Sector

Each track is divided further into sectors for easy calculation. The sectors are the smallest unit of measurement in the floppy disk. Each sector is about 512 bytes long.

Cluster

The clusters are the collection of sectors.the number of sectors per cluster varies according to the file system we are handling and the type of formatting used. Various file systems uses various types of file handling, its functionality differs from one type to the other thereby making the difference. The clusters are normally used by the Fat (File Allocation Table) for referring the data present in the floppy disk. The fat table only contains the details about the number of used and unused clusters in the floppy. The floppy contains an index hole to enable the read/write head for performing the operation required. There is also a write-Protect latch available for Locking and unlocking the option of Writing to the floppy disk. When this latch is in locked state it is impossible to write anything to the floppy disk.

The scanning option of the floppy tracks each sector one by one and checks the state of the sector. It will count the number of bad sectors present in the disk, and records the details about all the other information such as free space, Number of sector/cluster etc.

HardDisk Geometry

First start by getting an understanding of the way that a harddisk is laid out and the way that a computer accesses a hard disk.

- 8 bit = 1 Byte
- 1024 Byte = 1 KB
- 1024 KB = 1 MB
- 1024 MB = 1 GB

Firstly, there is the real disk geometry and then there is the "real real" disk geometry! What does this mean? Basically, a hard disk contains a set of electromagnetic platters stacked on top of one another (a bit like many CDs in a stack) with a narrow gap between each platter. Each platter is usually double-sided, although for the purposes of this explanation, this makes no difference. Unlike an old vinyl record, each platter has a set of concentric rings that contain the data. Each ring is then split into segments.

Remember that each platter is double sided, and each side has it's own read/write head (a bit like a vinyl record being read simultaneously on both sides by a needle). So, if you've got 4 platters, you've probably got 8 heads). Now, for the purposes of getting the terminology right, it's the number of heads which is important, rather than the number of platters.

So, each head reads from one of the concentric rings (technically called a "track") on the cylinder. Interestingly, all the heads move at the same time and are positioned to read or write to the same track on their respective platter. So, if head 3 is positioned to read from track 77, head 7 will also be positioned to read from track 77. Now, what do you get if you stack a load of tracks on top of one another? A cylinder! Therefore, we might say that head 3 is positioned to read from cylinder 77 (which implies that head 7 is also positioned to read from cylinder 77).

Finally, each track is split into small segments. Each segment is called a "sector". Now that we have the description, let's start to use the correct terminology.

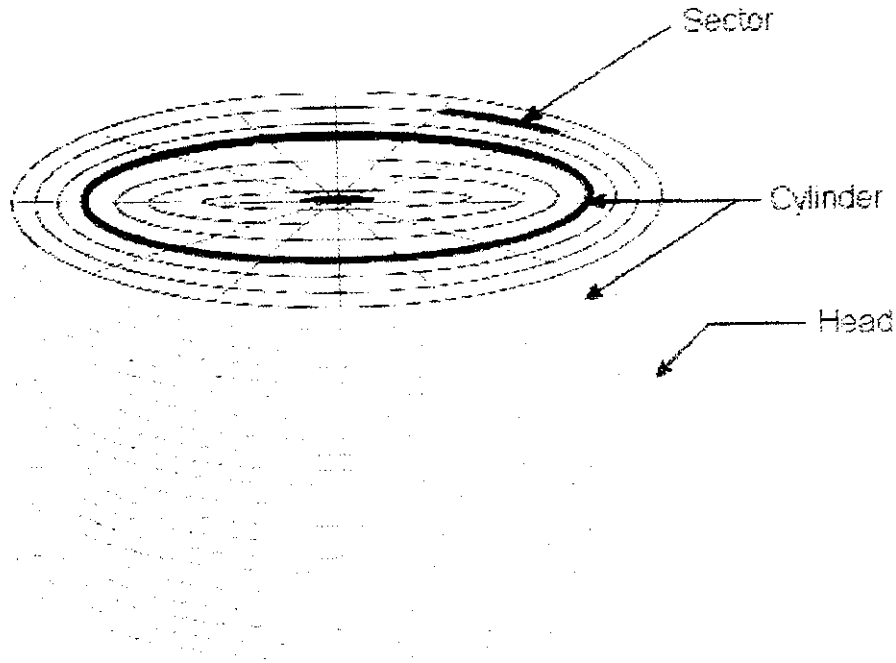
Heads on a Platter	: Head
A stack of Rings or "Tracks"	: Cylinder
Segment	: Sector

If we wanted to access one particular sector, we could reference it by specifying which head it was on, which cylinder it was on and finally which sector it was on. That would then uniquely identify the sector that we wanted to access.

How the physical disk is accessed

Knowing which sector we want to access is only part of the story. How do we actually get the disk reader to the right position physically on the disk? Somewhere in the picture, there must be some software that "knows" that if, for example, you want to access cylinder 23 then you have to move the head reader 2.5434567cm from the edge of the platter. Fortunately, such code comes with the hard disk itself. It is called the "disk controller" and it allows us to specify only the Cylinder, Head and Sector that we want to access. The disk controller calculates where the data is physically on the disk and hands the data back to us (or more accurately a pointer in memory to the data).

If that were the end of the story, we'd have a recipe for disaster. For example, one manufacturer could produce a hard disk with 125336566 cylinders, 2 heads and 198 sectors per cylinder. Another manufacturer could produce a disk with 1 cylinder, 2624626 heads and 1 sector per head (though a strange disk indeed that would be!). Therefore, a set of standards called the ATA (AT-Attachment) came into being in 1989 that effectively put a boundary on what was possible in terms of cylinder, head and sector numbers



The disk manufacturers agreed to the standards, but continued to produce disks that didn't necessarily meet those specifications! (This isn't actually surprising or bad because the IDE specification allows hard disks to have more sectors on the outer tracks than the inner tracks. This is called "Zone Bit Recording" or "Zone Density Recording".) However, they compensated for the fact by getting the disk controller to "advertise" a set of parameters that met the standards, and then translated the sector requests into ones that were appropriate for the physical disk.

Therefore, only the disk controller "knows" the real real disk geometry. What most people call the real disk geometry is actually the geometry that the disk controller "pretends" to exist.

Real Geometry --> Disk Controller --> "Real real" geometry --> Physical disk

Because we don't care what the "real real" geometry is, we can concentrate on the real geometry from now on!

The Standards

There are many standards for disk geometry, but the one pertinent to this discussion is the ATA standard. (Note: For details on individual variants of the ATA standard, such as ATA-5t:

1. The cylinder number should be represented as 16 binary digits.
2. The head number should be representable as 4 binary digits.
3. The sector number should be representable as 8 binary digits.
4. There are 512 bytes per sector.

This means that the maximum specification of an IDE disk that conforms to the ATA standard is as follows.

Cylinders	65536 (Numbered 0 to 65535)
Heads	16 (Numbered 0 to 15)
Sectors	256 (Numbered 0 to 255)
Bytes per sector	512
Total space	128 GB

Using this method the total space of the harddisk is calculated by the formula=c*h*s*bs

C=Number of cylinders in the harddisk

H=Number of heads in the harddisk

S=Number of sectors in the harddisk.

BS=Bytes per Sector

Normally the manufacturer of the harddisk specifies these details. So there is difference in harddisk from one manufacturer to the other one. It is not sure that a 20 GB harddisk of two different companies should contain the same number of cylinders and sides. It is up to the Manufacturer to determine the values.

Master Boot Record

The harddisk contains the Mbr(Master Boot Record) in the Cylinder 0, head 0, sector 1. The Mater Boot Record is the one which contains all the required details about the harddisk. Such as the partition details and the fat details etc.

The Master Boot Record is about 512 bytes long. The first 446 bytes is allocated for the normal boot code and the next 64 bytes for the partition table.

The partition table is about 64 bytes long which can accommodate 4 partition entries. The final 2 bytes is 55,AA which is common for all the Master Boot Record.

Finally the Master Boot Record is described as

Boot Sector - 446 bytes

Partition Table - 64 bytes

55, AA - 2 bytes

Totally - 512 bytes

Offset	Length	Content
0	446	MBR
446	16	1 PT
462	16	2 PT
478	16	3 PT
494	16	4 PT
510	2	55,AA

The partition table is present only in the mater boot record of the harddisk. The partition table contains the details about the size and the characteristics of each partition in the harddisk. The size of each partition, File system , geometric location(starting sector, track, ending cylinder, ending track etc).

The Partition table is only 16 bytes long.

462		Bootable Pt
463		Starting head
464		Starting Cylinder
465		Starting Sector

466	1	System ID
467	1	End Sector
468	1	End Head
469	1	End Cylinder
470	4	Relative sectors
474	4	Total Size

When the bootable partition is set to 80, that partition is termed as the bootable partition .The system id specifies the type of the operating system used. The Display of partition information seeks information only from the Pt for reading the partitions available in the given harddisk.

The important Details about the Master Boot Record are,

- The BIOS Parameter Block (BPB) starts at offset 0.
- The boot sector program starts at offset 3e.
- The DOS hidden file names start at offset 1e6.
- The boot sector signature is at offset 1fe.

The first 62 bytes of the boot sector contains the BPB. Following is the details available in the Bpb. They are,

```
db JMP instruction     at 7c00  size  2 = eb3c
db NOP instruction        7c02        1   90
db OEMname                7c03        8   'MSDOS5.0'
dw bytesPerSector         7c0b        2   0200
db sectPerCluster         7c0d        1   01
dw reservedSectors        7c0e        2   0001
db numFAT                 7c10        1   02
dw numRootDirEntries      7c11        2   00e0
dw numSectors             7c13        2   0b40 (ignore numSectorsHuge)

db mediaType              7c15        1   f0
```

dw numFATsectors	7c16	2	0009
dw sectorsPerTrack	7c18	2	0012
dw numHeads	7c1a	2	0002
dd numHiddenSectors	7c1c	4	00000000
dd numSectorsHuge	7c20	4	00000000
db driveNum	7c24	1	00
db reserved	7c25	1	00
db signature	7c26	1	29
dd volumeID	7c27	4	5a541826
db volumeLabel	7c2b	11	'NO NAME '
db fileSysType	7c36	8	'FAT12 '

The Entire Master Boot Record will look like this when opened under a Hex Editor. It's size is about 512 bytes.

The Master Boot Record in hex and ascii

```
OFFSET 0 1 2 3  4 5 6 7  8 9 A B  C D E F  "0123456789ABCDEF"
000000 eb3c904d 53444f53 352e3000 02010100 ".<.MSDOS5.0......"
000010 02e00040 0bf00900 12000200 00000000 "...@............"
000020 00000000 0000295a 5418264e 4f204e41 "......)ZT.&NO NA"
000030 4d452020 20204641 54313220 2020fa33 "ME    FAT12   .3"
000040 c08ed0bc 007c1607 bb780036 c5371e56 ".....|...x.6.7.V"
000050 1653bf3e 7cb90b00 fcf3a406 1fc645fe ".S.>|........E."
000060 0f8b0e18 7c884df9 8947c2c7 078e7cfb "....|.M..G....|."
000070 cd137279 33c03906 137c7408 8b0e137c "..ry3.9..|t....|"
000080 890e207c a0107cf7 26167c03 061c7c13 ".. |..|.&.|...|."
000090 161e7c03 060e7c83 d200a350 7a891652 "..|...|....Pz..R"
0000a0 7ca3497c 89164b7c b82000f7 261176bb "|.I|..K|. ..&.v."
0000b0 1e0b7c03 c348f7f3 0106497c 831e4b7c "..|..H....I|..K|"
0000c0 00bb0005 8b16527c a1507ce8 92007219 "......R|.P|...r."
0000d0 b001e8ac 0072168b fbb90b00 bee87df3 ".....r........}."
0000e0 a6750a8d 7f20b90b 00f3a674 18be9e7c ".u... .....t...|"
0000f0 e85f0033 c0cd165e 1f8f048f 4402cd19 "._.3...^....D..."
000100 585858eb e88b471a 48488a1e 0d7c32ff "XXX...G.HH...|2."
000110 f7e30306 497c1316 4b7cbb00 07b90300 "....I|..K|......"
000120 505251e8 3a0072d8 b001e854 00595a58 "PRQ.:.r....T.YZX"
000130 72bb0501 0083d200 031e0b7c e2e28a2e "r..........|...."
000140 157c8a16 247c8b1e 497ca14b 7cea0000 ".|..$|..I|.K|...."
000150 7000ac0a c07429b4 0ebb0700 cd10ebf2 "p.....t)........"
000160 3b16187c 7319f736 187cfac2 88164f7a ";..|s..6.|....Oz"
000170 33d2f736 1a7c8816 257ca34d 7cf8c2f9 "3..6.|..%|.M|..."
000180 c3b4028b 164d7cb1 06d2e60a 564f7c8b ".....M|.....VO|."
000190 ca86e98a 16247c8a 36257ccd 13c30d0a "....$|.6%|....."
0001a0 4e6f6e2d 53797374 656d2064 69736b20 "Non-System disk "
0001b0 6f722064 69736b20 65727272 0d0a5265 "or disk error..Re"
0001c0 706c6163 65206120 6e642070 72657373 "place a nd press"
0001d0 20616e79 206b6579 20776865 6e207265 " any key when re"
0001e0 6164790d 0a00494f 20202020 20205359 "ady...IO      SY"
0001f0 534d5344 4f532020 00005953 "SMSDOS   SYS..U."
```

BPB

The BPB stands for the BIOS Parameter Block. This block contains the important details required for the harddisk. These are the details specified by the manufacturer of the harddisk .

These details are accessed by the Bios hence the name BPB . The Boot Sector is the other Part which contains details about the information required for Boot Sequence. The Details available available in the BPB are

1. No of bytesPerSector
2. No of sectPerCluster
3. No of ReservedSectors
4. No of Sectors
5. Media Type
6. No of SectorsPerTrack
7. No of Heads
8. The volume Label
9. The file system Type etc..

The Format:

Normally the Formatting is of two types High-level and Low-level Formatting. When a floppy is formatted under dos, it performs both High and Low-level formatting. The formatter uses a high sensitive algorithm called CRC. The CRC is abbreviated as Cyclic Redundancy Check. It is highly sensitive algorithm , it can track even a very small change in the data present.

Initially the low-level formatting takes place where the algorithm reads each sector and writes the respective details in the trailer of each sector. After completing the operation it rechecks each sectors data and their Trailer details. If it doesn't match, the respective sector will be marked as a Bad Sector. The respective address of the bad sector is stored in the Ram for future reference. After completion of the Recheck the low level format is over.

In the High Level Formatting the sectors will be cleared and the new Fat(File Allocation table) will be written in reference with the location of bad sectors specified in the Ram. It rewrites the Boot Sector and finally the Formatting is over.

The formatting option provided by us is an unconditional formatting where there is no recheck by CRC. It just clears the contents of the floppy drive and rewrites the MBR. The unconditional floppy does not keep hands on the fat table.

System Design

Module Design

The main modules in this project is three, They are

Scanner

To Scan a given Floppy disk

Formatter

To Format a given floppy Disk

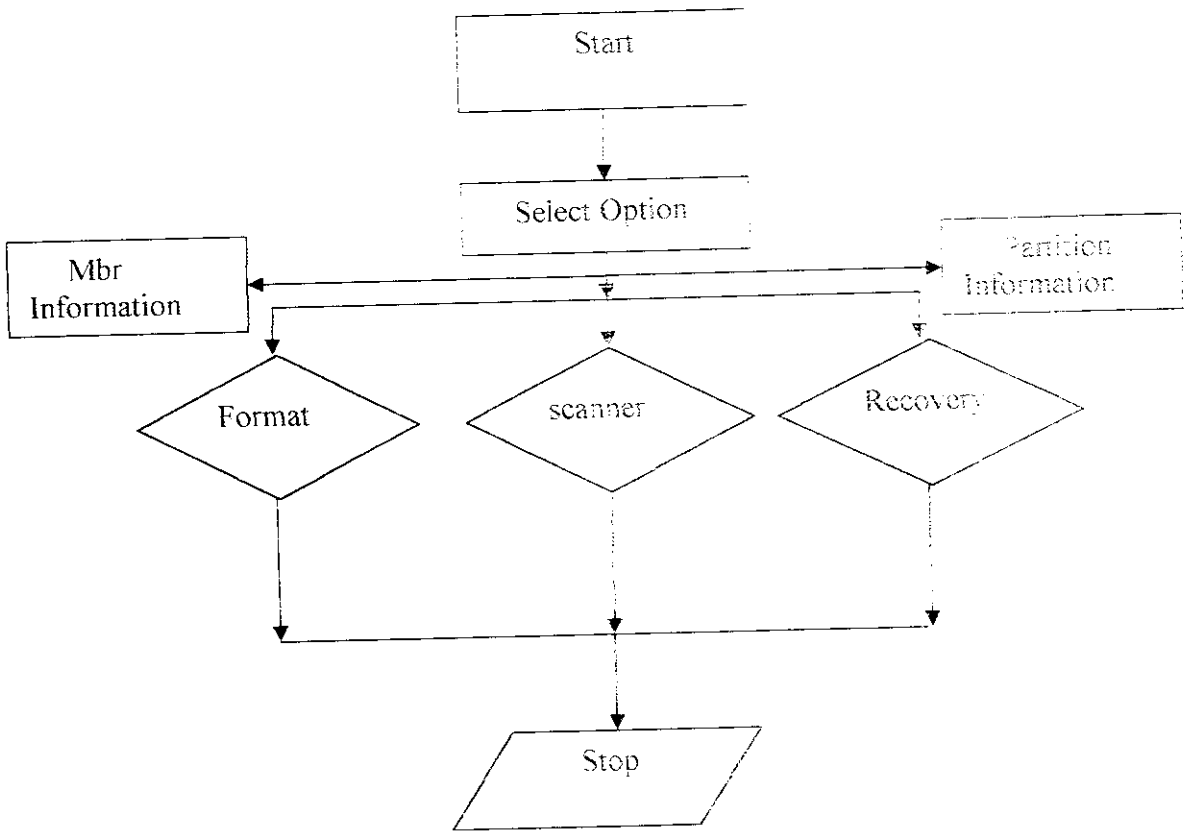
Recovery

To Recover a unusable floppy.

The additional option for saving and restoring the mbr of the floppy are provided. The partition information is also displayed for the given harddisk. The functionality of the tool can be summarized as follows. The user has to choose the option required, when he chooses the formatter option the user will be prompted for providing the floppy to format. When the floppy is entered, the formatter will start formatting it. Here it is very important to note that the formatter will provide only unconditional formatting. When the option is to scan the floppy, the floppy will be scanned and the report will be given which contains details about the number of bad sectors, free space etc. The recovery unit will first check for the first track and continues its operations when it is fair to continue.

The System Flow Diagram

The floppy disk is the source for our tool.



Partition Information

To Display the Partition Information of the Harddisk.

Master Boot Recovery

To Save and Restore the Master Boot Record.

User Interface:

The user interface for our tool is developed in C. We tried our level best to give a good interface to the user. While developing the interface we have used many graphics functions available in C. Some of the graphics function is discussed here,

Rectangle

Line

Floodfill

Setcolor

Settextstyle

Initgraph etc.,

Rectangle

This function is used to draw a rectangle with the given arguments

Line

This function is used to draw the line in the screen

Floodfill

This function is used to fill the screen with the given color

Setcolor

This function is used to set the color for the text

Settextstyle

This function is used to set the style for the text. We can also change the font and the size of the font.

Initgraph

This fuction is used to initialize the graphical drivers in C

There is no mouse interface available in our tool. The reason is that we can use the same tool in Dos version also. So in order to fulfill the platform dependency we are not providing any Mouse Interface in our tool. So it is compatible with both the Windows and the Dos versions. The Interface is up to the level of User-friendly in the view of us . The tool is fully focused in the operations related with the floppy disk.

Implementation

The tool has been developed in C. We have used various functions available in C for making this project comes true. Since C is a powerful language we can perform even complex operations in a easier way using C.C is a machine-dependent language hence the communication with the machines are very easier in C.

The Implementation of the various functions available in C have made this project successful. The flexibility of c have made our job somewhat easier in implementing even complicated operations like formatting and reading the mbr available in the floppy or Harddisk.

The Major functions used by us in this project are,

Functions

Biosdisk

Absread

Abswrite

Ultoa

Memcpy

Memset etc.,

Interrupt Services Used

intdos

intdosx

int86x

int86

Biosdisk

The bios disk is used to read/write any sector of the harddisk as well as the floppy disk. It uses Interrupt 13 for performing the disk-oriented operations directly. The arguments used in the Biosdisk is

Command

Drive

Head

Track

Sector

Nsect

Buffer

Command

It is used to specify whether we have to read or write a sector. Other options like verify, status are also available with the command

0 - Reset

1 - Status

2 - Read

3 - Write

Drive

It is normally used to specify the drive to which the requested operation has to be performed. Normally the drives are numbered as follows,

0 – A drive

1 - B drive

0x80 - Harddisk 1

0x81 - Harddisk 2 and so on

Head

It is used to specify the head number from which the requested operation gas to be performed.

Track

It is used to specify the track number, where the required sector is located.

Sector

It is used to specify the exact sector number from which the requested operation has to be performed.

Nsect

It is used to specify the total number of sectors from which the requested operation gas to be performed.

Buffer

It is used to store the contents, when the Biosdisk performs a read operation.

It acts as a buffer.

Absread

The absread operation is used to read any sector of the disk directly. It uses Interrupt 25 for reading the sector of a disk.

The arguments used here are

Drive

Command

Sector

buffer

These arguments are same as the arguments available in the bios disk.

Abswrite

The Abswrite operation is used to read any sector of the disk directly. It uses Interrupt 26 for writing in any sector of a disk.The arguments used here are

Command, Sector, Drive and the Buffer

Ultoa

This function is used to convert an unsigned long to String. It is very useful in handling large amount of integer values . For example the total number of sectors in a harddisk will be very high etc..

Memset

This function is used to allocate the memory for the given n bytes.

Memcpy

This function is used to copy the memory to a given string

Apart from using functions we have also handled various interrupt Services

Intdosx and Intdos

These are dos interrupt services which uses Interrupt 21 to handle the command specified. Using this we can execute any command in the 8086 processor. These functions use a carry flag if it is set, there is some error in execution

Int86x and Int86

Using this interrupts we can invoke a software interrupt in the 8086 processor. These contain a register to store the values available after execution. In addition to this we have used many graphic functions which are explained in the User interface Section.

Testing

Any project is not complete without testing. The testing is the very important phase in the project, which needs more time and resources. Since we are dealing with very fragile and costly things, we have to be very careful while testing since a small error may lead to high loss.

We paid so much attention while testing our project to ensure safety to Both the system and devices. Since we are directly accessing the interrupts, which communicates with harddisk and floppies. So great care should be taken unless it will lead to total system crash or harddisk crash.

The testing of the floppy Scanning is the easiest one to check, since it handles only floppy. Similarly the testing of the Formatter is also very easy, Since it handles very cheap resources.

The Floppy recovery option needs high degree of testing since the floppies with physical damage cannot be recovered through our tool. So we are very keen in avoiding the floppies with physical damages. The floppies are also verified with the normal windows after recovery for ensuring the safety.

Problems Faced

In the recovery and restoring option of the Master Boot Record, there is so much care and money required while testing. Since The Master Boot Record is the key in the harddisk, if there is any error in the Master Boot Record. It will affect the Entire harddisk which cannot be reused. So we used harddisk of very low storage capacity such as 400MB and 261 MB for our Testing. In the Tool it will restore the Master Boot Record and save it in a file.

First Attempt

In out first attempt the Master Boot Record cannot retrieved be as such. There is some variations in the size of the Master Boot Record retrieved. So while restoring the Master Boot Record the normal Master Boot Record is retrieved with some lack of code .So the harddisk as a whole become unusable, Since it gets affected in the first track.

Second Attempt

In out Second attempt the Master Boot Record is retrieved as such. There is some variations in the size of the Master Boot Record written. So while restoring the Master Boot Record the normal Master Boot Record is retrieved with some lack of code .So the harddisk as a whole become unusable, Since it gets affected in the first track.

Future Enhancement

- The Scanning operation for the harddisk is also in our mind it will be added to our tool in the future.

- User interfaces play a major role in determining the response of the project. So the user interface screen should be customized to provide much more sophistication aiming at better design and end-user's comfort.

- The Option for Scanning the Hard disk is also under consideration for the future

- The idea of resizing a existing partition without affecting the data present is to be added in our tool in the future.

Conclusion

"Necessity is the Mother of Invention" by Thomas Alva Edison makes up our Project a successful one.

"Disk Utilities" is implemented for the windows as well as dos perfectly without any bugs up to our level. Transparency, Robustness and Capacity, which are the core objectives of the project, has been achieved.

The analysis phase provided a better understanding of the various devices and their implementation. The existing tools are tested and the drawbacks were found. After a detailed study, the proposed product's features were designed.

During the course of the project various risks like computer failure leading to delays, data losses were identified and timely measures were taken to minimize them. The risk that computer failure would cause delays in the project was managed by ensuring more than one system at a time. Similarly risk of data loss was minimized by taking regular backups. Thus, by eliminating the risk of delays and data loss, the schedule was kept on track. The risk of system failures due to errors in the codes was handled with great care to avoid very high loss

Bibliography

1. www.ata-atapi.com

2. www.teleport.com

3. www.symantec.com

4.www.zelepes.com

5. www.grsoftware.net

6. www.seagate.com

7.www.Funducode.com

Text Books:

1. Writing tsr through c by yashavant P.Kanetkar

2. C with Assembly language by Steven Holzner

3. C Odyssey Dos

4. C Odyssey Advanced Dos

5. Let us C by yashavant P.Kanetkar

Appendix A

System Requirements Specification

PROJECT OVERVIEW

Floppies are generally used as a portable storage device for carrying small amount of data from one place to the other with ease. The Floppies are working on the principle of magnetism. The head of the Floppy Disk Drive records the data through Electromagnetic transfer of bits between the disk surface and the read/write head of the floppy disk drive.

The floppy is divided into tracks, sectors and clusters for the sake of convenience. .The floppy contains the MASTER BOOT RECORD at the Track 0,Sector 0. It contains the details about the contents of the floppy, space availability and all the other necessary details.

The first track of the Floppy if contains any errors, the floppy as a whole will be unusable. So steps are taken in our tool to make the floppy reusable by making some necessary changes in the MBR. So that the floppy will be usable by starting reading from the second track itself.

A scanning option for the floppy and harddisk is provided. This scanning will generate a report about the bad sectors, sectors per cluster, Free space, total clusters, total sectors, used space, number of tracks, number of sectors and so on.

The partitioning of the harddisk is dividing the harddisk into separate logical drives. Since large capacity harddisk is not advisable to use as it is. So the disk is partitioned into different logical drives to increase the performance of the disk.

Hardware requirement

* PENTIUM 486 or 386 Processor

* 64 MB RAM

* 4 GB HDD

* 3.5 FDD

Software requirement

* Turbo C++

Module description

Scanning of floppy

Option for scanning the floppy .

The floppy utility

To make the unusable floppy usable

The Floppy Formatter

To Format the given Floppy.

Future Modification:

To Scan the harddisk in addition to the Floppy disk.

The formatter option for the harddisk is also in our mind to complete.

Source:

Most of our references are only through the internet which yields a lot of information. Some of the websites where we got our materials for gathering our information are

Websites:

1. www.ata-atapi.com
2. www.teleport.com
3. www.symantec.com
4. www.zelepes.com
5. www.grsoftware.net
6. www.seagate.com

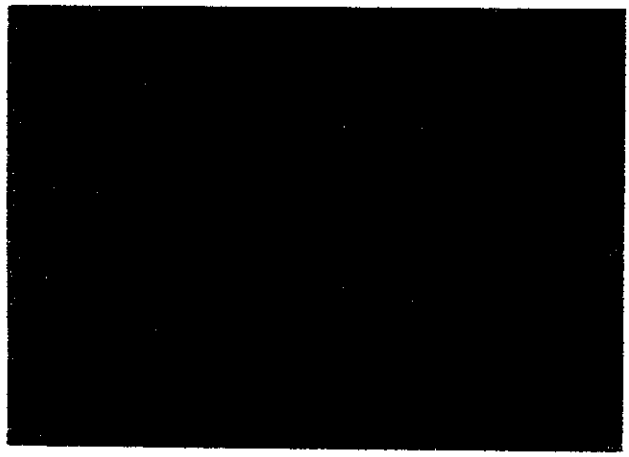
Text Book:

1. Writing tsr through c by yashavant P.Kanetkar
2. C with Assembly language by Steven Holzner
3. C Odyssey Advanced Dos

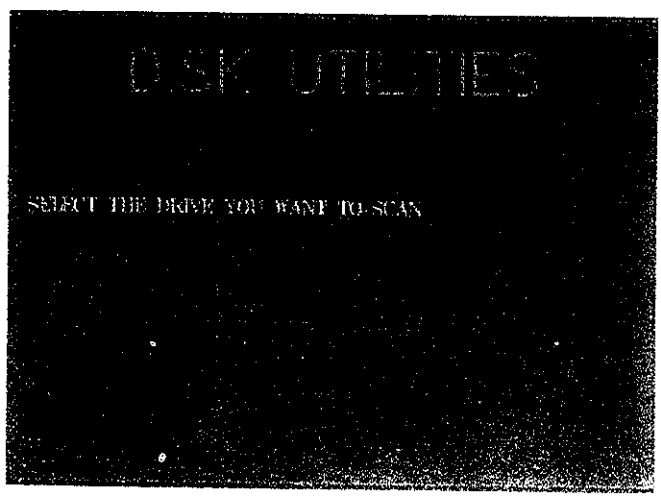
Appendix B

Screen Shots of our Tool

The main screen of our tool ,



The Scanner Screen



Appendix c

Sample Coding

A sample code for scanning the Floppy disk,

```
#include <stdio.h>
#include <string.h>
#include<graphics.h>
#include<conio.h>
#include<stdlib.h>
#include<bios.h>
#include<dos.h>

void main()
{
                    /* request auto detection */
  int gdriver = DETECT, gmode, errorcode;
  char msg[30]="DISK UTILITIES";
  struct diskfree_t free;
  long avail;
  float mb;
 struct fatinfo diskinfo;
  int count=0,f=0,c=0;
  char m;
  int i,j,e=0,n=0,k;
   float x;
```

```
    char *b[4096];
    fflush(stdin);                          //to clean the buffer stream
    clrscr();
                    /* to initialize graphic mode */
initgraph(&gdriver, &gmode, "d:\\tc\\tc\\bin");

/* to read  the result of initialization */
errorcode = graphresult();

if (errorcode != grOk)  /* an error occurred */
{
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      printf("Press any key to halt:");
      getch();
      exit(1);                          // return with error code
}
main:
clrscr();
setfillstyle(SOLID_FILL,12);
floodfill(2,2,4);
setcolor(9);                            //set border color
rectangle(3,3,638,477);                 //to draw border lines
rectangle(6,6,636,475);                 //to draw border lines
setcolor(9);                            //set line color
line(6,90,637,90);                      //drawing line
line(6,92,637,92);                      //drawing line
```

```
setcolor(15);                                   //set title color
settextstyle(3,0,7);                            //setting text style
//settextstyle(5,0,1);
outtextxy(120,15,msg);                          //display title
settextstyle(5,0,1);                            //by title
setcolor(0);

settextstyle(1,0,2);
setcolor(14);
outtextxy(20,180,"SELECT THE DRIVE YOU WANT TO SCAN");

setcolor(1);
settextstyle(1,0,2);
outtextxy(40,230,"1.Floppy");
outtextxy(40,270,"2.Exit");
settextstyle(1,0,1);
setcolor(0);
outtextxy(40,300,"Press 1 to scan or any other key to exit");
c=getch();                                      //to get the user's response
 if(c==49)
 {
        getfat(1,&diskinfo);
        //to get disk details
        settextstyle(0,0,1);
        setcolor(0);
        outtextxy(40,340,"Scanning Under progress Please wait.... ");
```

```
for(i=0;i<79;i++)      //10
 {
        for(j=0;j<2;j++) //10
                {
               rectangle(80,360,598,384);  //first rectangle
                for(x=83;x<=(7.5*i);x++)        //second one
                {
                setcolor(1);   //color of the status bar
                rectangle(82,362,x,382);//incrementing the values for the
                status bar
                delay(5);
                if(x==608)
                        cleardevice();
                }
                //f=biosdisk(2,0,1,i,j,j+1,b);  //to read the sectors
                        switch(f)
                        {
                        case 0x04:
                                //printf("\nSector not found");
                                n++;
                        break;
                        case 0x0a:
                                printf("\n\t%5d",n);
                                n=0;
                                setcolor(0);
```

```
                count++;
        break;
                default:
                        e++;
                }
        }
}
 cleardevice();
 setfillstyle(SOLID_FILL,14);
 floodfill(151,66,0);
 setcolor(14);                                          //set border color
 setbkcolor(8);
 printf("\n   Number of bad Sectors :%3d",count);
 printf("\n   Sectors per cluster: %5d\n",diskinfo.fi_sclus);
 printf("  Number of clusters:   %5ld\n",(long)diskinfo.fi_nclus);
 printf("   Bytes per sector:     %5d\n",diskinfo.fi_bysec);
 if (_dos_getdiskfree(1, &free) != 0)
        {
                printf("Error in _dos_getdiskfree() call\n");
                exit(1);
        }
avail = (long) free.avail_clusters       //to calculate
                * (long) free.bytes_per_sector          //the free
                * (long) free.sectors_per_cluster;      //space available
printf("   Space available in bytes: %6ld", avail);
mb=(float)(avail/1024);
```

```
mb=(float)mb/1024;
printf("\n    Free Space: %0.2fMB",mb);
rectangle(3,3,320,130);                    //to draw border lines
rectangle(6,6,318,128);
settextstyle(6,0,2);
outtextxy(90,200,"Press any Key to main menu or Esc to exit");
  m=getch();
   if(m=='x'||m=='X')
                system("inter.exe");
        else goto main;
                getch();
}
//closegraph();
 //  return 0;
}
```