Introduction to the Minix File System

This is a brief introduction to the Minix file system compiled by Scott D. Heavner (**sdh@po.cwru.edu**). Please direct any comments or corrections back to the author. You are free to distribute this document to anyone in any form as long as this notice remains intact.

1 Definitions

(All are probably relevant to this document only.)

inode: Stores all the information about a file except its name.

block: A unit of size which is determined by the medium or programmer. For example, most devices use 1024 byte (1k) blocks, including hard disks and floppy disks (stored in `BLOCK_SIZE`).

zone: probably used interchangeably with block in this document. A zone is the part of the disk where the files data exists.

super block: the first block on a disk which contains information about the type and size of the file system.

2 Physical Layout

Minix FS Layout

	Size (Blocks)	Description
Blank Block	1	Reserved for partition boot code.
Super Block	1	Info about the file system.
Inode Map	#Inodes/BLOCK_SIZE	Keeps track of used/unused inodes.
Zone Map	#Data Zones/BLOCK_SIZE	Keeps track of used/unused zones.
Inode Table	$((32 or 16) \times$ #Inodes)/BLOCK_SIZE	Store info about files/devices.
Data Zones	Big	File/Directory contents.

3 Inodes

From <linux/minix.h> (Isn't source code wonderful?)

```
struct minix_inode {
        unsigned short i_mode;
        unsigned short i_uid;
        unsigned long i_size;
        unsigned long i_time;
        unsigned char i_gid;
        unsigned char i_nlinks;
        unsigned short i_zone[9];
};
```

Minix Inode Entry																
	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0x00	MODE		UID		SIZE				TIME				GID	LINKS	ZONE 0	
0x10	ZONE 1		ZONE 2		ZONE 3		ZONE 4		ZONE 5		ZONE 6		ZONE 7		ZONE 8	

The inode contains all the important information about a file, except its name. It contains the file permissions, file type, user, group, size, modification time, number of links, and the location and order of all the blocks in the file. All values are stored in low byte – high byte order. The maximum number of links is 250 (`MINIX_LINK_MAX`). Notice that the group number is limited to one byte whereas <gnu/types.h> defines it as short.

Now, for the zones. The first seven zone pointers (0-6) are point to file data. They are two byte numbers which point to a BLOCK on the disk which contains the file's data. The eighth is a pointer to an indirect block; this block continues the tradition of the first seven zone pointers, and contains 512 (`BLOCK_SIZE/2`) zone pointers. The ninth zone pointer in the inode points to a double indirect block. The double indirect block contains 512 pointers to more indirect blocks, each of which points to 512 data zones. Each indirect block adds 1k to the file. It's no big deal, but it's nice that small files (under 8k) don't have this overhead. Technically you can make 262M files ($7 + 512 + 512 \times 512$ k, see also s_max_size in the superblock information), but since the Minix fs uses unsigned shorts for block pointers, it is limited to 64M partitions.

To determine the meaning of the mode entry, consult <linux/types.h> which is included via <sys/types.h>. Below is a list of octal numbers which can be extracted from the mode entry. This information can also be found in the stat(2) and chmod(2) man pages.

```
#define S_IFSOCK 0140000 /* Socket */
#define S_IFLNK  0120000 /* Symbolic Linux */
#define S_IFREG  0100000 /* Regular file */
#define S_IFBLK  0060000 /* Block device */
#define S_IFDIR  0040000 /* Directory */
#define S_IFCHR  0020000 /* Character device */
#define S_IFIFO  0010000 /* FIFO/Named pipe */
#define S_ISUID  0004000 /* Set user id upon execution */
#define S_ISGID  0002000 /* Set group id upon execution */
#define S_ISVTX  0001000 /* Sticky bit */

#define S_IRUSR 00400 /* User readable */
#define S_IWUSR 00200 /* User writable */
#define S_IXUSR 00100 /* User executable */

#define S_IRGRP 00040 /* Group readable */
#define S_IWGRP 00020 /* Group writable */
#define S_IXGRP 00010 /* Group executable */

#define S_IROTH 00004 /* World/Other readable */
#define S_IWOTH 00002 /* World/Other writable */
#define S_IXOTH 00001 /* World/Other executable */
```

4 Directories

From <linux/minix.h>

```
struct minix_dir_entry {
        unsigned short inode;
        char name[0];
};
```

Minix Directory Entry			
	00	01	02 – 1F
0x00	INODE NUMBER		char[30/14] FILENAME

The directory entry associates a filename with an inode. The first two bytes of a directory entry indicate the inode, the remainder is the filename. On the Minix FS, the root directory is inode 1 (`MINIX_ROOT_INO`). You can fit 32 directory entries on a 1k block which can exist anywhere in the data zone. Hard links are made when two entries point to the same node. Soft links are stored in the data zone, indexed by an inode entry. A hard links uses only a directory entry, whereas a soft link uses a directory entry, an inode, and at least one data zone.

5 Super Block

Again, from <linux/minix.h>:

```
struct minix_super_block {
        unsigned short s_ninodes;       /* Number of inodes */
        unsigned short s_nzones;        /* Number of data zones */
        unsigned short s_imap_blocks;   /* Space used by inode map (blocks) */
        unsigned short s_zmap_blocks;   /* Space used by zone map  (blocks) */
        unsigned short s_firstdatazone; /* First zone with ``file'' data */
        unsigned short s_log_zone_size; /* Size of a data zone =
                                               (1024 << s_log_zone_size) */
        unsigned long s_max_size;       /* Maximum file size (bytes) */
        unsigned short s_magic;         /* Minix 14/30 ID number */
        unsigned short s_state;         /* Mount state, was it
                                                       cleanly unmounted */
};
```

Minix Inode Entry																
	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0x00	No. inodes		No. zones		Imap blocks		ZMap blocks		First data zone		Log zone size		Max. file size			
0x10	Minix magic		Mount state													

The superblock for the minix fs is offset 1024 bytes from the start of the disk, this is to leave room for things like LILO and other boot code. The superblock basically details the size of the file system. It contains the number of inodes, numner of data zones, space used by the inode and zone map, the blocksize of the filesystem and a two character ID number indicating that it is indeed a Minix file system.