Minix2 File System (1A)

Young Won Lim
11/18/16

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice.

Based on

Based on Minix2

http://minix1.woodhull.com/current/2.0.4/

Basic Data Types

cache2.c
cache.c
device.c
filedes.c
inode.c
link.c
lock.c
main.c
misc.c
mount.c
open.c
path.c
pipe.c
protect.c
read.c
stadir.c
super.c
table.c
time.c
utility.c
write.c
buf.h
const.h
dev.h
file.h
fproc.h
fs.h
glo.h
inode.h
lock.h
param.h
proto.h
super.h
type.h

file.h – file pointer structure

```
/* This is the filp table.  It is an intermediary between file descriptors and
 * inodes.  A slot is free if filp_count == 0.
 */

EXTERN struct filp {
  mode_t           filp_mode;      /* RW bits, telling how file is opened */
  int              filp_flags;     /* flags from open and fcntl */
  int              filp_count;     /* how many file descriptors share this slot?*/
  struct inode *   filp_ino;       /* pointer to the inode */
  off_t            filp_pos;       /* file position */
} filp[NR_FILPS];

#define FILP_CLOSED     0     /* filp_mode: associated device closed */

#define NIL_FILP (struct filp *) 0      /* indicates absence of a filp slot */
```

inode.h – inode table (1)

Inode table.
This table holds inodes that are currently in use.

In some cases they have been opened by an open() or creat() system call,
in other cases the file system itself needs the inode for one reason or another,
such as to search a directory for a path name.

The first part of the struct holds fields that are present on the disk
the second part holds fields not present on the disk.

The disk inode part is also declared in "type.h"
as 'd1_inode' for V1 file systems
and 'd2_inode' for V2 file systems.

inode.h – inode table (2)

```
EXTERN struct inode {
  mode_t  i_mode;          /* file type, protection, etc. */
  nlink_t i_nlinks;        /* how many links to this file */
  uid_t   i_uid;           /* user id of the file's owner */
  gid_t   i_gid;           /* group number */
  off_t   i_size;          /* current file size in bytes */
  time_t  i_atime;         /* time of last access (V2 only) */
  time_t  i_mtime;         /* when was file data last changed */
  time_t  i_ctime;         /* when was inode itself changed (V2 only)*/
  zone_t  i_zone[V2_NR_TZONES];
                           /* zone numbers for direct, ind, and dbl ind */
```

inode.h – inode table (3)

```
EXTERN struct inode {
  …
  /* The following items are not present on the disk. */
  dev_t                 i_dev;      /* which device is the inode on */
  ino_t                 i_num;      /* inode number on its (minor) device */
  int                   i_count;    /* # times inode used; 0 means slot is free */
  int                   i_ndzones;  /* # direct zones (Vx_NR_DZONES) */
  int                   i_nindirs;  /* # indirect zones per indirect block */
  struct super_block *i_sp;         /* pointer to super block for inode's device */
  char                  i_dirt;     /* CLEAN or DIRTY */
  char                  i_pipe;     /* set to I_PIPE if pipe */
  char                  i_mount;    /* this bit is set if file mounted on */
  char                  i_seek;     /* set on LSEEK, cleared on READ/WRITE */
  char                  i_update;   /* the ATIME, CTIME, and MTIME bits here */
} inode[NR_INODES];
```

inode.h – inode table (4)

```
#define NIL_INODE       (struct inode *) 0  /* indicates absence of inode slot */

/* Field values.  Note that CLEAN and DIRTY are defined in "const.h" */
#define NO_PIPE         0     /* i_pipe is NO_PIPE if inode is not a pipe */
#define I_PIPE          1     /* i_pipe is I_PIPE if inode is a pipe */
#define NO_MOUNT        0     /* i_mount is NO_MOUNT if file not mounted on*/
#define I_MOUNT         1     /* i_mount is I_MOUNT if file mounted on */
#define NO_SEEK         0     /* i_seek = NO_SEEK if last op was not SEEK */
#define ISEEK           1     /* i_seek = ISEEK if last op was SEEK */
```

super.h – super block (1)

Super block table.
The root file system and every mounted file system has an entry here.
The entry holds information about the sizes of the bit maps and inodes.
The s_ninodes field gives the number of inodes available
for files and directories, including the root directory.
Inode 0 is on the disk, but not used.
Thus s_ninodes = 4 means that 5 bits will be used in the bit map,
bit 0, which is always 1 and not used,
and bits 1-4 for files and directories.

A super_block slot is free if s_dev == NO_DEV.

super.h – super block (2)

The disk layout is:

Item	# blocks
boot block	1
super block	1
inode map	s_imap_blocks
zone map	s_zmap_blocks
inodes	(s_ninodes + 'inodes per block' - 1)/'inodes per block'
unused	whatever is needed to fill out the current zone
data zones	(s_zones - s_firstdatazone) << s_log_zone_size

super.h – super block (3)

```
EXTERN struct super_block {
  ino_t         s_ninodes;        /* # usable inodes on the minor device */
  zone1_t       s_nzones;         /* total device size, including bit maps etc */
  short         s_imap_blocks;    /* # of blocks used by inode bit map */
  short         s_zmap_blocks;    /* # of blocks used by zone bit map */
  zone1_t       s_firstdatazone;  /* number of first data zone */
  short         s_log_zone_size; /* log2 of blocks/zone */
  off_t         s_max_size;       /* maximum file size on this device */
  short         s_magic;          /* magic number to recognize super-blocks */
  short         s_pad;            /* try to avoid compiler-dependent padding */
  zone_t        s_zones;          /* number of zones (replaces s_nzones in V2) */
 ....
} super_block[NR_SUPERS];

#define NIL_SUPER   (struct super_block *) 0
#define IMAP            0     /* operating on the inode bit map */
#define ZMAP            1     /* operating on the zone bit map */
```

super.h – super block (4)

```
EXTERN struct super_block {
  ...
  /* The following items are only used when the super_block is in memory. */
  struct inode *    s_isup;           /* inode for root dir of mounted file sys */
  struct inode *    s_imount;         /* inode mounted on */
  unsigned          s_inodes_per_block;  /* precalculated from magic number */
  dev_t             s_dev;            /* whose super block is this? */
  int               s_rd_only;        /* set to 1 iff file sys mounted read only */
  Int               s_native;         /* set to 1 iff not byte swapped file system */
  int               s_version;        /* file system version, 0 means bad magic */
  int               s_ndzones;        /* # direct zones in an inode */
  int               s_nindirs;        /* # indirect zones per indirect block */
  bit_t             s_isearch;        /* inodes below this bit number are in use */
  bit_t             s_zsearch;        /* all zones below this bit number are in use*/
} super_block[NR_SUPERS];
```

References

[1] http://minix1.woodhull.com/current/2.0.4/
[2]