Df(1)

**名称：** df-报告文件系统磁盘空间利用情况

**语法：** df [选项参数] [文件]

**描述：**

这份帮助手册记录GNU版本的df。

Df 以文件名为参数输出磁盘的可用量。如果没有指定的文件名，那将输出所有已经挂载的文件系统中可用的空间。

文件系统的block默认为1K，可以通过设置环境变量POSIXLY\_CORRECT将block大小改为512b。

如果参数一个包含有已挂载文件系统的磁盘设备节点（设备节点参见http://blog.csdn.net/bestrem\_9/article/details/4004335）的绝对文件名，df会输出文件系统的可用空间，而不是输出包含有设备节点的文件系统。（总是根文件系统）

这个版本的df不能显示出未挂载文件系统的可用空间。因为在大多数的系统中要做到输出未挂载文件系统的信息都要求要有非常复杂的关于文件系统结构的内部知识。

**选项：**

显示存有文件的文件系统的信息，或者默认显示所有文件系统的信息。

对于长选项的强制参数对于短选项也是强制的。

-a --all:包括虚拟文件系统

-B --block-size=SIZE ： 用SIZR大小的blocks

--direct: : 最后一列显示文件的路径而不是挂载点（感觉好废）

--total: 生成一个最终的总值

-h , --human-readable: 以人类容易阅读的格式和大小输出

-H，--si: 使用1000进制而不是1024

-i, --inode: 列出inode的使用信息而不是block的使用信息

-k:同—block-size=1K

-l, --local: 只列出本地文件系统

--no-sync: 在得到使用情况信息之前调用sync

-t, --type=TYPE: 列出指定TYPE类型的文件系统

-T，--print-type: 输出文件系统的类型

-x. –exclude-type: 和-T相反

-v : （忽略）

--help: 显示帮助并且退出

-- version: 输出版本信息并且退出

值的显示是以--block-size中设置的SIZE为单位的。或者来自环境变量： DF\_BLOCK\_SIZE，BLOCK\_SIZE 和 BLOCKSIZE。如何没有对以上进行设置，那么默认的单位为1024字节（即1K，如果设置了环境变量POSIXLY\_CORRECT， 则为512字节）

SIZE可以是整型数后面跟随一下任意的单位： KB, K, MB, M, G, T,

P, E, Z, Y

注意： 这里KB，MB是1000进制，而K，M是1024进制

Du(1)

**名字：**Du-评估文件使用空间

**语法：** du [选项] [文件]

du [ 选项]… --files0-from=F

**描述：**

总结每个文件的磁盘使用状况，对目录而言是递归执行的

对长选项的强制参数对短选项也强制

-a, all: 输出对所有文件的计算（注意： 不加选项-a 的时候，系统只是不把普通文件的大小单独列出来，但是仍然把普通文件的大小加入文件大小中）

-apparent-size: 表面用量大小而非磁盘用量；虽然形式大小一般更小，但也有可能因为稀疏文件之间的孔，内部碎片，没有指向的block之类的东西而变得更大。

问： 磁盘用量和表面大小分别是什么？

-B，--block-size=SIZE: 使用SIZE字节的block

-b，bytes: 等同于“-apparent-size --block-size=1”

-c, --total: 产生一个总值

-D, --dereference-args: 显示实际文件的大小而不是显示软连接（符号链接）的大小（找到真正的目的地）（du -D 软连接名）

--file0-from=F: 输出文件F中文件名为无效字符的文件的磁盘用量；如果F是-，那么从标准输入来输入文件名

-H： 同—dereference-args(-D)

-h, --human-readable: 按人类习惯的单位格式输出

--si：同-h，但是是1000进制而不是1024进制

-k: 同 –block-size=1K

-l，--count-links : 如果是硬链接，就多次计算他的大小

-m : 同—block-size=1M

-L, --dereference: 找出任何符号链接指示的真正目的地

-P, --no-dereference : 不跟随任何符号链接（默认）

-O, --null: 将每个换行视作0字节而非换行符

-S，--separate-dirs: 目录大小不包含子目录的大小

-s, --summarize: 只列出总的目录大小，但大小包含了目录内所有文件/子目录的大小

-x, --one-file-system: 跳过属于不同文件系统之上的目录

-X, --exclude-from=FILE: 排除与指定文件中模式描述相符合的文件

--exclude=PATTERN: 排除符合模式的文件

--max-depth=N: 指定统计N层及N层一下目录的大小。但N=0时，命令相当于summarize

--time: 输出目录中文件/子目录上次的修改时间（可以配合-a一起用）

--time=WORD: 输出指定的时间而非mtime，WORD可以 ： atime, access, use, ctime or status

--time-style=STYLE:以指定的STYLE风格输出时间，风格可以是：full-iso, long-iso, iso, + FORMAT 。 样式解释规则同”date”

值的显示是以--block-size中设置的SIZE为单位的。或者来自环境变量： DF\_BLOCK\_SIZE，BLOCK\_SIZE 和 BLOCKSIZE。如何没有对以上进行设置，那么默认的单位为1024字节（即1K，如果设置了环境变量POSIXLY\_CORRECT， 则为512字节）

SIZE可以是整型数后面跟随一下任意的单位： KB, K, MB, M, G, T,

P, E, Z, Y

**模式：**

模式是shell模式（而不是常规的表达式）。 模式？匹配任一单一字符，而\*匹配任意字符串（包含0个，1个或多个字符）

例如：\*.o 会匹配以.o结尾的文件。

命令du --exclude=’\*.o’ 会忽略所有以.o结尾的文件/目录（包括文件.o本身）

Fdisk(8)

**名字：** fdisk——linux分区表操作

**语法：**

Fdisk[-uc] [-b sectorsize] [-C suls] [-H heads] [-S sects] 设备名

Fdisk -l [-u] [设备]

Fdisk -s 分区

Fdisk -v

Fdisk -h

**描述：**

硬盘可以被封为一个或多个的逻辑磁盘，称作分区。 这些分区可以在分区表中找到，而分区表储存在磁盘的0号扇区中（因此第一个磁盘非常重要，也因为它只有512btyes的容量，因此限制了只能有4个主分区）

在BSD世界中，会讨论磁盘片和磁盘标签

Linux需要至少一个分区，作为根文件系统。 它可以使用swap 文件或/和swap 分区，而这之中swap分区更为有效。 所以一般来说我们会划分一个分区作为swap分区。在英特尔的兼容硬件中，用于启动系统的BIOS一般只能进入前1024个柱面。 因此，一般会划分一个只有几M的分区作为boot分区，用于储存一些内核图片和辅助性文件，来确保BIOS在启动系统的时候能够获取这些东西（所以boot分区应该是用于确保系统的正常启动的） 为了保障系统安全，减轻管理，备份信息或者测试等原因，会划分出一些其它分区。

Fdisk（在第一种形式的调用中） 是一种用于建立和操作分区表的菜单式程序。 它可以识别DOS类型分区表和BSD或SUN 类型分区标签。

Fdisk 不能识别GUID 分区表 ，也不能用于处理比较大的分区。在很多特殊情况下，我们会使用GUN的更加先进的**parted(8)**

设备一般是/dev/sda , /dev/sdb 之类的。 一个设备名指代的是整个磁盘（注意不是某一个分区，所以设备名后面是不加数字的）

较旧的系统没有libata（linux内核的一个库，支持AT A host contrillers和设备）,在IDE 和SCSI两种磁盘上有所区分。 这种情况下设备名称会是 /dev/hd\* (IDE) 或 /dev/sd\* (SCSI).

分区是一种有分区号命名的设备。 例如，/dev/sda1是系统中第一块磁盘的第一个分区。 （参阅 linux内核资料 ( Documentation/devices.txt 文件)）

BSD/SUN 类型磁盘标签可以被描述为8个分区，其中第三个分区应该是“全磁盘”分区。 不要从柱面0的第一个扇区开始划分磁盘（九号线swap 分区一样），因为这样会毁掉磁盘标签

Mkfs(8)

**名字：**mkfs——建立linux文件系统

**语法：** mkfs [-V ] [-t 文件系统类型] [fs选项] filesys [blocks]

**描述：**

Mkfs 是用于在设备上创建文件系统，常用于文件分区。 Filesys 可以是设备名称（如 /dev/hda1, /dev/sda2），或者是包含文件系统的常规文件。 Blocks指的是用于文件系统的blocks的数量

Mkfs操作成功返回0，失败返回1

事实上,mkfs只是linux中一个很多文件系统创建工具的前端工具（mkfs.fstype）。具体的创建工具可能可以在一些目录下找到（/sbin, .sbin/fs, /sbin/fs.d, /etc/fs, /etc等，具体的列表是在编译的时候定义的，但至少会存在于/sbin 和 /sbin/fs中），并且最终会在PATH环境变量列出的目录中。 详情常见file syste,-specific builder manual pages

**选项：**

-V : 产生冗长的输出，包括执行的所有文件系统特定的命令（前面说过mkfs只是这些命令的前端工具）Specifying this option more than once inhibits execution of any file system-specific commands

-t 文件系统类型： 指定要建立的文件系统的类型 如果没有制定，那么使用默认文件系统（目前是ext2）

**Fs选项**： 文件系统特定的选项会被传输到背后真正的文件系统建立工具。虽然无法保证，但是大多数的文件系统创建工具都支持下面的选项：

-c ：在创建文件系统之前检查设备的bad blocks

-l 文件名： 通过文件名读取bad blocks（即是检查该文件的bad blocks）

-v : 生成冗长的输出

**相关命令：**

Fs(5), badblocks(8), ~~~~~~

Fsck(8)

名字： fsck——检查和修复linux文件系统。

语法： fsck [-sAVRTMNP] [-C [fd] ] [-t 文件系统类型] [filesys…] [--] [fs-specific-options]

描述：

Fsck用于检查并且有选择地修复一个或者多个linux文件系统。 Filesys可以使设备名（比如/dev/hdc1, /dev/sdb2） 或者挂载点（如/,/usr/home），或者ext2标签，或者UUIDspecifier~~~~正常情况下，fsck程序会平行地处理所有的物理磁盘来节省检测的时间。

如果命令行上没有指定的文件系统，也没有制定-A选项，那么fsck会默认串行地检测/etc/fstab下的文件系统。这就相当于是选项-As。

Fsck执行退出后的返回值是以下情况的总和：

0 -没有错误

1 -文件系统的错误被纠正

2 - 系统需要重启

4 - 文件系统错误未解决

8 - 操作性错误

16 - 使用或者是语法错误

32 - fsck 被用户请求取消

128 – 分享库错误

当多个被检测的文件系统的返回值是每个已经被检测的文件系统的返回值的按位或时，返回返回值。

事实上fsck和mkfs类似，只是多个文件系统检测器的前端工具(fsck.fstype)。 特定的文件系统检测工具可以在/sbin, /etc/fs 和/etc以及PATH环境变量列出的目录中找到。

选项：

-s : 串行化fsck操作。 在检测多个文件系统并且检测工具在互动模式的时候这个参数会非常有用。（注意： e2fsck(8)默认执行互动模式。 如果要使e2fsck（8）进入非互动模式，就必须要制定-p或者-a选项（如果希望错误自动修正），或者-n选项(如果不希望错误自动修正)）

-t fslist: 制定要检测的文件系统类型。 当指定-A 标记，只有符合fslist的系统文件才会被检测。 Fslist的参数是由文件系统名和选项说明符组成的逗号分隔的列表。 可以通过在文件系统前面加取反字符“no” 或者’!’， 来告诉系统只检测这些文件系统之外的文件系统。

选项说明符要被包含在都好风格的fslist中。 格式必须是opts=fs选项。 如果

~~~~~~~~~~~~~~~~~~~~~~~~~·~~~