



# EXT2READ

Use, Design and Implementation

By: Manish Regmi  
regmi.manish@gmail.com

# What is Ext2read?

- **Ext2read** is a set of utilities which can be used to read/write files and folders of an Ext2/Ext3 Partitions.
- For now there are two programs, `ext2sh` and `ext2explore`.
- `Ext2sh` is a shell like utility from where you can list, change directory and copy `ext2/ext3` files.
- `Ext2explore` is an explorer like utility to display and read `ext2/ext3` files.

# How does it work?

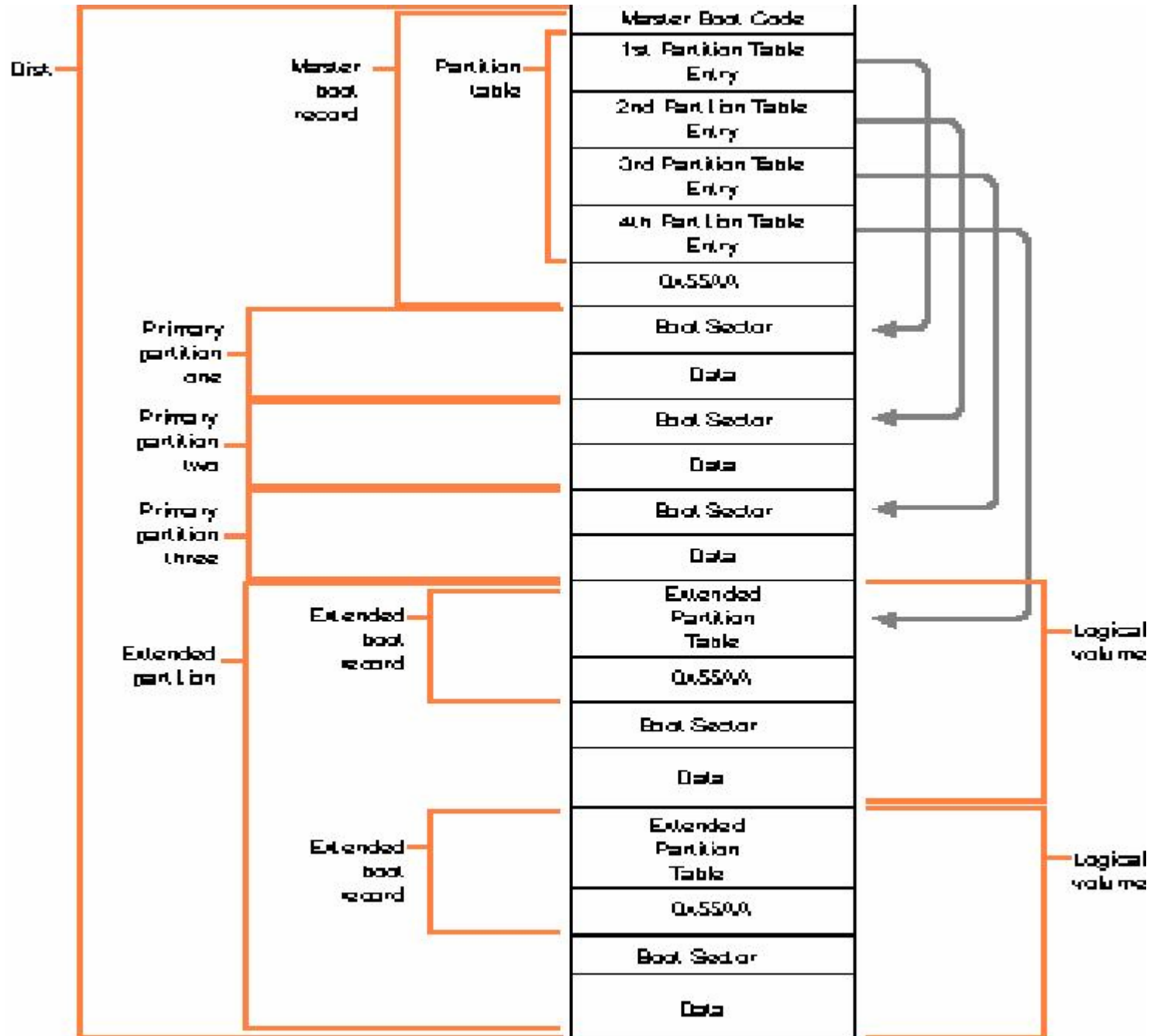
- When started it scans the System for attached disks.
- It then scans each attached disk for partitions.
- When found it shows the list of ext2/ext3 partitions.
- When user chooses the partitions it shows the contents of root folder.

# Inside partition table

- Partition related info is stored in MBR (master boot record).
- It resides in 1<sup>st</sup> sector of the disk.
- First 446 bytes is the MBR code.
- Follows a table of 4 entries called partition table entry.
- Last two bytes are the Signatures which is always 0x55AA.
- Each entry is 16 bytes.
- Contains boot indicator, system id, starting and ending CHS, relative sector and total sector.

# Linked List

- So, there can be only four partitions and they are called primary partitions.
- A technique is used to support more than four partitions.
- Among the four primary partitions one can be extended partition.
- Instead of pointing to a partition it points to another table.
- The table points to a partition called logical partition.
- The logical partitions are linked as a linked list.



# Inside Ext2 File System

- File System allows users to organize, manipulate and access different files.
- File is a container for data.
- There are special types of files called directories which contains other files.
- UNIX has other kinds of files like device files, socket files etc.
- Files has attributes describing size, type, owner, timestamps etc.
- The OS gives system call interface and some utilities to access files.

## System (contd...)

- Originally Linux used Minix File system.
- Later on April 1992, a new FS was designed called “Extended File System” for Linux 0.96c.
- It had some limitations so two new FS was proposed.
- Xia FS based on Minix and Ext2 based on Extended File system.
- EXT2 later became the de-facto standard for Linux.

## System (contd...)

- Shares same concepts of file systems of other UNIX.
- File system structured in hierarchal tree.
- Directory is a special file and contains other files and sub-directories.
- Each file has an associated structure called Inode.
- Each file has a unique number called Inode number.
- Devices are also accessed through files.
- File system related info is stored in Super Block.

## System (contd...)

- Inspired by BSD's FFS.
- FS divided into several Block Groups.

Boot Sector	Block Group 1	Block Group 2	...	Block Group N
-------------	---------------	---------------	-----	---------------

- Each Block group contains redundant copy of Super Block and Group descriptor.

Super Block	Group Descriptor	Block Bitmap	Inode Bitmap	Inode Table	Data Blocks
-------------	------------------	--------------	--------------	-------------	-------------

# System (contd...)

- SuperBlock
  - The crucial information of File System is stored.
  - Only SuperBlock of group 1 is used.
  - SB contains information like
    - Total number of inodes
    - Total blocks, number of blocks reserved for root.
    - No. of free inodes and blocks.
    - No of blocks in a group.
    - Magic number (always 0xEF53)
    - Volume name.
    - Block size.

## System (contd...)

- The super block is followed by Block group table.
- It is a table of structures called Group Descriptors.
- Stores info on each group in a Partition.
- Contains information like:
  - Block no of block bitmap.
  - Block no of inode bitmap.
  - Block no of first block of inode table.
  - No of free blocks and free inodes.
  - No of inodes allocated to directories.

## System (contd...)

- **Block Bitmap**
  - A bitmap to identify which block is used and which one is free.
  - A set bit means used block and clear bit means free block.
- **Inode Bitmap.**
  - identifies which inode is used and which one is free.
  - A set bit means used inode and clear bit means free inode.

## System (contd...)

- Inode table is a table of Inode structures.
- There is a separate inode for each file.
- Each file is identified by an inode no.
- The inode no. is not stored in inode structure but in a directory.
- Inode structure contains:
  - Mode defining permission and file type.
  - Size and timestamps.
  - Pointer to data blocks.
  - Uid, gid.

## System (contd...)

- Data block addressing is efficient and suitable for both large and small files.
- Addressing info is stored in `i_block` array of 15 elements.

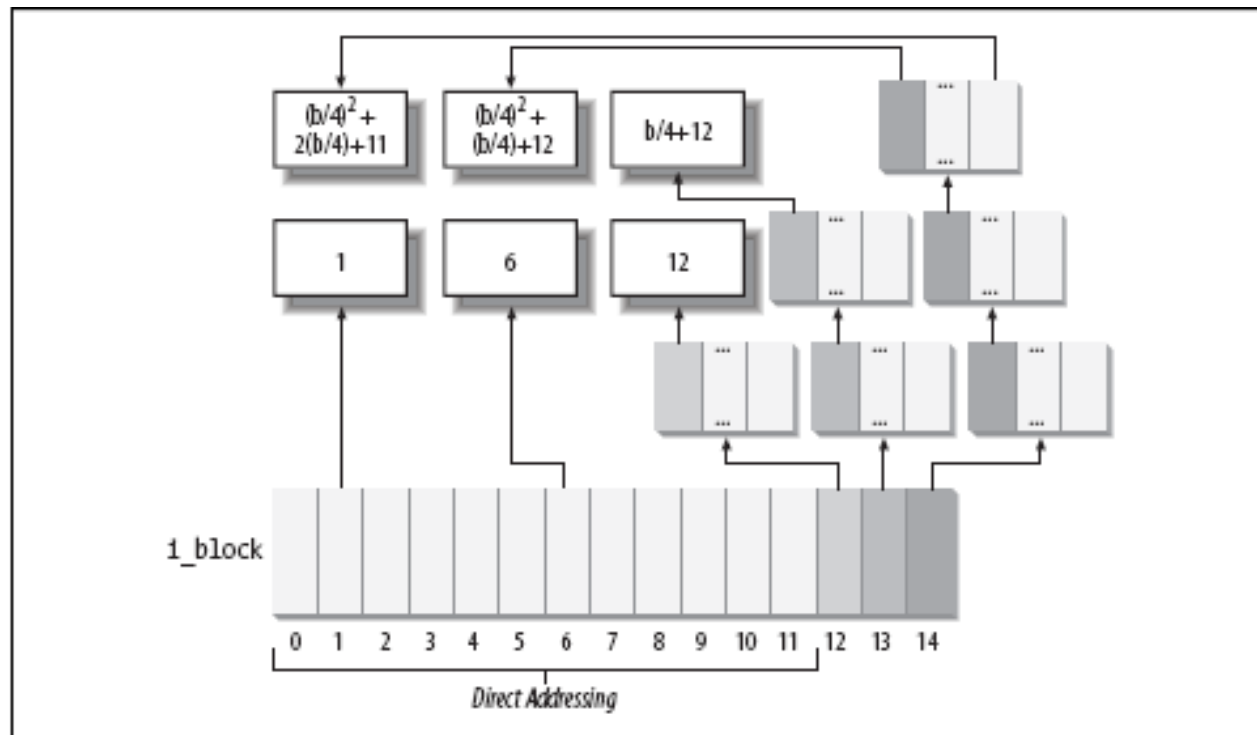


Figure 17-5. Data structures used to address the file's data blocks

## System (contd...)

- Data blocks for directory contains list of directory structures.
- The structure contains info on file or sub-directory.
- The structure contains:
  - Inode number (32 bit)
  - Record length. (16 bit)
  - Name length (8 bit)
  - File type
  - File name (255 characters)

# How to read disk using C/C++?

- In UNIX/LINUX:
  - Everything is a file concept.
  - Linux = /dev/hdx, solaris = /dev/rdisk/c0dxp0
  - use file handling c/c++ api.
  - Open(), fopen(), fstream etc
- In windows:
  - Also represents devices as files calls symbolic links in win32.
  - \\.\physicaldrive0, \\.\physicaldrive1 etc.
  - Use Win32 Api, CreateFile(), ReadFile() etc.

# How Ext2read Works?

- Scans the system for no of disks attached.
- Scans each disk for partitions.
- Asks the user to select Ext2 partition.
- Reads the superblock, group descriptor table.
- Reads the inode no 2.
- It is always a directory.
- Lists the files and sub-directory.
- (and so on)

# What is Ext3?

- A new version of extended File System.
- 100% backward compatible with ext2.
- More features more robust.
- A journaling file system.
  - All operations to the filesystem is first journaled (i.e stored in a log entry).
  - After journaling it is committed to FS.
  - If there is a power failure during a FS operation, recovery can be done through journal.

# What's Next?

- Kernel Mode Driver for windows.
  - Requires DDK and IFS kit.
  - Trying to use mingw and bo-barten's ntifs.h to develop a driver.
- FUSE for windows.
  - Put file system driver to user space.
  - Requires a kernel module which acts as a proxy.
  - It passes a FS related I/O request to user space server.



**Thank You.**

Questions?