SystemV Filesystem

It implements all of

- Xenix FS,
- SystemV/386 FS,
- Coherent FS.

To install:

- Answer the 'System V and Coherent filesystem support' question with 'y' when configuring the kernel.
- To mount a disk or a partition, use:

  ```
  mount [-r] -t sysv device mountpoint
  ```

  The file system type names:

  ```
  -t sysv
  -t xenix
  -t coherent
  ```

  may be used interchangeably, but the last two will eventually disappear.

Bugs in the present implementation:

- Coherent FS:
  - The "free list interleave" n:m is currently ignored.
  - Only file systems with no filesystem name and no pack name are recognized. (See Coherent "man mkfs" for a description of these features.)
- SystemV Release 2 FS:

  The superblock is only searched in the blocks 9, 15, 18, which corresponds to the beginning of track 1 on floppy disks. No support for this FS on hard disk yet.

These filesystems are rather similar. Here is a comparison with Minix FS:

- Linux fdisk reports on partitions
  - Minix FS 0x81 Linux/Minix
  - Xenix FS ??
  - SystemV FS ??
  - Coherent FS 0x08 AIX bootable
- Size of a block or zone (data allocation unit on disk)
  - Minix FS 1024
  - Xenix FS 1024 (also 512 ??)
  - SystemV FS 1024 (also 512 and 2048)
  - Coherent FS 512
- General layout: all have one boot block, one super block and separate areas for inodes and for directories/data. On SystemV Release 2 FS (e.g. Microport) the first track is reserved and all the block numbers (including the super block) are offset by one track.
- Byte ordering of "short" (16 bit entities) on disk:
  - Minix FS little endian 0 1
  - Xenix FS little endian 0 1
  - SystemV FS little endian 0 1
  - Coherent FS little endian 0 1

  Of course, this affects only the file system, not the data of files on it!
- Byte ordering of "long" (32 bit entities) on disk:
  - Minix FS little endian 0 1 2 3
  - Xenix FS little endian 0 1 2 3
  - SystemV FS little endian 0 1 2 3
  - Coherent FS PDP-11 2 3 0 1

  Of course, this affects only the file system, not the data of files on it!
- Inode on disk: "short", 0 means non-existent, the root dir ino is:

Minix FS	1
Xenix FS, SystemV FS, Coherent FS	2

- Maximum number of hard links to a file:

Minix FS	250
Xenix FS	??
SystemV FS	??
Coherent FS	>=10000

- Free inode management:
  - Minix FS
    a bitmap
  - Xenix FS, SystemV FS, Coherent FS
    There is a cache of a certain number of free inodes in the super-block. When it is exhausted, new free inodes are found using a linear search.
- Free block management:
  - Minix FS
    a bitmap
  - Xenix FS, SystemV FS, Coherent FS
    Free blocks are organized in a "free list". Maybe a misleading term, since it is not true that every free block contains a pointer to the next free block. Rather, the free blocks are organized in chunks of limited size, and every now and then a free block contains pointers to the free blocks pertaining to the next chunk; the first of these contains pointers and so on. The list terminates with a "block number" 0 on Xenix FS and SystemV FS, with a block zeroed out on Coherent FS.
- Super-block location:

Minix FS	block 1 = bytes 1024..2047
Xenix FS	block 1 = bytes 1024..2047
SystemV FS	bytes 512..1023
Coherent FS	block 1 = bytes 512..1023

- Super-block layout:
  - Minix FS:

```
unsigned short s_ninodes;
unsigned short s_nzones;
unsigned short s_imap_blocks;
unsigned short s_zmap_blocks;
unsigned short s_firstdatazone;
unsigned short s_log_zone_size;
unsigned long s_max_size;
unsigned short s_magic;
```

  - Xenix FS, SystemV FS, Coherent FS:

```
unsigned short s_firstdatazone;
unsigned long  s_nzones;
unsigned short s_fzone_count;
unsigned long  s_fzones[NICFREE];
unsigned short s_finode_count;
unsigned short s_finodes[NICINOD];
char           s_flock;
char           s_ilock;
char           s_modified;
char           s_rdonly;
unsigned long  s_time;
short          s_dinfo[4]; -- SystemV FS only
unsigned long  s_free_zones;
unsigned short s_free_inodes;
short          s_dinfo[4]; -- Xenix FS only
unsigned short s_interleave_m,s_interleave_n; -- Coherent FS only
char           s_fname[6];
char           s_fpack[6];
```

    then they differ considerably:

    Xenix FS:

```
char           s_clean;
char           s_fill[371];
long           s_magic;
long           s_type;
```

    SystemV FS:

```
        long           s_fill[12 or 14];
        long           s_state;
        long           s_magic;
        long           s_type;
```

Coherent FS:

```
        unsigned long  s_unique;
```

Note that Coherent FS has no magic.

- Inode layout:
  - Minix FS:

    ```
    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[7+1+1];
    ```

  - Xenix FS, SystemV FS, Coherent FS:

    ```
    unsigned short i_mode;
    unsigned short i_nlink;
    unsigned short i_uid;
    unsigned short i_gid;
    unsigned long  i_size;
    unsigned char  i_zone[3*(10+1+1+1)];
    unsigned long  i_atime;
    unsigned long  i_mtime;
    unsigned long  i_ctime;
    ```

- Regular file data blocks are organized as
  - Minix FS:
    - 7 direct blocks
    - 1 indirect block (pointers to blocks)
    - 1 double-indirect block (pointer to pointers to blocks)
  - Xenix FS, SystemV FS, Coherent FS:
    - 10 direct blocks
    - 1 indirect block (pointers to blocks)
    - 1 double-indirect block (pointer to pointers to blocks)
    - 1 triple-indirect block (pointer to pointers to pointers to blocks)

Minix FS	32	32
Xenix FS	64	16
SystemV FS	64	16
Coherent FS	64	8

- Directory entry on disk
  - Minix FS:

    ```
    unsigned short inode;
    char name[14/30];
    ```

  - Xenix FS, SystemV FS, Coherent FS:

    ```
    unsigned short inode;
    char name[14];
    ```

Minix FS	16/32	64/32
Xenix FS	16	64
SystemV FS	16	64
Coherent FS	16	32

- How to implement symbolic links such that the host fsck doesn't scream:
  - Minix FS normal
  - Xenix FS kludge: as regular files with chmod 1000
  - SystemV FS ??
  - Coherent FS kludge: as regular files with chmod 1000

Notation: We often speak of a "block" but mean a zone (the allocation unit) and not the disk driver's notion of "block".