NAME

fstab - static information about the file systems

SYNOPSIS

#include <fstab.h>

DESCRIPTION

The file **fstab** contains descriptive information about the various file systems. **fstab** is only read by programs, and not written; it is the duty of the system administrator to properly create and maintain this file. Each file system is described on a separate line; fields on each line are separated by tabs or spaces. The order of records in **fstab** is important because fsck(8), mount(8), and umount(8) sequentially iterate through **fstab** doing their thing.

The first field, (*fs_spec*), describes the special device or remote file system to be mounted. The contents are decoded by the strunvis(3) function. This allows using spaces or tabs in the device name which would be interpreted as field separators otherwise.

The second field, (*fs_file*), describes the mount point for the file system. For swap partitions, this field should be specified as "none". The contents are decoded by the strunvis(3) function, as above.

The third field, (*fs_vfstype*), describes the type of the file system. The system can support various file system types. Only the root, /usr, and /tmp file systems need be statically compiled into the kernel; everything else will be automatically loaded at mount time. (Exception: the FFS cannot currently be demand-loaded.) Some people still prefer to statically compile other file systems as well.

The fourth field, (*fs_mntops*), describes the mount options associated with the file system. It is formatted as a comma separated list of options. It contains at least the type of mount (see *fs_type* below) plus any additional options appropriate to the file system type. See the options flag (**-o**) in the mount(8) page and the file system specific page, such as mount_nfs(8), for additional options that may be specified. All options that can be given to the file system specific mount commands can be used in **fstab** as well. They just need to be formatted a bit differently. The arguments of the **-o** option can be used without the preceding **-o** flag. Other options need both the file system specific flag and its argument, separated by an equal sign. For example, mounting an msdosfs(5) filesystem, the options

```
-o sync -o noatime -m 644 -M 755 -u foo -g bar
```

should be written as

```
sync,noatime,-m=644,-M=755,-u=foo,-g=bar
```

in the option field of **fstab**.

If the options "userquota" and/or "groupquota" are specified, the file system is automatically processed by the quotacheck(8) command, and user and/or group disk quotas are enabled with quotaon(8). By default, file system quotas are maintained in files named *quota.user* and *quota.group* which are located at the root of the associated file system. These defaults may be overridden by putting an equal sign and an alternative absolute pathname following the quota option. Thus, if the user quota file for */tmp* is stored in */var/quotas/tmp.user*, this location can be specified as:

userquota=/var/quotas/tmp.user

If the option "failok" is specified, the system will ignore any error which happens during the mount of that filesystem, which would otherwise cause the system to drop into single user mode. This option is implemented by the mount(8) command and will not be passed to the kernel.

If the option "noauto" is specified, the file system will not be automatically mounted at system startup. Note that, for network file systems of third party types (i.e., types supported by additional software not included in the base system) to be automatically mounted at system startup, the *extra_netfs_types* rc.conf(5) variable must be used to extend the rc(8) startup script's list of network file system types.

If the option "late" is specified, the file system will be automatically mounted at a stage of system startup after remote mount points are mounted. For more detail about this option, see the mount(8) manual page.

If the option "update" is specified, it indicates that the status of an already mounted file system should be changed accordingly. This allows, for example, file systems mounted read-only to be upgraded read-write and vice-versa. By default, an entry corresponding to a file systems that is already mounted is going to be skipped over when processing **fstab**, unless it's a root file system, in which case logic similar to "update" is applied automatically.

The "update" option is typically used in conjunction with two **fstab** files. The first **fstab** file is used to set up the initial set of file systems. The second **fstab** file is then run to update the initial set of file systems and to add additional file systems.

The type of the mount is extracted from the *fs_mntops* field and stored separately in the *fs_type* field (it is not deleted from the *fs_mntops* field). If *fs_type* is "rw" or "ro" then the file system whose name is given in the *fs_file* field is normally mounted read-write or read-only on the specified special file.

If *fs_type* is "sw" then the special file is made available as a piece of swap space by the swapon(8) command at the end of the system reboot procedure. For swap devices, the keyword "trimonce" triggers

the delivery of a BIO_DELETE command to the device. This command marks the device's blocks as unused, except those that might store a disk label. This marking can erase a crash dump. To delay **swapon** for a device until after **savecore** has copied the crash dump to another location, use the "late" option. For vnode-backed swap spaces, "file" is supported in the *fs_mntops* field. When *fs_spec* is an md(4) device file ("md" or "md[0-9]*") and "file" is specified in *fs_mntopts*, an md(4) device is created with the specified file used as backing store, and then the new device is used as swap space. Swap entries on *.eli* devices will cause automatic creation of encrypted devices. The "ealgo", "aalgo", "keylen", "notrim", and "sectorsize" options may be passed to control those geli(8) parameters. The fields other than *fs_spec* and *fs_type* are unused. If *fs_type* is specified as "xx" the entry is ignored. This is useful to show disk partitions which are currently unused.

The fifth field, (*fs_freq*), is used for these file systems by the dump(8) command to determine which file systems need to be dumped. If the fifth field is not present, a value of zero is returned and **dump** will assume that the file system does not need to be dumped. If the fifth field is greater than 0, then it specifies the number of days between dumps for this file system.

The sixth field, (*fs_passno*), is used by the fsck(8) and quotacheck(8) programs to determine the order in which file system and quota checks are done at reboot time. The *fs_passno* field can be any value between 0 and 'INT_MAX-1'.

The root file system should be specified with a *fs_passno* of 1, and other file systems should have a *fs_passno* of 2 or greater. A file system with a *fs_passno* value of 1 is always checked sequentially and be completed before another file system is processed, and it will be processed before all file systems with a larger *fs_passno*.

For any given value of *fs_passno*, file systems within a drive will be checked sequentially, but file systems on different drives will be checked at the same time to utilize parallelism available in the hardware. Once all file system checks are complete for the current *fs_passno*, the same process will start over for the next *fs_passno*.

If the sixth field is not present or is zero, a value of zero is returned and fsck(8) and quotacheck(8) will assume that the file system does not need to be checked.

The *fs_passno* field can be used to implement finer control when the system utilities may determine that the file system resides on a different physical device, when it actually does not, as with a ccd(4) device. All file systems with a lower *fs_passno* value will be completed before starting on file systems with a higher *fs_passno* value. E.g. all file systems with a *fs_passno* of 2 will be completed before any file systems with a *fs_passno* of 3 or greater are started. Gaps are allowed between the different *fs_passno* values. E.g. file systems listed in */etc/fstab* may have *fs_passno* values such as 0, 1, 2, 15, 100, 200, 300, and may appear in any order within */etc/fstab*.

```
#define    FSTAB_RW        "rw"      /* read/write device */
#define    FSTAB_RQ        "rq"      /* read/write with quotas */
#define    FSTAB_RO        "ro"      /* read-only device */
#define    FSTAB_SW        "sw"      /* swap device */
#define    FSTAB_XX        "xx"      /* ignore totally */

struct fstab {
        char    *fs_spec; /* block special device name */
        char    *fs_file;   /* file system path prefix */
        char    *fs_vfstype;        /* File system type, ufs, nfs */
        char    *fs_mntops;         /* Mount options ala -o */
        char    *fs_type; /* FSTAB_* from fs_mntops */
        int     fs_freq;    /* dump frequency, in days */
        int     fs_passno;          /* pass number on parallel fsck */
};
```

The proper way to read records from *fstab* is to use the routines getfsent(3), getfsspec(3), getfstype(3), and getfsfile(3).

FILES

/etc/fstab The file **fstab** resides in */etc*.

EXAMPLES

```
# Device  Mountpoint      FStype    Options             Dump    Pass#
#
# UFS file system.
/dev/da0p2          /               ufs       rw                  1       1
#
# Swap space on a block device.
/dev/da0p1          none            swap      sw                  0       0
#
# Swap space using a block device with GBDE/GELI encryption.
# aalgo, ealgo, keylen, sectorsize options are available
# for .eli devices.
/dev/da1p1.bde      none            swap      sw                  0       0
/dev/da1p2.eli      none            swap      sw                  0       0
#
# tmpfs.
tmpfs               /tmp            tmpfs     rw,size=1g,mode=1777      0 0
#
```

```
# UFS file system on a swap-backed md(4).  /dev/md10 is
# automatically created.  If it is "md", a unit number
# will be automatically selected.
md10                    /scratch    mfs         rw,-s1g                 0           0
#
# Swap space on a vnode-backed md(4).
md11                    none                    swap        sw,file=/swapfile   0 0
#
# CDROM.  "noauto" option is typically used because the
# media is removable.
/dev/cd0  /cdrom                cd9660    ro,noauto 0           0
#
# NFS-exported file system.  "serv" is an NFS server name
# or IP address.
serv:/export            /nfs                    nfs         rw,noinet6              0           0
```

SEE ALSO

getfsent(3), getvfsbyname(3), strunvis(3), ccd(4), dump(8), fsck(8), geli(8), mount(8), quotacheck(8), quotaon(8), swapon(8), umount(8)

HISTORY

The **fstab** file format appeared in 4.0BSD.