

# **Paragon NTFS and HFS+ for Linux 8.5**

## **User Guide**

© 2011 Paragon Software Group

Generated 5/30/2011

# Paragon NTFS and HFS+ for Linux 8.5

---

## User Guide

### Abstract

*This document covers implementation of NTFS and HFS+ file systems support in Linux operating systems using Paragon NTFS and HFS+ file system drivers. Basic installation procedures are described. Detailed mount options description is given. File system creation (formatting) and checking utilities are described. Short list of supported NTFS/HFS+ features is given with limitations imposed by Linux. Advanced troubleshooting section is also included.*

# Paragon NTFS and HFS+ for Linux 8.5

---

## User Guide

© 2011 Paragon Software Group

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Printed: May 2011 in Freiburg, Germany.

### We welcome your feedback

*Please send your feedback to  
sales@paragon-software.com  
or use your User account with  
Paragon Software.*

### Special thanks to:

*All the people who contributed to this document,  
either by writing text, developing solutions to  
various issues, performing tests, collecting  
information or by requested support from our  
team. To our customers who continue to support us  
and help us to improve the product by constantly  
demanding more.*

# Table of Contents

<b>Part I Introduction</b>	<b>2</b>
1 Historical review.....	2
2 Paragon UFSD technology.....	2
3 How UFSD works on Linux.....	3
4 Key Features.....	3
<b>Part II System requirements</b>	<b>6</b>
1 Hardware requirements.....	6
2 Software requirements.....	6
<b>Part III Installation</b>	<b>9</b>
1 Shipment.....	9
2 Components.....	9
3 Installing the Drivers.....	9
4 Uninstalling the Drivers.....	11
<b>Part IV Using The Driver</b>	<b>14</b>
1 Getting started.....	14
2 Mounting NTFS/HFS+ Partitions.....	14
3 Dirty flag issues.....	15
4 GPT issues.....	15
5 Lazy open.....	16
6 Issues with large HDDs.....	17
7 Unmounting NTFS/HFS+ Partitions.....	17
8 Choosing the codepage/charset for NTFS/HFS+ Partitions.....	17
<b>Part V Mount options</b>	<b>20</b>
1 Mount options.....	20
<b>Part VI Additional Utilities</b>	<b>25</b>
1 NTFS utilities.....	25
infntfs.....	25
chkntfs.....	27
mkntfs.....	28
dfntfs .....	30
wipe .....	33
mftpack.....	34
hdlnk .....	36

fsutil .....	38
junction.....	53
2 HFS+ utilities.....	56
mkhfts.....	56
chkhfs.....	57
<b>Part VII Troubleshooting</b>	<b>60</b>
1 Troubleshooting processes.....	60
2 Mount troubleshooting.....	61
3 The install.sh script can't find kernel sources.....	62
4 Can't compile the NTFS/HFS+ for Linux driver.....	62
5 "Can't load module" message at the end of installation.....	62
6 ufsd Module: kernel-module version mismatch.....	63
7 ufsd Module: create_module: operation is not permitted.....	63
8 insmod: a module named as ufsd already exists.....	63
9 When I run the "insmod ufsd.o" command, there are some error messages.....	63
10 I can't mount NTFS/HFS+ volume.....	63
<b>Part VIII NTFS &amp; HFS Compatibility</b>	<b>65</b>
1 NTFS features.....	65
2 HFS features.....	65
<b>Part IX Frequently Asked Questions</b>	<b>67</b>
1 What are 'minor errors' reported by chkntfs utility?.....	67
2 Warnings on Windows7/Vista when NTFS HDD is reconnected from Linux.....	68
3 Recently changed file has its modification time a few hours ahead of or behind the current system time. Why?.....	72
<b>Part X Legal questions</b>	<b>74</b>
1 NTFS legal questions.....	74
2 HFS legal questions.....	74

**Part**

---



**I**

**Introduction**

## 1.1 Historical review

Historically, different operating systems supported different file systems. Sharing files among different platforms was not an easy task. For instance, documents that were created in Windows and are stored on NTFS partitions may be inaccessible under Linux, because Linux does not include full support for NTFS. For example, open-source NTFS-3G NTFS driver does not support random write access to compressed files.

Paragon NTFS&HFS driver for Linux solves these problems — now everyone can access NTFS and HFS+ partitions from Linux in a usual manner with maximum performance and reliability. The driver allows mounting NTFS and HFS+ partitions, so that programs may work transparently with these mounted partitions — browse contents, open documents, run applications, work with existing files (delete/copy/modify) and create new ones.

Paragon combined NTFS&HFS driver for Linux is commercial Linux driver for local access to NTFS and HFS+ volumes. It supports full read/write access. The driver is a Kernel module, which guarantees rapid and transparent access to supported file systems. Mount volumes manually or insert into `fstab`, and NTFS & HFS+ partitions will be available like any other directory tree.

Paragon NTFS&HFS Professional also includes useful additional utilities that provide the ability to check integrity, create/wipe/defrag NTFS volumes, perform many NTFS file system related tasks and copy (backup) files, saving all their attributes, between NTFS and native Linux file systems.

## 1.2 Paragon UFSD technology

UFSD (Universal File System Driver) is an unique technology developed by Paragon Software to provide full access (read/write, format, etc.) to volumes of the popular file systems: NTFS, FAT, Ext2Fs, Ext3Fs, HFS, HFS+ etc. under various platforms, including Windows, Linux, Mac OS X, etc. in case these file systems are not otherwise supported.

UFSD technology provides access directly to the physical devices that is why it can process partitions regardless of their support by the current Operating System (OS). With UFSD it is possible to mount NTFS and HFS+ partitions under Linux, thus getting access to its contents, just the way it is implemented in the NTFS&HFS for Linux driver, and the technology also allows direct access via physical device addressing, the way it is implemented in the driver too.

Paragon UFSDs are designed to be readily integrated into any solution using our UFSD Software Development Kit (UFSD SDK), which includes all of the necessary tools to develop applications with the following main features:

- Access to un-mounted partitions (i.e. drive letter not assigned);
- Access to other file systems that normally would not be supported by the operating system;
- Platform-independent UFSD API.

Note: NTFS and HFS+ drivers for Linux as well as utilities were written using UFSD SDK.

## 1.3 How UFSD works on Linux

Modern operating systems are based on the concept of Installable File System drivers (IFS). User simply needs to provide an operating system with the proper file system driver to work with the file system in usual manner. Paragon NTFS&HFS for Linux includes NTFS and HFS+ drivers for Linux environment. Once appropriate components of Paragon NTFS&HFS for Linux are installed, the operating system can mount these file systems and work with directories/files stored on the file systems.

## 1.4 Key Features

Paragon NTFS&HFS for Linux Combo 8.5 is released in the Express and Professional Editions. All of the products share the following features:

- Transparent read-write access to NTFS and HFS volumes — single Kernel module provides support both NTFS and HFS+ file systems
- High performance (in some cases even better than Ext3 FS);
- Easy installation and uninstallation (assistant scripts);
- Support for the latest Linux Kernels and distributions;
- Support for SMP kernels;
- File sharing over network via SAMBA;
- No system degradation during data transfers;
- Unlimited file and volume size (within NTFS/HFS+ and Kernel limitations).

What's new in Paragon NTFS&HFS for Linux 8.5:

- Support for modern Linux Kernels (up to 2.6.38);
- Improved read/write performance;
- Optimized disk space allocation to reduce fragmentation of files;
- Improved performance of HFS+ journal;
- All known bugs are fixed.

NTFS-specific features:

- Full support for compressed files (random access for reading and writing with no limitations);
- Sparse files;
- Alternate data streams;

NTFS compatibility information:

File system version	Comments
---------------------	----------



NTFS version 1.2	Originates from Microsoft Windows NT 4.0
NTFS version 3.0	Originates from Microsoft Windows 2000
NTFS version 3.1	Originates from Microsoft Windows XP/2003/Vista and 7

Additional features of the Professional edition:

- Support for encrypted files copying (**cpntfs** utility);

Additional features of the Professional Edition:

- Additional NTFS utilities:
  - [mkntfs](#)<sup>[28]</sup> utility - format any partition as NTFS under Linux;
  - [chkntfs](#)<sup>[27]</sup> utility - check NTFS partition integrity and fix errors;
  - [infntfs](#)<sup>[25]</sup> utility - show detailed information about NTFS partitions;
  - [dfntfs](#)<sup>[30]</sup> utility - defragment a NTFS volume;
  - [wipe](#)<sup>[33]</sup> utility - fill with zeros free space on a NTFS/FAT volume;
  - [mftpack](#)<sup>[34]</sup> utility - pack/truncate MFT (Master File Table) on a NTFS volume;
  - [hdlnk](#)<sup>[36]</sup> utility - enumerate all hard links on a NTFS volume;
  - [junction](#)<sup>[53]</sup> utility - show reparse points on a NTFS volume;
  - [fsutil](#)<sup>[38]</sup> utility - perform many NTFS file system related tasks. Powerful utility;
  - **cpntfs** utility - create an archive of the NTFS volume or separate files/directories including all streams and attributes.
- Additional HFS+ utilities:
  - [mkhfs](#)<sup>[56]</sup> utility - format any partition as HFS+ under Linux;
  - [chkhfs](#)<sup>[57]</sup> utility - check HFS+ partition for integrity and fix errors;

**Part**

---



**II**

**System  
requirements**

This topic highlights requirements to hardware and software that may be used to run Paragon NTFS&HFS for Linux driver.

## 2.1 Hardware requirements

### Minimum hardware requirements:

- Processor: Intel Pentium 300 MHz and higher, or compatible;
- both 32- and 64-bit CPUs are supported.
- 16MB of RAM.

Due to unique technology our NTFS&HFS for Linux drivers have low system requirements. For example, it is enough for our driver to have 500KB of free RAM to work with NTFS partitions larger than 250 GB. Combined NTFS&HFS Kernel module itself occupies around 430 Kb of RAM.

### Real-life values

200 Kb maximum while executing 5 commands like `dd if=/dev/zero of=/mnt/sda1/test bs=1M count=1000&` in background.

516 Kb maximum while executing `rsync -r /home /mnt/sda1` command.

17 Mb maximum while compiling bench test on Desktop Linux system in virtual environment using NTFS file system: a file-tree with a size about 220 Mb was created and patched, simulating Linux Kernel installation process.

RAM consumption depends first of all on whole amount of memory available in the system. If it is low then the driver wouldn't keep a lot of descriptors opened to keep the memory usage at minimum.

## 2.2 Software requirements

### Supported Linux kernels:

- Linux with kernel versions 2.4.x;
- Linux with kernel versions 2.6.x (NTFS&HFS driver was tested with Kernels up to 2.6.38.4).

### Linux distributions the products were tested with:

- Ubuntu 10.04, 11.04
- OpenSUSE 11.4
- Manrdiva Free 2011
- Debian 6.01

- LinuxMint 10
- Slackware 13.1
- Fedora 15
- CentOS 5.5

## Development Environment

A development environment is required to compile Linux drivers and utilities. Please verify that these tools are all functional. The easiest way is to choose the developer toolkit when installing Linux.

### What must be installed:

- Kernel source code (recommended) or Kernel header files (doesn't always work);

```
#rpm -qa | grep kernel-source (for RPM based kernel-sources)
```

- GNU C compiler (GCC);

```
#gcc --version
```

- GNU C++ compiler (g++) — for Professional version only;

```
#g++ --version
```

- GNU Make;

```
#make --version
```

- GNU ld (binutils);

```
#ld --version
```

- Modutils (module-init tools);

```
#insmod -V
```

## Limitations

- GNU C compiler (gcc) version 2.95 or higher is required.
- The user should login as root to install the drivers and utilities.
- Correct operation is not guaranteed when using Linux with kernel versions 2.3.x and 2.5.x (which are known for their instability).
- Correct operation is not guaranteed for customized Linux kernels. Commercial porting service to customized Linux kernels is available from Paragon Software Group — for more information send e-mail to [sales@paragon-software.com](mailto:sales@paragon-software.com)).

**Part**

---

**III**

**Installation**

USER MANUAL

This section describes workflows related to installing and using Paragon NTFS&HFS for Linux driver.

## 3.1 Shipment

The setup files for each product of the family are provided as the downloadable TGZ archives, which can be downloaded from the company site.

## 3.2 Components

The package includes the following components:

- The source files for the NTFS&HFS for Linux driver;
- The source files for additional utilities (for Professional edition only);
- Assistant script files, which are purposed to simplify the installation and uninstallation routines.

Paragon NTFS&HFS Linux driver and utilities must be compiled on the end user's system for correct configuration. These modules are the open source code with libraries. Before installing the modules, one must build driver and utilities by using the GNU development tools listed above.

## 3.3 Installing the Drivers

First, NTFS&HFS Combo driver must be built and installed.

### Steps to install the NTFS & HFS for Linux driver are as follows:

1. Log in as root. This step is obligatory;
2. Build and install the NTFS & HFS Combo driver using `install.sh` script. Alternatively, driver binary module may be built manually using `'make'` command.
3. Install the NTFS & HFS Combo driver (this step will make the modules available for use);
4. Activating (loading) the driver. After building and installing, the NTFS & HFS Combo driver can be referenced as "used file system driver" when mounting NTFS and HFS partitions.

The steps 1-3 should be made only once while the step 4 is the standard way of using file system drivers in Linux environment.

NTFS & HFS for Linux include a set of assistant script files for the simplification of building, installing and uninstalling procedures. Note that these assistant scripts may fail to work in customized Linux configurations or unsupported Linux distributions.

Use `install.sh` and `uninstall.sh` script files to install and uninstall (correspondingly) NTFS & HFS combo driver and utilities. The sections below describe the installation procedure in details.

## Unpacking Setup Files

The setup files of the NTFS & HFS for Linux driver are provided in the form of TGZ archives. The archives should be copied on a hard disk and decompressed. Unpack the archive files to directories using, for example, the following commands:

For the NTFS&HFS for Linux driver and utilities:

```
tar zxC /usr/tmp -f /mnt/cdrom/NtfsHfsForLinux/ntfshfslin_drv.tgz  
or
```

```
tar xzf /usr/tmp/ntfshfslin_drv.tgz
```

 – in case you have already copied the TGZ archive to the `/usr/tmp/` directory.

Next, change the current directory to the `/usr/tmp`:

```
cd /usr/tmp
```

Next actions are to build and install the NTFS&HFS for Linux driver and additional utilities.

## Using the INSTALL.SH Assistant Script

The assistant `install.sh` scripts provide easy and flexible way to build combined NTFS & HFS driver, install them in the system and mount all or selected NTFS and HFS+/HFSX partitions which currently exist on local system. Additionally, the script configures all NTFS and HFS+/HFSX partitions to be mounted automatically at system startup.

However, `install.sh` script requires that development tools and kernel sources present on end-user system in their default locations.

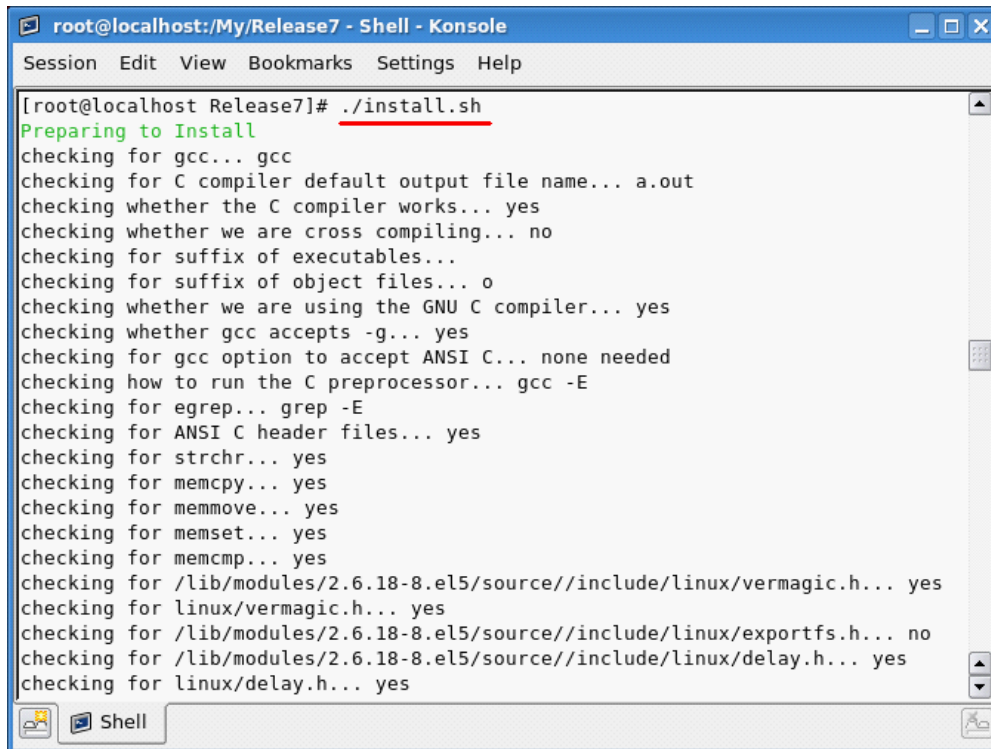
### Installation

Just run the `install.sh` script:

```
./install.sh
```

The assistant script will automatically perform the following actions:

- 1) Detect the Linux type and kernel version;
- 2) Find kernel header files, kernel-config file and libraries needed for building the drivers;
- 3) Build driver and utilities as (binary modules);
- 4) Install driver and utilities;
- 5) Detect all NTFS and HFS+/HFSX partitions on all local hard disks, mount all NTFS/HFS+ partitions;
- 6) Reconfigure the file `/etc/fstab` to automatically mount NTFS and HFS+/HFSX partition at Linux startup;



```
root@localhost:~/My/Release7 - Shell - Konsole
Session Edit View Bookmarks Settings Help

[root@localhost Release7]# ./install.sh
Preparing to Install
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ANSI C... none needed
checking how to run the C preprocessor... gcc -E
checking for egrep... grep -E
checking for ANSI C header files... yes
checking for strchr... yes
checking for memcpy... yes
checking for memmove... yes
checking for memset... yes
checking for memcmp... yes
checking for /lib/modules/2.6.18-8.el5/source/include/linux/vermagic.h... yes
checking for linux/vermagic.h... yes
checking for /lib/modules/2.6.18-8.el5/source/include/linux/exportfs.h... no
checking for /lib/modules/2.6.18-8.el5/source/include/linux/delay.h... yes
checking for linux/delay.h... yes
```

### INSTALL.SH default mode for the NTFS/HFS+ for Linux driver

- The assistant script `install.sh` always names the NTFS/HFS+ for Linux driver module as `ufsd` (it is the abbreviation of the project name Universal File System Driver);
- The assistant script `install.sh` always mounts NTFS/HFS+ partitions to directories named like `"/mnt/ntfs_0"`, `"/mnt/hfsp_1"`, `"/mnt/hfsx_1"` etc.

Now you can mount any NTFS/HFS+ partition: `mount -t ufsd <device> <mount_point>`.

Note: The `/lib/modules/kernel_version/extra/` or `/lib/modules/kernel_version/kernel/fs/ufsd` directory will contain the `ufsd.ko` kernel binary module.

## 3.4 Uninstalling the Drivers

To completely remove the drivers and the utilities from the system, one should dismount all NTFS/HFS+ partitions mounted with the driver, uninstall the drivers and optionally remove binary files.

NTFS&HFS for Linux provides tools for the drivers/utilities uninstall automation.

The assistant script `uninstall.sh` completely removes the drivers/utilities from the system, including unmounting all NTFS/HFS+ partitions.

### Using the UNINSTALL.SH Assistant Script

The assistant script `uninstall.sh` provides the extremely easy and flexible way to deactivate and remove the drivers and utilities from the system. The script performs the correct deactivation,



uninstallation and the complete removing of the driver's and utilities' files.

## Uninstalling

Just run the `uninstall.sh` script:

```
./uninstall.sh
```

The assistant script will automatically perform the following actions:

1. Unmount all currently mounted NTFS/HFS+ partitions. Additionally, the script removes the appropriate mount-points and deletes reference to these partitions from the `fstab`. If some NTFS/HFS+ partitions are in use, the script (for the NTFS&HFS+ driver) will not unmount these partitions. The further script execution is aborted in this case;
2. Deactivate the driver modules. If the drivers is still in use, the further script execution is aborted;
3. Uninstall the drivers;
4. Remove all binary and source files of the driver and utilities.

**Part**

---

**IV**

**Using The Driver**

After building and installing the NTFS&HFS for Linux driver, it can be automatically loaded at the system startup. The driver allows to mount NTFS/HFS+ partitions and to get a plain access to their contents.

## 4.1 Getting started

The goal of this section is to help quickly find out how to use the product. It describes general approach to mounting partitions using UFSD file system driver and helps to avoid common issues. We strongly recommend reading this section before starting using our driver.

To mount volume using UFSD driver, standard mount command is used, with File System (FS) type set to ufsd, e.g.:

```
/ # mount -t ufsd /dev/sda1 /mnt/sda1
```

After this command is executed there can be several mount scenarios for a disk (for more information see [Mount troubleshooting](#)<sup>[61]</sup> subsection):

- The disk is “clean” (without any errors), mounted by the driver and ready to use.
- Disk can’t be mounted. In this case can be several scenarios:
  1. Disk has “dirty” flag set (for more information see [Dirty flag issues](#)<sup>[15]</sup> subsection):
    - Use `chkntfs/chkhfs` utilities with `-a -f` options to check the volume for errors and inconsistencies and fix them (if any). This is recommended approach (see [chkntfs](#)<sup>[27]</sup> or [chkhfs](#)<sup>[57]</sup> subsections);
    - Use ‘force’ mount option (see [Dirty flag issues](#)<sup>[15]</sup> subsection).
  2. The disk is a GPT-partitioned disk – check [GPT issues](#)<sup>[15]</sup> subsection for more information.
  3. Follow other steps on the [Mount troubleshooting](#)<sup>[61]</sup> diagram to find the cause of the issue.

Analyze returned status and check output of (`dmesg | tail`). In case of failure, follow the [Mount troubleshooting](#)<sup>[61]</sup> diagram to find possible causes and try to mount the partition again using the same or different mount options, if needed (see [Mount options](#)<sup>[20]</sup> subsection).

If there is still a problem mounting the partition fill out Paragon's online request form from your user account so we could help you with the issue.

## 4.2 Mounting NTFS/HFS+ Partitions

To gain access to a NTFS/HFS+ partition, use standard `mount` command with a file system type set to ufsd. For example:

```
mount -t ufsd /dev/hdb1 /mnt/ntfs
```

### 4.3 Dirty flag issues

'Dirty' flag is special feature implemented in most of the modern file systems, including NTFS and HFS+. This flag is set after volume is mounted in read/write mode and cleared after volume is correctly unmounted (see notes on 'force' mount option for more information). Without 'dirty' flag it is impossible to tell if given volume was correctly unmounted or not. Detecting incorrectly unmounted volumes helps to detect possible errors as early as possible. Thus, this flag helps to preserve file system consistency. Please note that even in case 'dirty' flag is set on the volume, file system is not necessarily corrupt.

Paragon NTFS and HFS+ for Linux drivers version 8 support 'dirty' flag on both NTFS and HFS+. By default, driver refuses to mount volumes with 'dirty' flag set. Recommended course of action is to check the volume for errors and repair any inconsistencies found using `chkntfs/chkhfs` utility with `-a -f` command line options (see [Additional Utilities](#) <sup>[25]</sup> section). Run with `-a` command line option, the utilities check dirty flag state and in case it is set, they performs all necessary checks. If 'dirty' flag is not set, file system checking utilities exits immediately. If `-f` command line option is specified, the utilities repair any errors or inconsistencies that they find and finally clear 'dirty' flag. This approach is similar to the way Windows and MacOS handle 'dirty' volumes. See corresponding sections on NTFS and HFS utilities and 'File system checking utilities return codes' section for more information.

To make driver mount dirty volumes without checking for possible errors and correcting them, 'force' mount option can be used (while it is not recommended). This way, 'dirty' flag is not cleared and any possibly existing errors or inconsistencies are not fixed. 'Dirty' flag will remain set until volume is checked for errors using Paragon `chkntfs/chkhfs` with `-f` command line option or using Windows `chkdsk` utility with `/f` switch or MacOS Disk Utility (for NTFS and HFS+ volumes, respectively).

### 4.4 GPT issues

Some Linux Kernels do not behave correctly when there is EFI partition on GPT-partitioned devices. This is most often the case with HDDs partitioned using MacOS Disk Utility.

This leads to seemingly wrong operation of UFSD driver(s) that refuse to mount partition. In that case `fdisk` may report that there is only one EFI partition on the device, ignoring some or all of the following NTFS and/or HFS+ partition(s). To work around the issue, attention must be paid to volume type reported by `fdisk -l` command on GUID-partitioned disks.

Example:

```
/ # fdisk -l
```

```
Disk /dev/sda: 80.0 GB, 80026361856 bytes
255 heads, 63 sectors/track, 9729 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sda1		1	9730	78150743+	ee	EFI GPT

`fdisk` reports that `/dev/sda` only contains single EFI partition that spans via entire disk. EFI

partitions are formatted as FAT32 and therefore cannot be mounted by NTFS/HFS+ driver(s). Nevertheless, mounting partition `/dev/sda2` to `/mnt/hda` succeeds:

```
/ # mount -t ufsd /dev/sda2 /mnt/hda
```

And after that mount command issued without arguments lists, among others, mounted partition `/dev/sda2` that was not listed by `fdisk -l` (marked with red below).

```
/dev/root on / type squashfs (ro)
none on /dev type devfs (rw)
none on /proc type proc (rw,nodiratime)
devpts on /dev/pts type devpts (rw)
none on /sys type sysfs (rw)
none on /tmp type ramfs (rw)
/dev/mtdblock/2 on /usr/local/etc type yaffs (rw,noatime)
/dev/rd/0 on /mnt/rd type vfat
(rw,nodiratime,fmask=0022,dmask=0022,codepage=cp437,ioccharset=iso8859-1)
/dev/sda2 on /mnt/hda type ufsd
(rw,nodiratime,nls=iso8859-1,uid=0,gid=0,fmask=22,dmask=22,nocase)
/dev/scsi/host2/bus0/target0/lun0/part1 on /tmp/usbmounts/sdb1 type ufsd
(ro,nodiratime,nls=utf8,uid=0,gid=0,fmask=0,dmask=0)
```

Older Linux kernels do not support GPT at all. To work around this issue, Paragon NTFS&HFS driver for Linux can read and parse GUID partitioning table and use it to mount the first HFS+ partition on that disk. To mount the first HFS+ volume on GPT disk the following command may be used:

```
/ # mount -t ufsd /dev/sda /mnt/hda
```

Please note that entire disk device is specified instead of specific partition. This approach may only be used to mount the first HFS+ volume on GPT disks.

## 4.5 Lazy open

'Lazy open' is a feature of Paragon NTFS&HFS file system driver. This feature allows to improve performance of directory enumeration operations for the price of consistency of some information provided to by the file system driver to the Kernel during directory enumeration operations (like `opendir/readdir`). The 'lazy open' behavior can be observed with native Windows NTFS driver. A lot of metadata on NTFS is duplicated and triplicated, but up-to-date information for a file always presents in Mft record associated with the file. As NTFS on-disk structures for directories are stored separately from Mft records for files, obtaining up-to-date information on a file can be expensive operation in terms of performance. The information includes access time, number of hardlinks etc.

This feature is enabled by default. To disable the feature, `nolazy` [mount option](#)<sup>[20]</sup> can be used.

### Performance impact

Test environment: Ubuntu 10.04 Linux on x86 Desktop machine. UFSd version 8.1.044. Test directory contains 10 subdirectories, each, in turn, containing 10000 'touch'ed files.

Test command: `time ls -Rlah /mnt/ntfs/test`

Mode	Lazy open enabled (default)	Lazy open disabled (nolazy mount option)
Mount options	rw,umask=0000,fmask=000,dmask=000	nolazy,rw,umask=0000,fmask=000,dmask=000
Test execution time	real 6.70 user 1.28 sys 4.46	real 13.97 user 1.06 sys 11.61

## 4.6 Issues with large HDDs

Though our driver supports partitions larger than 2 Tb (tested on 16 Tb partitions on real hardware and on 25 Tb partitions in virtual environment), not all versions of Linux Kernels support block devices larger than 2 Tb on all possible interfaces. E.g. Ubuntu 10.04 does not support 2.5 Tb SATA HDD attached via USB->SATA converter, while the same HDD with the same converter is mounted OK on Windows 7 and the same HDD connected to Ubuntu 10.04 via SATA interface can be mounted and used successfully.

If there is similar issue, please perform the test cases described above to make sure where the root cause of the issue is (in Paragon's driver or in Linux Kernel).

## 4.7 Unmounting NTFS/HFS+ Partitions

To unmount a NTFS partition, use the standard command `umount`. For example:

```
umount /dev/hdb1
```

## 4.8 Choosing the codepage/charset for NTFS/HFS+ Partitions

The format of filenames on NTFS/HFS+ partitions differs from text standard presentation used in Linux. To accommodate NTFS/HFS+ standards to Linux ones, character translation is required. The character translation uses **charset** or **codepage** information for correct translation non-English characters between NTFS/HFS+ and Linux.

Unfortunately Linux is unable to automatically detect NTFS/HFS+ **codepage/charset** settings. For this reason, the user must assign character set for filenames translation manually.

The standard Linux command **mount** allows choosing the character set that is used for the filenames translation, the **iocharset** parameter is used for this purpose.

**iocharset** parameter of **install.sh** script provides the ability to define the character set for all automatically mounted partitions. One should realize that character set assigned to the driver should conform to the actual locale settings in Linux. Otherwise, non-English filenames will remain unreadable.

**Examples:**

1. Mounting a partition:

```
mkdir /mnt/test  
mount -t ufsd /dev/hda6 /mnt/test
```

2. Dismounting a partition:

```
umount /mnt/test
```

3. Mounting partition in read-only mode:

```
mount -t ufsd -o ro /dev/hda6 /mnt/test
```

4. Choosing the character set to be used with NTFS partitions, when installing Linux driver:

```
./install.sh --iocharset=utf8
```

5. Choosing character set to be used with NTFS/HFS+ when mounting partitions manually:

```
mount -t ufsd -o iocharset=koi8-r /dev/hdb1 /mnt/test
```

**Part**

---



**Mount options**



This section describes mount options for mounting NTFS and HFS+ partitions.

## 5.1 Mount options

### SYNOPSIS

```
mount -t ufsd [-o options] device mount_point
```

Option	NTFS	HFS+	Expected behavior and examples
<b>iocharset</b> or <b>nls</b> or <b>codepage</b>	+	+	<p><b>-o iocharset={NAME1} [, iocharset={NAME2}]</b>  <b>-o nls={NAME1} [, nls={NAME2}]</b>  <b>-o codepage={NAME1} [, codepage={NAME2}]</b></p> <p>The NTFS/HFS+ file systems store all file/directory names in Unicode format (UTF-16), which can represent any character from any language. In case none of these options is set, the default codepage will be used (CONFIG_NLS_DEFAULT). If none of the specified codepages exist on the system, the default codepage will be used again. This option informs the driver how to interpret path strings and translate them to Unicode and back. Up to 8 different code pages can be specified. The driver tries to use the codepages from specified list in order until it manages to translate all the characters in the string. If none of the specified codepages allows to translate all the characters, Kernel's default codepages is used.</p> <p>Note:</p> <ul style="list-style-type: none"> <li>Paragon driver uses extended UTF-8 for Unicode number U+10000 characters support when 'utf8' is specified.</li> <li>Codepage, nls and iocharset mount option must be used in the form: codepage=... nls=cp... iocharset=cp...</li> </ul> <p>Examples:</p> <ul style="list-style-type: none"> <li><b>codepage=950</b></li> <li><b>nls=cp950</b></li> <li><b>iocharset=cp950</b></li> </ul>
<b>nocase</b>	+		<p><b>-o nocase</b></p> <p>All file and directory operations (open, find, rename) are case insensitive. Casing is preserved in the names of existing files and directories.</p>
<b>showmeta</b>	+	+	<p><b>-o showmeta</b></p> <p>Use this parameter to show all meta-files (System Files) on a mounted NTFS/HFS+ partition. By default, all meta-files are hidden.</p>

Option	NTFS	HFS+	Expected behavior and examples
<b>noatime</b>	+	+	<b>-o noatime</b> All files and directories will not update their last access time attribute if a NTFS/HFS+ partition is mounted with this parameter. This option can speed up file system operation.
<b>uid</b>	+	+	<b>-o uid={USERID}</b> By default all files on a mounted NTFS/HFS+ volume are owned by root. By specifying the uid parameter you can set an owner of files. The userid can be any name from /etc/passwd, or any number representing a user id.
<b>gid</b>	+	+	<b>-o gid={GROUPID}</b> By default all files on a mounted NTFS/HFS+ volume are owned by group root. By specifying the gid parameter you can set a owner group of the files. The groupid can be any name from /etc/group, or any number representing a group id.
<b>umask</b>	+	+	<b>-o umask={VALUE}</b> The default permissions given to a mounted NTFS/HFS+ volume are <b>rwX-----</b> (for security reasons). The umask option controls these permissions for files/directories created after the volume is mounted.  <b>mount -t ufsd /dev/hda1 /mnt/ntfs_0 -o umask=0222</b>
<b>fmask</b> <b>dmask</b>	+	+	<b>-o fmask={VALUE}</b> <b>-o dmask={VALUE}</b> umask option changes the permissions for new created files and directories; fmask is applied to files; dmask to directories that already exist on a mounted volume. The effect of these options can be combined. To mount Samba, FTP or NFS shares the combination of <b>umask=000 , fmask=000 , dmask=000</b> is usually specified.
<b>ro</b>	+	+	To mount an NTFS/HFS+ volume in read-only mode.
<b>bestcompr</b>	+		Instructs the driver to use highest compression level when writing compressed files. High CPU-load.
<b>nobuf</b>	+	+	Disables buffered read/write operations for metadata and directories. Useful option for embedded device with little memory (<64MB).  Note: this option is not supported on driver versions 8.2 and higher
<b>sparse</b>	+		Create new files as "sparse". This feature allows creating holes inside new created files (avoids filling unwritten space with zeroes). This option is useful in case NTFS partition is used for BitTorrent downloads.

Option	NTFS	HFS+	Expected behavior and examples
<b>force</b>	+	+	<p>Not recommended for use.</p> <p>Forces the driver to mount partitions even if 'dirty' flag (volume dirty) is set. It is recommended to use Paragon or OS-specific file system checking utility before mounting 'dirty' partitions to reset the 'dirty' flag.</p> <p>Note that if 'dirty' volume was mounted with 'force' mount option, dirty flag will not be cleared when volume is unmounted using <b>umount</b> command.</p>
<b>nohidden</b>	+		Files with the Windows-specific HIDDEN attribute will not be shown under Linux.
<b>sys_immutable</b>	+		Files with the Windows-specific SYSTEM attribute will be marked as system immutable files.
<b>clump</b>	+		<p><b>-o clump={size}</b></p> <p>Driver will pre-allocate space up to "size" in Kbytes during file extension operation. Preallocated space will be aligned up to the cluster size. This space will be preallocated, but the file size information will only show the real written size. This speeds up file write operations if write function is called with small buffer size, but may consume extra free space files.</p>
<b>acl*</b> <b>acl={1 0}*</b>	+	+	<p>Support POSIX ACLs (Access Control Lists). Effective if supported by Kernel.</p> <p>The option specified as <b>acl</b> or <b>acl=1</b> enables support for POSIX ACLs; <b>acl=0</b> disables it. Supported if driver is built with support for ext2-like handling of file/directory permissions — see .</p>
<b>user_xattr*</b> <b>user_xattr={1 0}*</b>	+	+	<p>Support user.* extended attributes. Effective if supported by Kernel.</p> <p>The options specified as <b>user_xattr</b> or <b>user_xattr=1</b> enables support for user.* extended attributes; <b>user_xattr=0</b> disables it.</p>
<b>bias*</b>	+		<p>Specify time difference between UTC and local time in minutes. Useful in case local system does not support time zone settings.</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>• Eastern time zone: +300</li> <li>• Paris, Berlin: -60</li> <li>• Moscow: -180</li> </ul>

Option	NTFS	HFS+	Expected behavior and examples
<b>nolazy**</b> <b>nolazy={1 0}</b> <b>**</b>	+		When used in the form <b>nolazy</b> or <b>nolazy=1</b> , the option disables 'lazy open' behavior of the driver (which is the default). See <a href="#">Lazy open</a> <sup>[16]</sup> subsection for more details.
<b>delalloc***</b>	+	+	Space allocation is delayed until Kernel is about to flush cache pages to disk. This improves performance and reduce file fragmentation in case write operations are performed to several files simultaneously in small chunks. Only influences files opened in buffered mode.

\* Supported in driver version 8.1.023 and later and in 8.2 or later.

\*\* Supported in driver version 8.3 and later.

\*\*\* Supported in driver version 8.5 and later

**Part**

---

**VI**

**Additional Utilities**

USER MANUAL

Additional utilities for Paragon NTFS&HFS for Linux provide the ability to check integrity and create NTFS/HFS+ volumes on block devices. Additional NTFS utilities allow to defragment, wipe, and perform many NTFS file system related tasks and copy (backup) files, saving all NTFS-specific data and attributes, between NTFS and native Linux file systems. Additional utilities for Paragon NTFS&HFS for Linux were developed with Paragon UFSD SDK.

## 6.1 NTFS utilities

There are 10 additional utilities for NTFS:

- [infntfs](#)<sup>[25]</sup> — show detailed information about NTFS partitions;
- [chkntfs](#)<sup>[27]</sup> — check NTFS partition for integrity and (optionally) fix errors;
- [mkntfs](#)<sup>[28]</sup> — format any partition as NTFS under Linux;
- [dfntfs](#)<sup>[30]</sup> — defragment a NTFS volume;
- [wipe](#)<sup>[33]</sup> — fill with zeros free space on a NTFS/FAT volume;
- [mftpack](#)<sup>[34]</sup> — pack/truncate MFT (Master File Table) on a NTFS volume;
- [hdlnk](#)<sup>[36]</sup> — enumerate all hard links on a NTFS volume;
- [junction](#)<sup>[53]</sup> — show reparse points on a NTFS volume;
- [fsutil](#)<sup>[38]</sup> — perform many NTFS file system related tasks. Powerful utility;
- **cpntfs** — creates an archive of the NTFS volume or separate files/directories including all streams and attributes.

### 6.1.1 infntfs

INFNTFS Utility - Show information about NTFS Volumes.

#### Name

**infntfs** — is intended for showing and changing common information about NTFS volumes.

#### Synopsis

```
infntfs [options] device
```

```
E.g.: infntfs --trace --verbose --label "New Volume" --dirty clear --serial  
AAAAAAA-BBBBBBBB /dev/hdb1;
```

```
E.g.: infntfs /dev/hdb1.
```

#### Options

label label	Set new volume label
-------------	----------------------

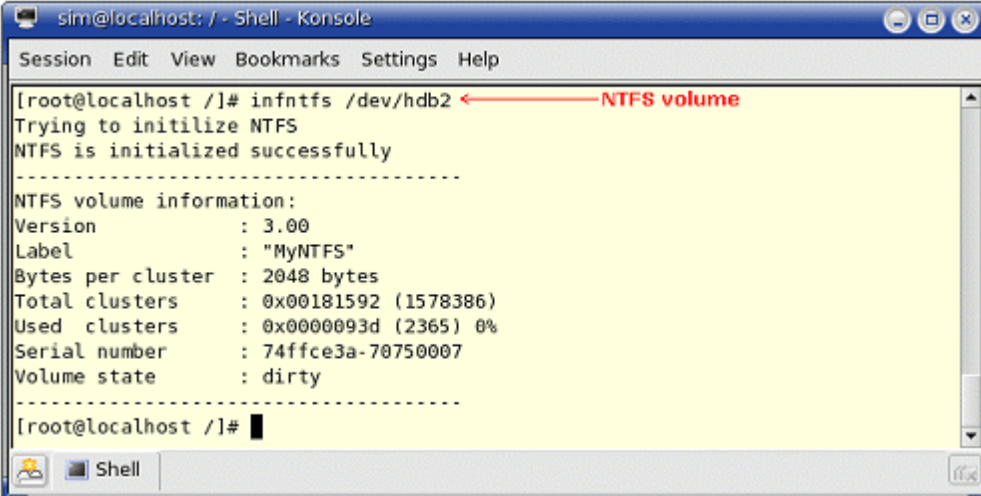
<b>dirty set</b>	Set dirty flag
<b>dirty clear</b>	Clear dirty flag
<b>serial lo-high</b>	Set a new serial number (in hex)
<b>trace</b>	Turn on UFSD trace
<b>verbose</b>	Explain what is being done
<b>help</b>	Display this help

## Description

**infntfs** shows NTFS volume label, used space, dirty flag, version, serial number and allows to change NTFS volume label, dirty flag and serial number.

## Screenshots

1. Showing common information about NTFS volume:



```
sim@localhost: / - Shell - Konsole
Session Edit View Bookmarks Settings Help

[root@localhost /]# infntfs /dev/hdb2 ← NTFS volume
Trying to initilize NTFS
NTFS is initialized successfully
-----
NTFS volume information:
Version       : 3.00
Label        : "MyNTFS"
Bytes per cluster : 2048 bytes
Total clusters : 0x00181592 (1578386)
Used clusters  : 0x0000093d (2365) 0%
Serial number  : 74ffce3a-70750007
Volume state   : dirty
-----
[root@localhost /]#
```

2. Changing common information about the NTFS volume:

```
sim@localhost: / - Shell - Konsole
Session Edit View Bookmarks Settings Help

[root@localhost /]# infntfs --label NewLabel --dirty clear --serial 85ffde4c-777
77777 /dev/hdb2
Trying to initialize NTFS
NTFS is initialized successfully
-----
NTFS volume information:
Version          : 3.00
Label            : "MyNTFS"
Bytes per cluster : 2048 bytes
Total clusters   : 0x00181592 (1578386)
Used clusters    : 0x0000093d (2365) 0%
Serial number    : 74ffce3a-70750007
Volume state     : dirty
-----
NTFS volume information is updated successfully!
-----
NTFS volume information:
Version          : 3.00
Label            : "NewLabel"
Bytes per cluster : 2048 bytes
Total clusters   : 0x00181592 (1578386)
Used clusters    : 0x0000093d (2365) 0%
Serial number    : 85ffde4c-77777777
Volume state     : clean
-----
[root@localhost /]#
```

## 6.1.2 chkntrfs

CHKNTFS Utility - Perform consistency checks on a NTFS volume.

### Name

**chkntrfs** - provide consistency checking of a NTFS volume and fixing errors.

### Synopsis

```
chkntrfs device [options]
```

E.g.: **chkntrfs /dev/hdb1**

### Options

<b>-f</b>	Fix errors on the disk.
<b>-a</b>	Perform checks only if 'dirty' flag is set.
<b>-h</b>	Display this help.



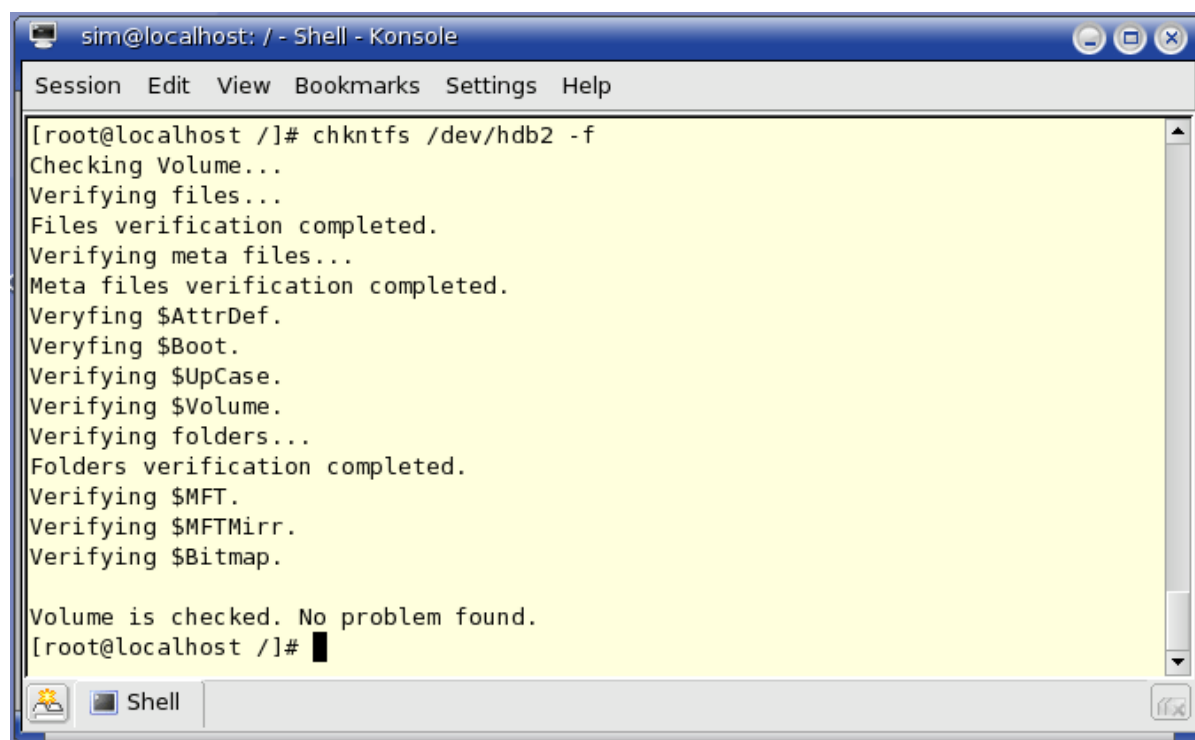
<b>--trace</b>	Turn on UFSD trace.
<b>--verbose</b>	Explain what is being done.
<b>--version</b>	Show the version and exit.

## Description

**chkntfs** creates and displays a status report about a NTFS file system. Chkntfs also lists and corrects errors on the disk, if any (**-f** flag must be specified).

## Screenshot

Verifying and fixing errors on the specified partition:



```
sim@localhost: / - Shell - Konsole
Session Edit View Bookmarks Settings Help

[root@localhost /]# chkntfs /dev/hdb2 -f
Checking Volume...
Verifying files...
Files verification completed.
Verifying meta files...
Meta files verification completed.
Verifying $AttrDef.
Verifying $Boot.
Verifying $UpCase.
Verifying $Volume.
Verifying folders...
Folders verification completed.
Verifying $MFT.
Verifying $MFTMirr.
Verifying $Bitmap.

Volume is checked. No problem found.
[root@localhost /]#
```

### 6.1.3 mkntfs

MKNTFS Utility - Create a NTFS volume on a partition.

## Name

**mkntfs** - create a NTFS volume (1.2, 3.0, 3.1 (Windows NT 4.0/2000/XP/2003/Vista/7) file system) on a user specified (block) device under Linux OS.

## Synopsis

```
mkntfs [options] device
```

E.g.: `mkntfs /dev/hdb1`

## Options

<code>-v:label</code>	Specify the volume label.
<code>-q</code>	Perform a quick format.
<code>-c</code>	Files created on the new volume will be compressed by default.
<code>-a:size</code>	Override the default allocation unit size.  Default settings are strongly recommended for general use.  NTFS supports 512, 1024, 2048, 4096, 8192, 16K, 32K, 64K.  NTFS compression is not supported for allocation unit sizes above 4096.
<code>-f</code>	Force the format without confirmation.
<code>-s:start</code>	Specify "hidden" sectors in the boot area.
<code>-g:tracks:sectors</code>	Specify the disk geometry that should be written in the boot area.  "tracks" – Specify the number of tracks per disk side.  "sectors" – Specify the number of sectors per track.  The most known geometries are:  NORMAL: 63 sectors per track and 15(16) tracks per cylinder.  LBA: 63 sectors per track and 255 tracks per cylinder.  In general Windows uses the LBA geometry ( <code>-g: 255 : 63</code> )  If <code>-g</code> is not specified this program gets geometry from Linux.
<code>--help</code>	Display this help.
<code>--trace</code>	Turn on UFSD trace.
<code>--verbose</code>	Explain what is being done.
<code>--version</code>	Show the version and exit.

## Description

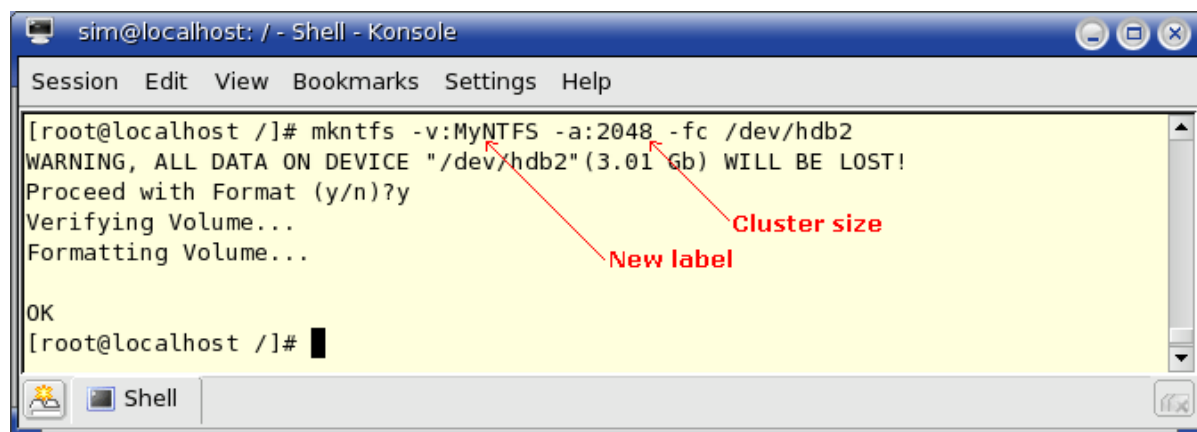
**mkntfs** is a standalone utility that allows to format NTFS partitions under Linux. It is used to create a NTFS 1.2, 3.0, 3.1 (Windows NT 4.0/2000/XP/2003/Vista/7) file system on a device (usually a disk partition).

Note: **mkntfs** doesn't change the Master Boot Record (MBR) when formatting a partition. It follows that most of Linux commands (like `fdisk -l`) will not define that the partition's files

system was changed to a NTFS one.

## Screenshots

Making NTFS partition:

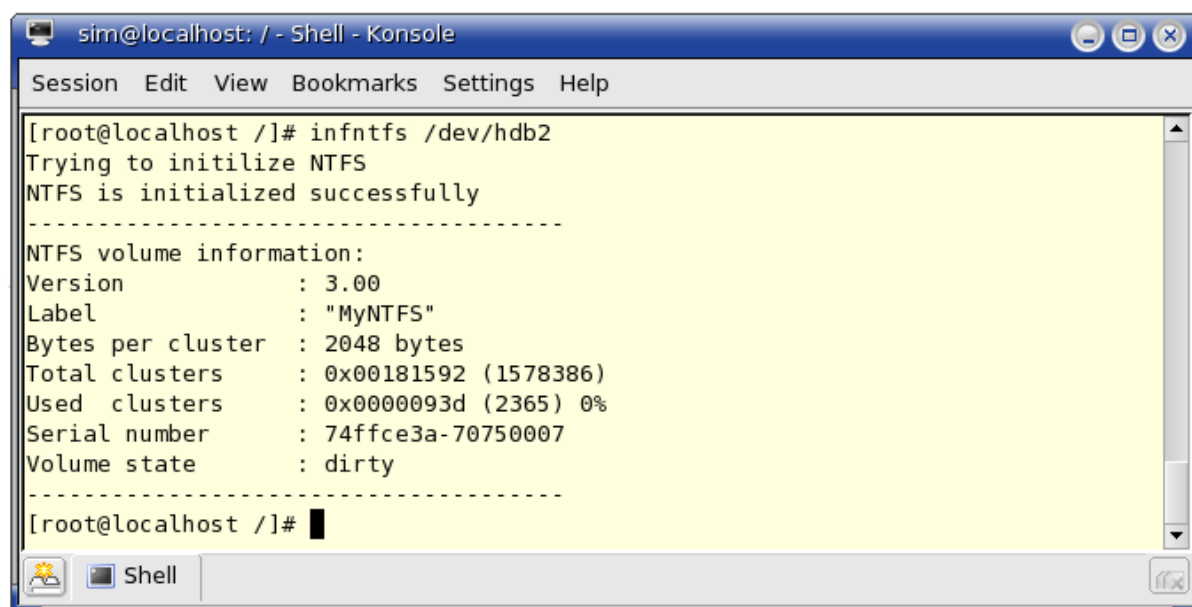


```
sim@localhost: / - Shell - Konsole
Session Edit View Bookmarks Settings Help

[root@localhost /]# mkntfs -v:MyNTFS -a:2048 -fc /dev/hdb2
WARNING, ALL DATA ON DEVICE "/dev/hdb2" (3.01 Gb) WILL BE LOST!
Proceed with Format (y/n)?y
Verifying Volume...
Formatting Volume...

OK
[root@localhost /]#
```

Result:



```
sim@localhost: / - Shell - Konsole
Session Edit View Bookmarks Settings Help

[root@localhost /]# infntfs /dev/hdb2
Trying to initialize NTFS
NTFS is initialized successfully
-----
NTFS volume information:
Version          : 3.00
Label            : "MyNTFS"
Bytes per cluster : 2048 bytes
Total clusters   : 0x00181592 (1578386)
Used clusters    : 0x0000093d (2365) 0%
Serial number    : 74ffce3a-70750007
Volume state     : dirty
-----

[root@localhost /]#
```

### 6.1.4 dfntfs

DFNTFS Utility - Defragment a NTFS Name

**dfntfs** – defragment a NTFS volume (1.2, 3.0, 3.1 (Windows NT 4.0/2000/XP/2003/Vista/7) file system) on a user specified (block) device under Linux OS.

## Synopsis

**dfntfs** [options] device

E.g.: `dfntfs -s- /dev/hdb1`

## Options

<code>-t+</code>	Creation time increasing (sort files and directories according to their “creation time” attribute in ascending order);
<code>-t-</code>	Creation time decreasing;
<code>-s+</code>	File size increasing (sort file and directories according to their “file size” attribute in ascending order);
<code>-s-</code>	File size decreasing;
<code>-d+</code>	Directory first (place directories ahead files);
<code>-d-</code>	Directory last;
<code>-l+</code>	Start cluster increasing (the order (according to the start cluster) of files and directories will be preserved and they will be placed continuously);
<code>-l-</code>	Start cluster decreasing (files and directories will be placed in the reserved sequence order and continuously);
<code>--help</code>	Display help;
<code>--trace</code>	Turn on UFSD trace;
<code>--verbose</code>	Explain what is being done;
<code>--version</code>	Show version and exit.

## Description

Defragmentation is the process of rewriting parts of a file to contiguous sectors on a hard disk to increase the speed of access and retrieval. When files are updated, the computer tends to save these updates on the largest continuous space on the hard disk, which is often on a different sector than the other parts of the file. When files are thus fragmented, the computer must search the hard disk each time the file is opened to find all of the file’s parts, which slows down response time.

This `dfntfs` utility provides the necessary functionality for the defragmentation of NTFS partitions.

## Screenshots

Let’s defragment a NTFS partition in the following way:

1. Place directories ahead files;
2. Sort file and directories according to their "file size" attribute in descending order.volume.

```
sim@localhost: / - Shell - Konsole
Session Edit View Bookmarks Settings Help

[root@localhost /]# fdisk -l

Disk /dev/hda: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1    *           1           719     5775336    83   Linux
/dev/hda2                720        1305     4707045     5   Extended
/dev/hda5                720         859     1124518+    82   Linux swap / Solaris
/dev/hda6                860        1305     3582463+    83   Linux

Disk /dev/hdb: 42.9 GB, 42949672960 bytes
255 heads, 63 sectors/track, 5221 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1    *           1          191     1534176     7   HPFS/NTFS
[root@localhost /]# dfntfs -d+ -s- /dev/hdb1
```

```
sim@localhost: / - Shell - Konsole
Session Edit View Bookmarks Settings Help

44,      5, 104 "/WINDOWS/PCHEALTH/HELPCTR/DataColl/CollectedData_36.xml"
45,      5, 63  "/WINDOWS/PCHEALTH/HELPCTR/DataColl/CollectedData_38.xml"
46,      5, 32  "/WINDOWS/Fonts/"
47,      5, 28  "/WINDOWS/inf/sti.PNF"
48,      5, 16  "/WINDOWS/comsetup.log"
49,      5, 8   "/WINDOWS/ntdtcsetup.log"
Bytes per volume   : 1570995712
Used bytes         : 679011328 (43%)
Free space fragments: 2787
Biggest free block : 240 Mb
Total files        : 6174
Fragmented files   : 965
Total folders      : 524
Fragmented folders : 21
Total MFT records  : 9602
Used MFT records   : 6719
Fragments per MFT  : 3
Defragging NTFS in memory...
Defragging NTFS on disk...
Analyzing NTFS...
52% /WINDOWS/system32/drivers/arp1394.sys
```

```
sim@localhost: / - Shell - Konsole
Session Edit View Bookmarks Settings Help

Used MFT records      : 6719
Fragments per MFT    : 3
Defragging NTFS in memory...
Defragging NTFS on disk...
Analyzing NTFS...
Bytes per volume      : 1570995712
Used bytes            : 679011328 (43%)
Free space fragments: 1
Biggest free block    : 850 Mb
Total files           : 6174
Fragmented files      : 0
Total folders         : 524
Fragmented folders    : 0
Total MFT records     : 9602
Used MFT records      : 6704
Fragments per MFT     : 1
Mapped clusters       : 325392
Remapped clusters     : 0

OK
[root@localhost /]#
```

## 6.1.5 wipe

WIPE Utility – Fill with zeros free space on a NTFS/FAT volume.

### Name

**wipe** – zero free space (unused clusters and tails of files/directories) on NTFS/FAT volumes.

### Synopsis

```
wipe [options] device
```

E.g.: `wipe -c -t /dev/hdb1`

### Options

<b>-c</b>	Wipe unused clusters;
<b>-t</b>	Wipe tails of files/directories;
<b>--help</b>	Display this help;
<b>--trace</b>	Turn on UFSD trace;
<b>--verbose</b>	Explain what is being done;
<b>--version</b>	Show the version and exit.

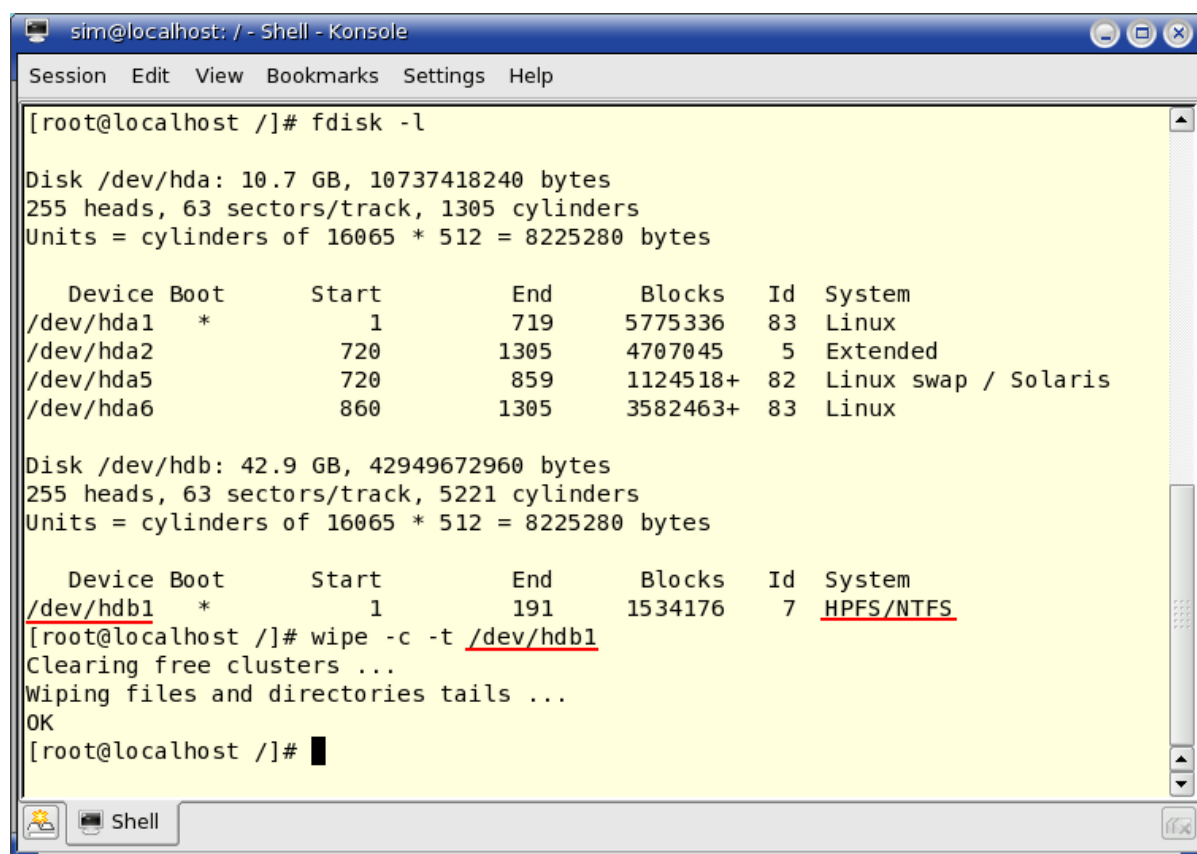
## Description

Wipe Partition function allows irreversibly destroying all contents of a partition by overwriting all of its sectors with unused data (zeroes).

This function can be used, if a user intends:

- destroying on-partition data without an ability of restoration any of their parts;
- reselling or renting a workable hard disk;
- surely exclude any traces of old data on a newly formatted partition;
- destroying non-standard protection/registration/deactivation hidden marks made by some software.

## Screenshot



```
sim@localhost: / - Shell - Konsole
Session Edit View Bookmarks Settings Help

[root@localhost /]# fdisk -l

Disk /dev/hda: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1    *           1           719     5775336    83  Linux
/dev/hda2                720        1305     4707045     5  Extended
/dev/hda5                720           859     1124518+    82  Linux swap / Solaris
/dev/hda6                860        1305     3582463+    83  Linux

Disk /dev/hdb: 42.9 GB, 42949672960 bytes
255 heads, 63 sectors/track, 5221 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1    *           1          191     1534176     7  HPFS/NTFS
[root@localhost /]# wipe -c -t /dev/hdb1
Clearing free clusters ...
Wiping files and directories tails ...
OK
[root@localhost /]#
```

### 6.1.6 mftpack

MFTPACK Utility – Pack/truncate MFT (Master File Table) on a NTFS volume.

#### Name

**mftpack** – pack MFT records and/or truncate MFT on NTFS volumes.

## Synopsis

**mftpack** [options] device

E.g.: **mftpack -c -t /dev/hdb1**

## Options

<b>-c</b>	Compact MFT records (move tail records to the head of \$MFT);
<b>-t</b>	Truncate MFT (remove unused tail records);
<b>--help</b>	Display this help;
<b>--trace</b>	Turn on UFSD trace;
<b>--verbose</b>	Explain what is being done;
<b>--version</b>	Show the version and exit.

## Description

Master File Table (MFT) is a relational database that consists of rows of file records and columns of file attributes (size, time and date stamps, permissions, data contents and so forth). It contains at least one entry for every file on an NTFS volume, including the MFT itself. MFT is similar to a FAT table in a FAT file system. In the course of time the MFT file can also be fragmented, bulky and inefficiently take up too much disk space, thus slowing down the speed at which data is accessed. The **mftpack** utility provides with all necessary functionality to defragment MFT. Please note this utility may release considerable disk space that MFT inefficiently takes up.

## Screenshot



```

sim@localhost: / - Shell - Konsole
Session Edit View Bookmarks Settings Help

Disk /dev/hda: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1           719     5775336    83  Linux
/dev/hda2             720          1305     4707045     5  Extended
/dev/hda5             720           859     1124518+    82  Linux swap / Solaris
/dev/hda6            860          1305     3582463+    83  Linux

Disk /dev/hdb: 42.9 GB, 42949672960 bytes
255 heads, 63 sectors/track, 5221 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1   *           1           191     1534176     7  HPFS/NTFS

[root@localhost /]# mftpack -c -t /dev/hdb1
Total MFT Records : 9602
Records to compact: 2625
Compacting MFT...
Truncating MFT: 9602 => 6712
OK
[root@localhost /]#
    
```

## 6.1.7 hdlnk

HDLNK Utility – Enumerate all hard links on NTFS volume.

### Name

**hdlnk** – enumerate all hard links on NTFS volumes and display to stdout (standard output).

### Synopsis

**hdlnk** device [options]

E.g.: **hdlnk** /dev/hdb1 -o report.txt

### Options

-o	A file name should be specified (where all hard links must be enumerated). Stdout is by default.
-v	Explain what is being done;
-h	Display this help;

<code>--trace</code>	Turn on UFSD trace;
<code>--version</code>	Show the version and exit

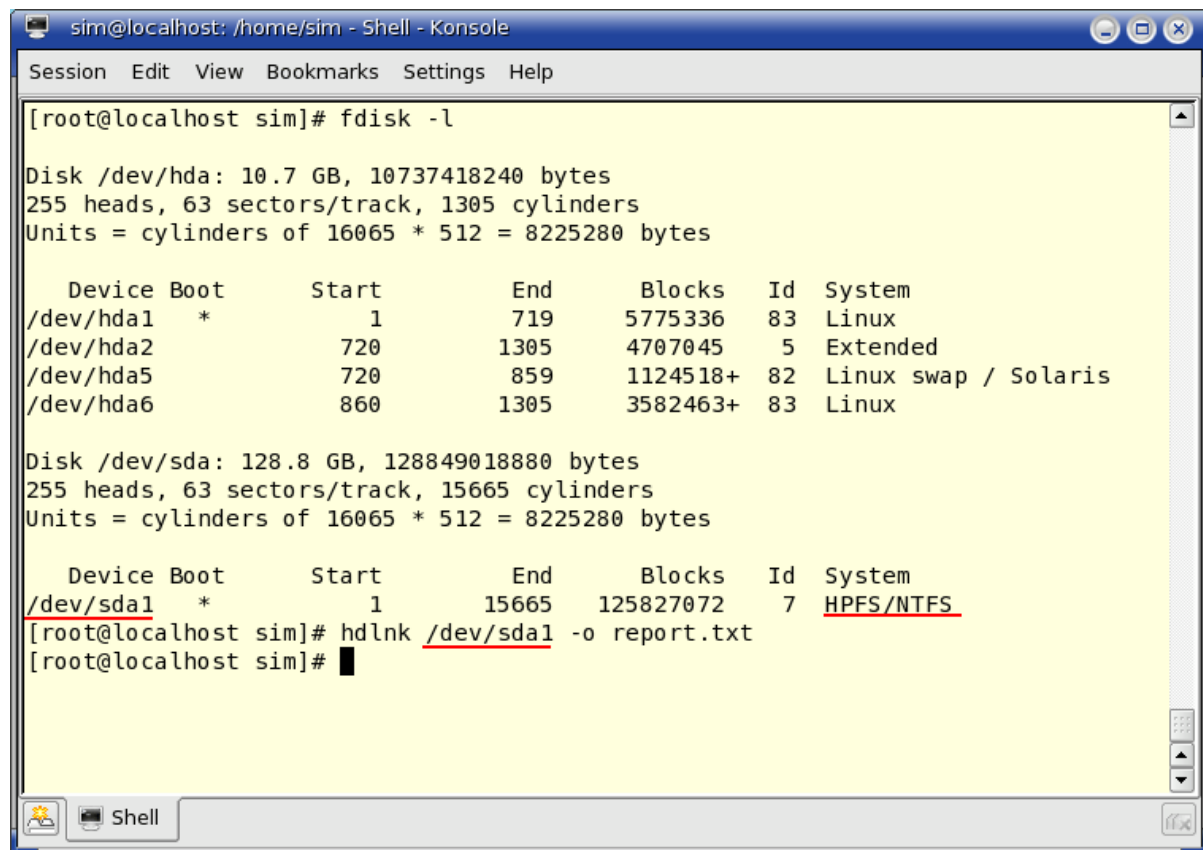
## Description

A hard link is a directory entry for a file. Every file can be considered to have at least one hard link. On NTFS volumes, each file can have multiple hard links, and thus a single file can appear in many directories (or even in the same directory with different names). Because all of the links reference the same file, programs can open any of the links and modify the file. A file is deleted from the file system only after all links to it have been deleted. After you create a hard link, programs can use it like any other file name.

All actual data on disk that have more than one hard link will be enumerated using the `hdlnk` utility.

## Screenshots

Let's enumerate all hard links on a Vista NTFS partition. The list of hard links must be written to a `report.txt` file (the file doesn't exist).



```
sim@localhost: /home/sim - Shell - Konsole
Session Edit View Bookmarks Settings Help

[root@localhost sim]# fdisk -l

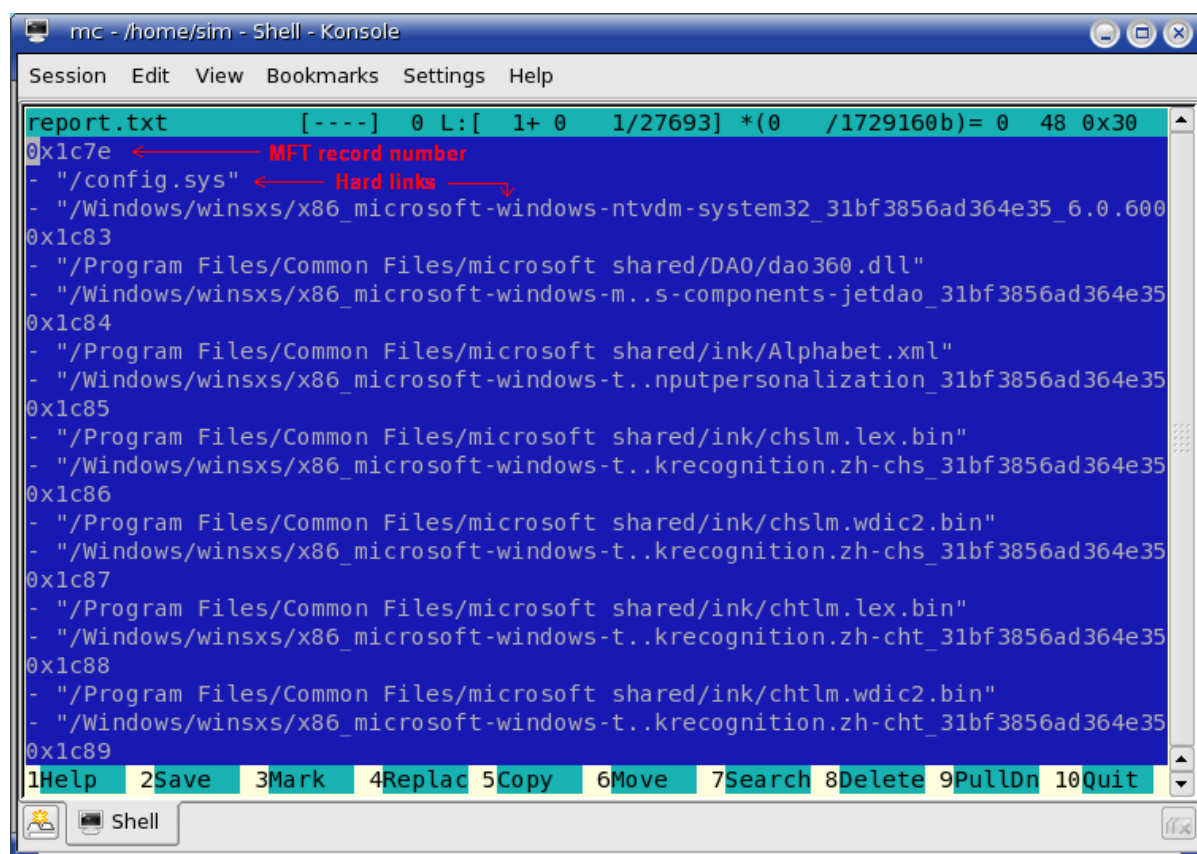
Disk /dev/hda: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1  *           1           719     5775336    83  Linux
/dev/hda2             720          1305     4707045     5  Extended
/dev/hda5             720           859     1124518+    82  Linux swap / Solaris
/dev/hda6             860          1305     3582463+    83  Linux

Disk /dev/sda: 128.8 GB, 128849018880 bytes
255 heads, 63 sectors/track, 15665 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1  *           1        15665    125827072    7  HPFS/NTFS
[root@localhost sim]# hdlnk /dev/sda1 -o report.txt
[root@localhost sim]#
```

The `report.txt` file:



### 6.1.8 fsutil

## FSUTIL Utility – Powerful Utility to Perform NTFS File System Related Tasks

## Name

**fsutil** – NTFS file system utility for advanced users (Windows XP fsutil analogue).

## Description

**Fsutil** is a Linux utility that you can use to perform many NTFS file system related tasks, such as managing file system information, compression, hardlinks and etc. Because fsutil is quite powerful, it should only be used by advanced users who have a thorough knowledge of NTFS file system.

Note: To view help for the available subcommands, type `fsutil`, type the subcommand, and then type `help` (that is, `fsutil subcommand help`).

## Synopsis

**fsutil <subcommand>**

## Subcommands

<b>behavior</b>	Control file system behavior.
<b>dirty</b>	Manage volume dirty bit.
<b>file</b>	File specific commands.
<b>fsinfo</b>	File system information.
<b>hardlink</b>	Hardlink namagement.
<b>objectid</b>	Object ID management.
<b>compress</b>	Manage compression.
<b>streams</b>	Streams management.
<b>sparse</b>	Sparse file control.

## Fsutil: behavior

Controls file system behavior. Queries, changes, enables, or disables the settings for generating 8.3 character-length file names and the amount of disk space reserved of the MFT Zone. Queries how many bytes of RAM NTFS for Linux library (UFSD) uses.

## Syntax

**fsutil behavior query <volume> <option>** - Query the file system behavior parameters.

E.g.: **fsutil behavior query /mnt/vol1/ memoryusage**

**fsutil behavior set <volume> <option> <value>** - Change the file system behavior parameters.

E.g.: **fsutil behavior set /mnt/vol1 disable8dot3 1**

## Options

<b>disable8dot3</b> {1 0}	Disables creation of 8.3 character-length file names on NTFS-formatted volumes.
<b>mftzone value</b>	The master file table (MFT) Zone is a reserved area that enables the MFT to expand as needed, in order to prevent MFT fragmentation. Set the value from 1 (default) to 4 (maximum). The value is in 8ths of the disk.
<b>memoryusage</b>	Shows memory usage of NTFS for Linux library (UFSD) in bytes.  <b>TotalBytes</b> – total amount of bytes.  <b>BytesPerDir</b> – how many bytes the library uses for opened directories.  <b>BytesPerFile</b> - how many bytes the library uses for opened files.

## Remarks

- Using `disable8dot3 {1|0}`

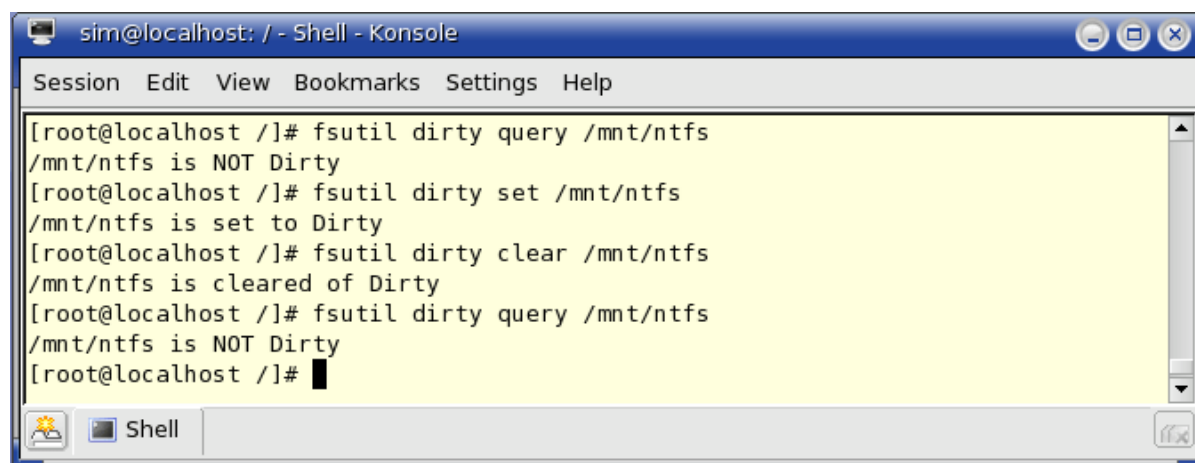
When `disable8dot3` is set to 0, every time you create a file with a long file name, NTFS creates a second file entry that has a 8.3 character-length file name. When NTFS creates files in a folder, it must find the 8.3 character-length file names associated with the long file names.

- Using `mftzone` value

The value is an approximation of the initial size of the MFT plus the MFT Zone for a new volume. It is set when mounting for each file system. As space on the volume is used, NTFS adjusts the space reserved for future MFT growth. If the MFT Zone is already large, the full MFT Zone size is not reserved again. MFT Zone shrinks as the space is used.

The file system does not redetermine the MFT Zone location until the current MFT Zone is completely used.

## Screenshots

A screenshot of a terminal window titled "sim@localhost: / - Shell - Konsole". The window has a menu bar with "Session", "Edit", "View", "Bookmarks", "Settings", and "Help". The terminal output shows a series of commands and their results: [root@localhost /]# fsutil dirty query /mnt/ntfs, /mnt/ntfs is NOT Dirty; [root@localhost /]# fsutil dirty set /mnt/ntfs, /mnt/ntfs is set to Dirty; [root@localhost /]# fsutil dirty clear /mnt/ntfs, /mnt/ntfs is cleared of Dirty; [root@localhost /]# fsutil dirty query /mnt/ntfs, /mnt/ntfs is NOT Dirty. The terminal ends with a prompt [root@localhost /]# and a cursor. The window has standard OS window controls (minimize, maximize, close) in the top right corner and a taskbar at the bottom with icons for a shell and a file manager.

## Fsutil: dirty

Queries to see whether a volume's dirty bit is set. Sets a volume's dirty bit. When a volume's dirty bit is set, autochk (for Windows OS only) automatically checks the volume for errors the next time the computer is restarted.

## Syntax

`fsutil dirty query <volume or device>` - Query the dirty bit.

E.g.: `fsutil dirty query /mnt/vol1`

`fsutil dirty set <volume or device>` - Set the dirty bit.

E.g.: `fsutil dirty set /mnt/vol1`

`fsutil dirty clear <volume or device> <option> <value>` - Clear the dirty bit.

E.g.: `fsutil dirty clear /mnt/vol1`

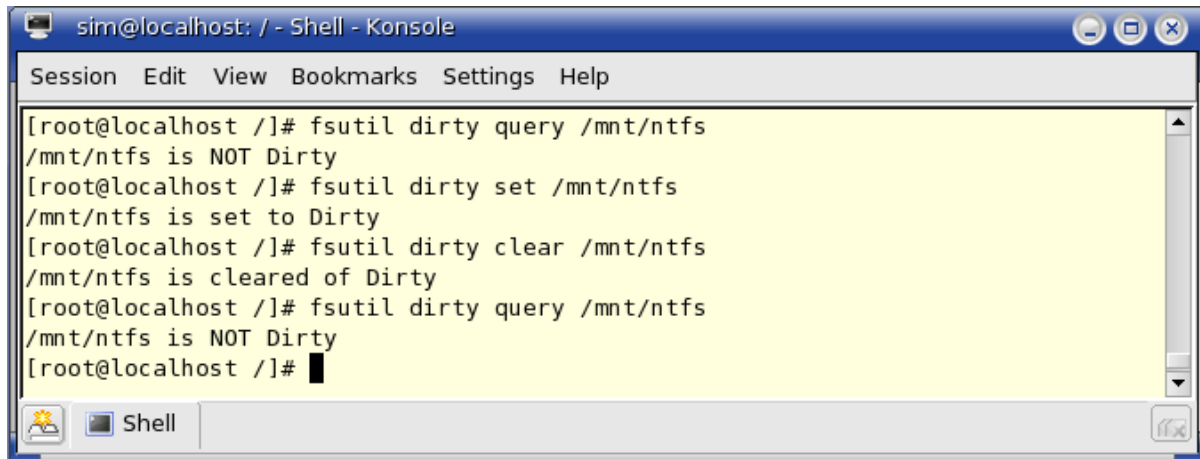
`<volume or device>`

You can specify the volume (mount point) in case the partition is mounted or you can specify the device name (`/dev/hda1`) in case the partition is not mounted.

## Remarks

- If a volume's dirty bit is set, this indicates that the file system may be in an inconsistent state. The dirty bit can be set because the volume is online and has outstanding changes, because changes were made to the volume and the computer shutdown before the changes were committed to disk, or because corruption was detected on the volume. If the dirty bit is set when the computer restarts, `chkdsk` (Windows utility) runs to verify the consistency of the volume.

## Screenshots



```
sim@localhost: / - Shell - Konsole
Session Edit View Bookmarks Settings Help

[root@localhost /]# fsutil dirty query /mnt/ntfs
/mnt/ntfs is NOT Dirty
[root@localhost /]# fsutil dirty set /mnt/ntfs
/mnt/ntfs is set to Dirty
[root@localhost /]# fsutil dirty clear /mnt/ntfs
/mnt/ntfs is cleared of Dirty
[root@localhost /]# fsutil dirty query /mnt/ntfs
/mnt/ntfs is NOT Dirty
[root@localhost /]#
```

## Fsutil: file

Typically used by support professionals. Queries allocated ranges for a file, sets a file's short name, sets a file's valid data length, sets zero data for a file and etc.

## Syntax

`fsutil file <queryallocranges> <filename>` - Query the allocated ranges for a file.

E.g.: `fsutil file queryallocranges /mnt/vol1/hello.txt`

`fsutil file <setshortname> <filename> <shortname>` - Set the short name for a file.

E.g.: `fsutil file setshortname /mnt/vol1/hello.txt short.txt`

`fsutil file <getsizes> <filename>` - Get the sizes for a file.

E.g.: `fsutil file getsizes /mnt/vol1/hello.txt`

`fsutil file <setvaliddata> <filename> <datalength>` - Set the valid data length for a

file.

E.g.: `fsutil file setvaliddata /mnt/vol1/hello.txt 4096`

`fsutil file <setzerodata> offset=<offset> length=<length> <filename>` - Set the zero data for a file.

E.g.: `fsutil file setzerodata offset=100 length=150 /mnt/vol1/hello.txt`

`fsutil file <dumprecord> <filename>` - Dumps raw file/directories record.

E.g.: `fsutil file dumprecord /mnt/vol1/hello.txt`

`fsutil file <dumprecordnum> <volume> <record_num>` - Dump raw record by its number.

E.g.: `fsutil file dumprecordnum /mnt/vol1/1234`

E.g.: `fsutil file dumprecordnum /mnt/vol1/ 0x1234`

## Options

<b>queryallocranges</b>	Queries the allocated ranges for a file on an NTFS volume. Useful for determining whether a file has sparse regions.
<b>setshortname</b>	Sets the short name (8.3 character-length file name) for a file on a NTFS volume.  <b>shortname</b> - Specifies the file's shortname.
<b>getsizes</b>	Shows three Windows sizes: Allocated, Data, Valid.
<b>setvaliddata</b>	Sets the valid data length for a file on an NTFS volume.  <b>datalength</b> - Specifies the length of the file in bytes.
<b>setzerodata</b>	Sets a range (specified by offset and length) of the file to zeroes, which empties the file. If the file is a sparse file, the underlying allocation units are decommitted.  <b>offset=offset</b> - Specifies the file offset, the start of the range to set to zeroes.  <b>length=length</b> - Specifies the length of the range to set to zero.
<b>dumprecord</b>	Shows all MFT records for the specified file.
<b>dumprecordnum</b>	Shows the specified MFT record for the specified file.  <b>record_num</b> - Specified the number of MFT record to show.

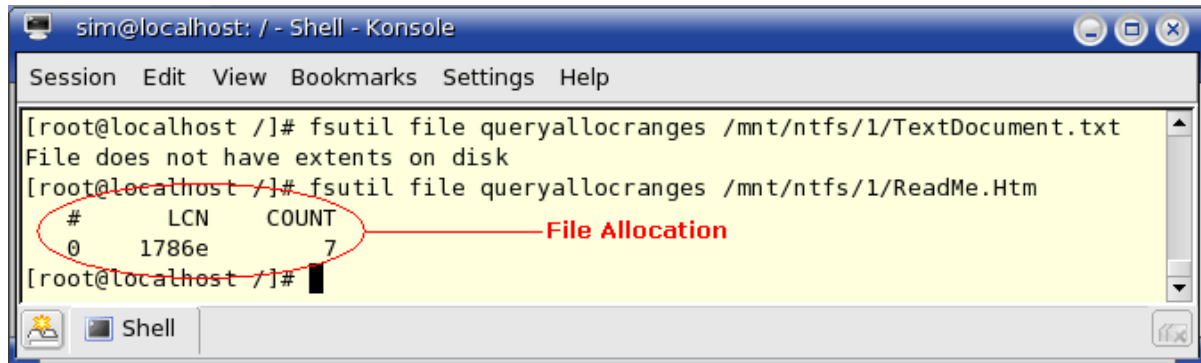
## Remarks

- Using `setvaliddata`

There are two important concepts of file length in NTFS: the End of File (EOF) marker and the Valid Data Length (VDL). The EOF indicates the actual length of the file. The VDL identifies the length of valid data on disk. Any reads between VDL and EOF automatically return 0.

## Screenshots

1. The queryallocranges option.



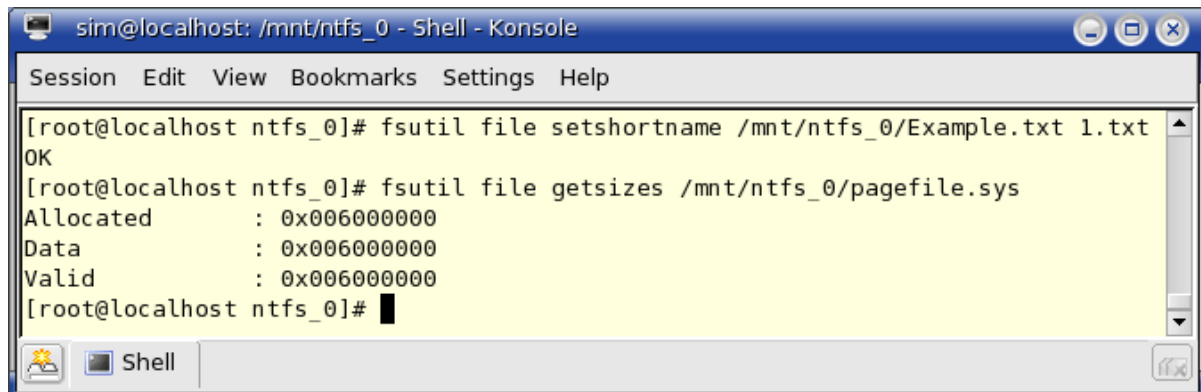
```
sim@localhost: / - Shell - Konsole
Session Edit View Bookmarks Settings Help

[root@localhost /]# fsutil file queryallocranges /mnt/ntfs/1/TextDocument.txt
File does not have extents on disk
[root@localhost /]# fsutil file queryallocranges /mnt/ntfs/1/ReadMe.Htm
#      LCN      COUNT
0      1786e     7
[root@localhost /]#
```

File Allocation

The /mnt/1/TextDocument.txt file lies in the MFT Zone that is why the file doesn't have extents on the disk. LCN – Logical Cluster Number.

2. Setshortname and getsizes options.



```
sim@localhost: /mnt/ntfs_0 - Shell - Konsole
Session Edit View Bookmarks Settings Help

[root@localhost ntfs_0]# fsutil file setshortname /mnt/ntfs_0/Example.txt 1.txt
OK
[root@localhost ntfs_0]# fsutil file getsizes /mnt/ntfs_0/pagefile.sys
Allocated      : 0x0060000000
Data           : 0x0060000000
Valid          : 0x0060000000
[root@localhost ntfs_0]#
```

3. The dumprecord option:



```

sim@localhost: / - Shell - Konsole
Session Edit View Bookmarks Settings Help

[root@localhost /]# fsutil file dumprecord /mnt/ntfs_0/Example.txt
File/Dir "/mnt/ntfs_0/Example.txt" (base 386) consist of 1 record(s).
0000: 46 49 4c 45 30 00 03 00 4f 46 8c 01 00 00 00 00 FILE0...OF.....
0010: 88 01 01 00 38 00 01 00 f0 01 00 00 00 04 00 00 ....8...p.....
0020: 00 00 00 00 00 00 00 00 08 00 00 00 82 01 00 00 .....
0030: 00 00 00 00 00 00 00 00 10 00 00 00 60 00 00 00 .....
0040: 00 00 00 00 00 00 00 00 48 00 00 00 18 00 00 00 .....H.....
0050: 0a 87 8c 3c 8a bd c5 01 a0 8f 6d 15 8c bd c5 01 ...<.=E. .m.=E.
0060: a0 8f 6d 15 8c bd c5 01 a0 8f 6d 15 8c bd c5 01 .m.=E. .m.=E.
0070: 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0080: 00 00 00 00 47 01 00 00 00 00 00 00 00 00 00 00 ....G.....
0090: 00 00 00 00 00 00 00 00 30 00 00 00 70 00 00 00 .....0...p...
00a0: 00 00 00 00 00 00 06 00 58 00 00 00 18 00 01 00 .....X.....
00b0: 05 00 00 00 00 00 05 00 7e 10 d0 cc 8b bd c5 01 .....~.PL.=E.
00c0: c6 72 d5 d6 8b bd c5 01 c6 72 d5 d6 8b bd c5 01 FrUV.=E.FrUV.=E.
00d0: c6 72 d5 d6 8b bd c5 01 40 00 00 00 00 00 00 00 FrUV.=E.@.....
00e0: 3d 00 00 00 00 00 00 00 20 00 00 00 00 00 00 00 =.....
00f0: 0b 03 45 00 78 00 61 00 6d 00 70 00 6c 00 65 00 ..E.x.a.m.p.l.e.
0100: 2e 00 74 00 78 00 74 00 40 00 00 00 28 00 00 00 ..t.x.t.@...(...)
0110: 00 00 00 00 00 00 05 00 10 00 00 00 18 00 00 00 .....
0120: 05 0a 50 83 7d 29 da 11 b8 f2 00 0c 29 42 0f 5c ..P.})Z.8r..)B.\
0130: 80 00 00 00 58 00 00 00 00 00 18 00 00 00 01 00 ...X.....
0140: 3d 00 00 00 18 00 00 00 0d 0a 54 68 69 73 20 69 =.....This i
0150: 73 20 73 69 6d 70 6c 65 20 74 65 78 74 20 66 69 s simple text fi
0160: 6c 65 20 74 68 61 74 20 77 61 73 20 63 72 65 61 le that was crea
0170: 74 65 64 20 75 6e 64 65 72 20 57 69 6e 64 6f 77 ted under Window
    
```

## Fsutil: fsinfo

Typically used by support professionals. Queries the drive type, queries volume information, queries NTFS-specific volume information, or queries file system statistics.

## Syntax

**fsutil fsinfo <volumeinfo> <volume pathname>** – Query volume information.

E.g.: **fsutil fsinfo volumeinfo /mnt/vol1**

**fsutil fsinfo <ntfsinfo> <volume pathname>** – Query NTFS specific volume information.

E.g.: **fsutil fsinfo ntfsinfo /mnt/vol1**

**fsutil fsinfo <statistics> <volume pathname>** - Query file system statistics.

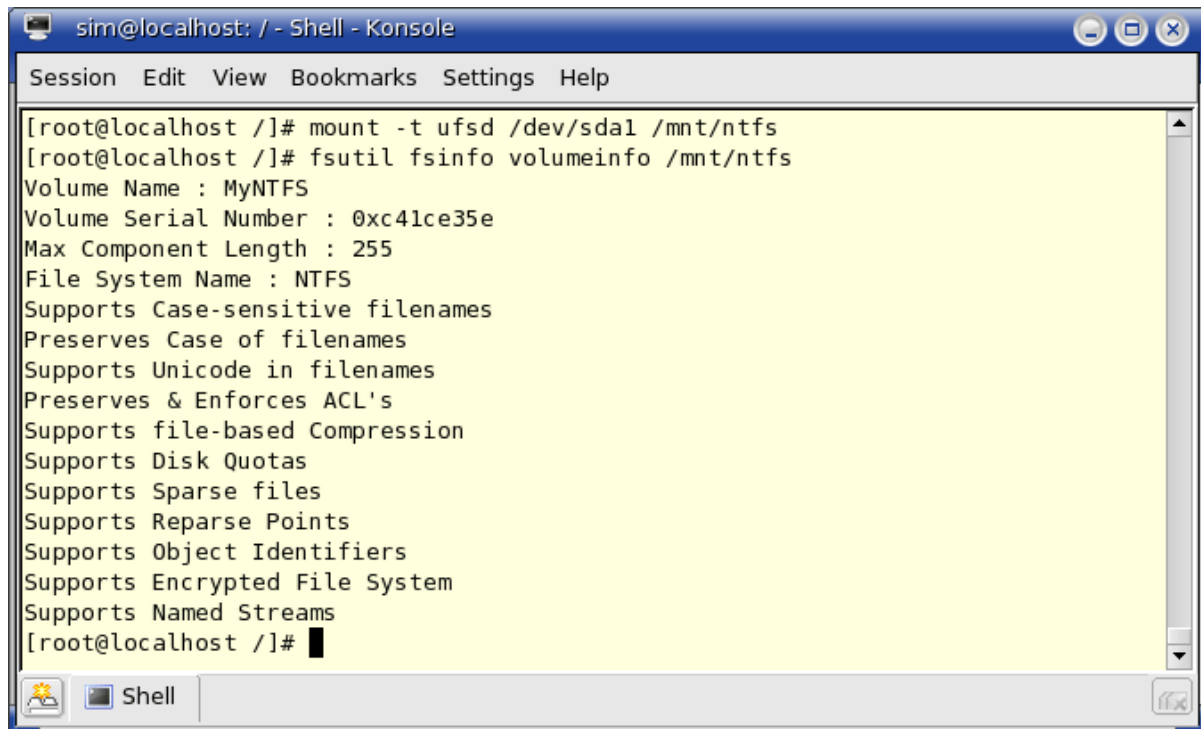
E.g.: **fsutil fsinfo statistics /mnt/vol1**

## Options

<b>volumeinfo</b>	Lists information for the specified volume, such as the file system, and whether the volume supports case-sensitive file names, unicode in file names, or disk quotas.
<b>ntfsinfo</b>	Lists NTFS specific volume information for the specified volume, such as the number of sectors, total clusters, free clusters, and the start and end of the MFT Zone.
<b>statistics</b>	Lists file system statistics for the specified volume, such as metadata, log file, and MFT reads and writes.

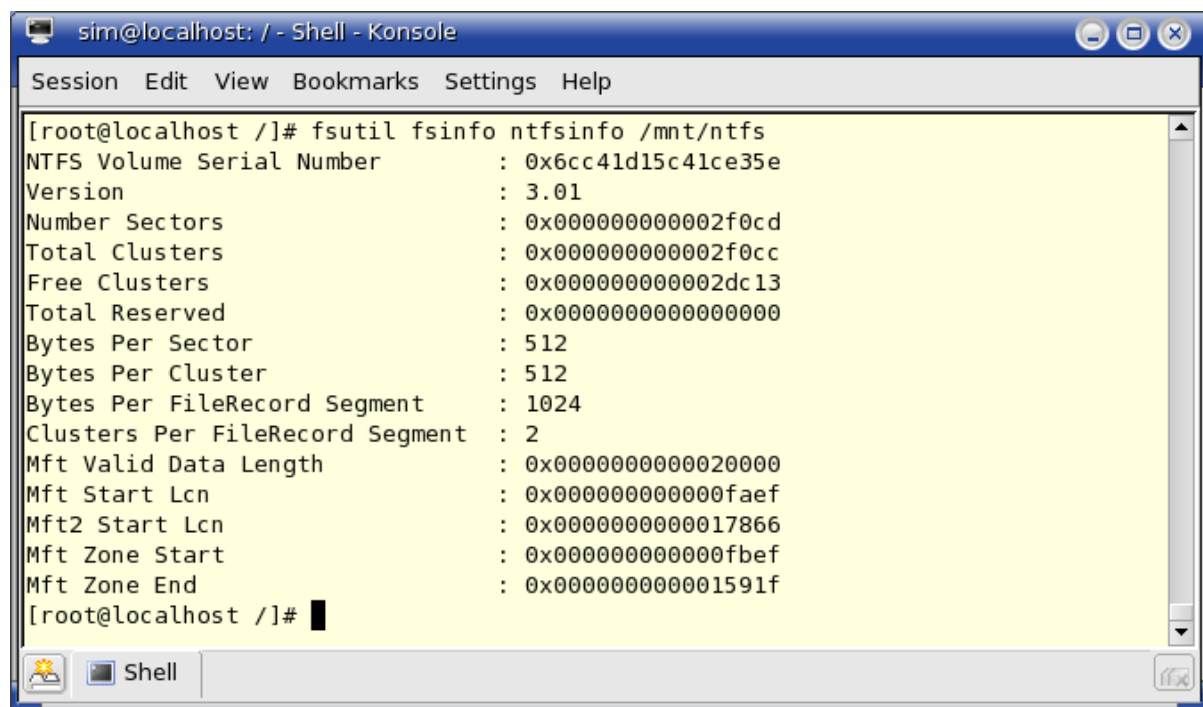
## Screenshots

### 1. The voluminfo option.



```
sim@localhost: / - Shell - Konsole
Session Edit View Bookmarks Settings Help
[root@localhost /]# mount -t ufsd /dev/sda1 /mnt/ntfs
[root@localhost /]# fsutil fsinfo volumeinfo /mnt/ntfs
Volume Name : MyNTFS
Volume Serial Number : 0xc41ce35e
Max Component Length : 255
File System Name : NTFS
Supports Case-sensitive filenames
Preserves Case of filenames
Supports Unicode in filenames
Preserves & Enforces ACL's
Supports file-based Compression
Supports Disk Quotas
Supports Sparse files
Supports Reparse Points
Supports Object Identifiers
Supports Encrypted File System
Supports Named Streams
[root@localhost /]#
```

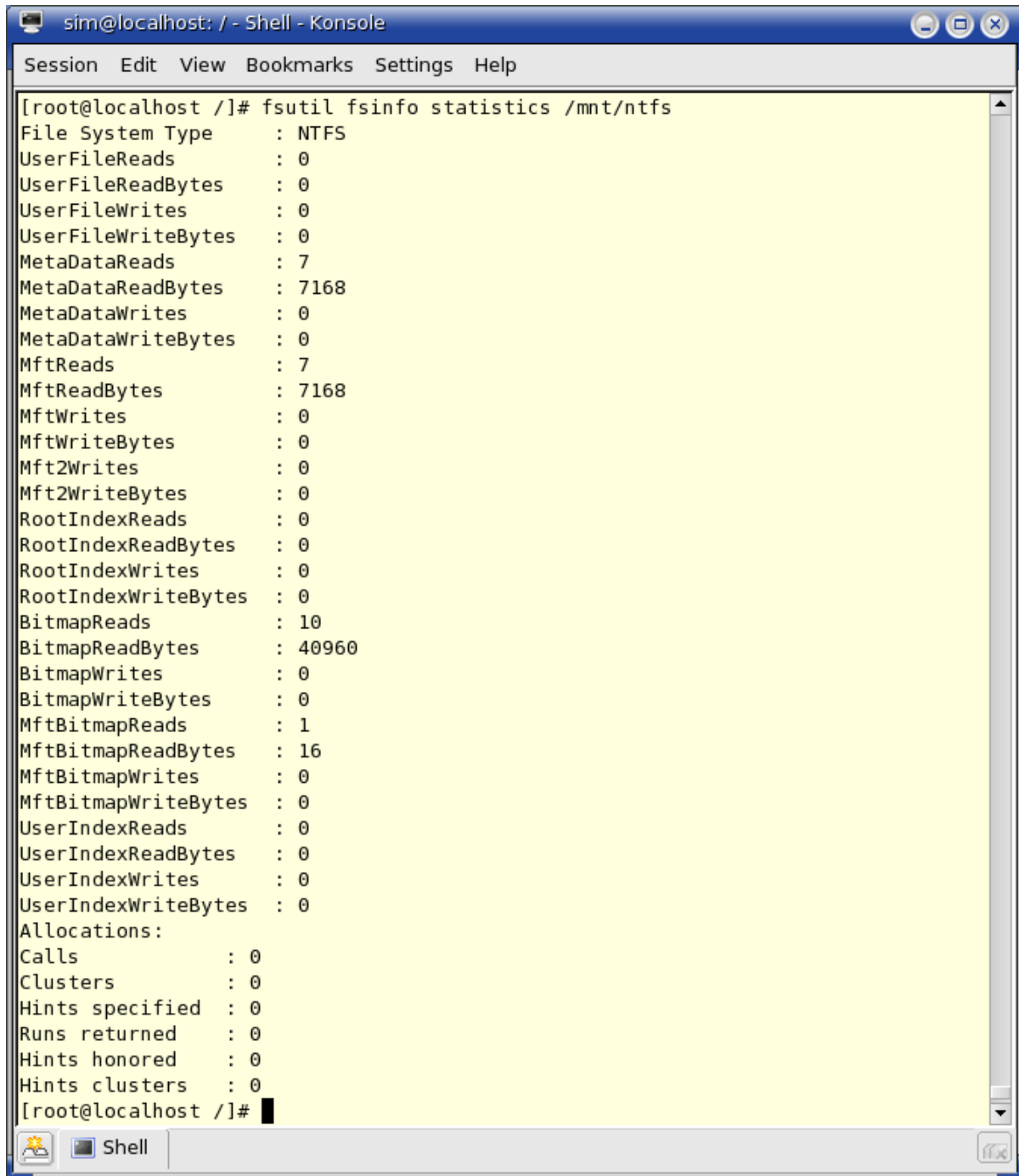
### 2. The ntfsinfo option.



```
sim@localhost: / - Shell - Konsole
Session Edit View Bookmarks Settings Help

[root@localhost /]# fsutil fsinfo ntfsinfo /mnt/ntfs
NTFS Volume Serial Number      : 0x6cc41d15c41ce35e
Version                        : 3.01
Number Sectors                 : 0x000000000002f0cd
Total Clusters                 : 0x000000000002f0cc
Free Clusters                  : 0x000000000002dc13
Total Reserved                 : 0x0000000000000000
Bytes Per Sector               : 512
Bytes Per Cluster              : 512
Bytes Per FileRecord Segment   : 1024
Clusters Per FileRecord Segment : 2
Mft Valid Data Length          : 0x0000000000020000
Mft Start Lcn                  : 0x000000000000faef
Mft2 Start Lcn                 : 0x00000000000017866
Mft Zone Start                 : 0x000000000000fbef
Mft Zone End                   : 0x0000000000001591f
[root@localhost /]#
```

3. The statistics option.



```
sim@localhost: / - Shell - Konsole
Session Edit View Bookmarks Settings Help

[root@localhost /]# fsutil fsinfo statistics /mnt/ntfs
File System Type      : NTFS
UserFileReads         : 0
UserFileReadBytes     : 0
UserFileWrites        : 0
UserFileWriteBytes    : 0
MetaDataReads         : 7
MetaDataReadBytes     : 7168
MetaDataWrites        : 0
MetaDataWriteBytes    : 0
MftReads              : 7
MftReadBytes          : 7168
MftWrites             : 0
MftWriteBytes         : 0
Mft2Writes            : 0
Mft2WriteBytes        : 0
RootIndexReads        : 0
RootIndexReadBytes    : 0
RootIndexWrites       : 0
RootIndexWriteBytes   : 0
BitmapReads           : 10
BitmapReadBytes       : 40960
BitmapWrites          : 0
BitmapWriteBytes      : 0
MftBitmapReads        : 1
MftBitmapReadBytes    : 16
MftBitmapWrites       : 0
MftBitmapWriteBytes   : 0
UserIndexReads        : 0
UserIndexReadBytes    : 0
UserIndexWrites       : 0
UserIndexWriteBytes   : 0
Allocations:
Calls                 : 0
Clusters              : 0
Hints specified       : 0
Runs returned         : 0
Hints honored         : 0
Hints clusters        : 0
[root@localhost /]#
```

## Fsutil: hardlink

A hard link is a directory entry for a file. Every file can be considered to have at least one hard link. On NTFS volumes, each file can have multiple hard links, and thus a single file can appear in many directories (or even in the same directory with different names). Because all of the links reference the same file, programs can open any of the links and modify the file. A file is deleted from the file system only after all links to it have been deleted. After you create a hard link, programs can use it like any other file name.

## Syntax

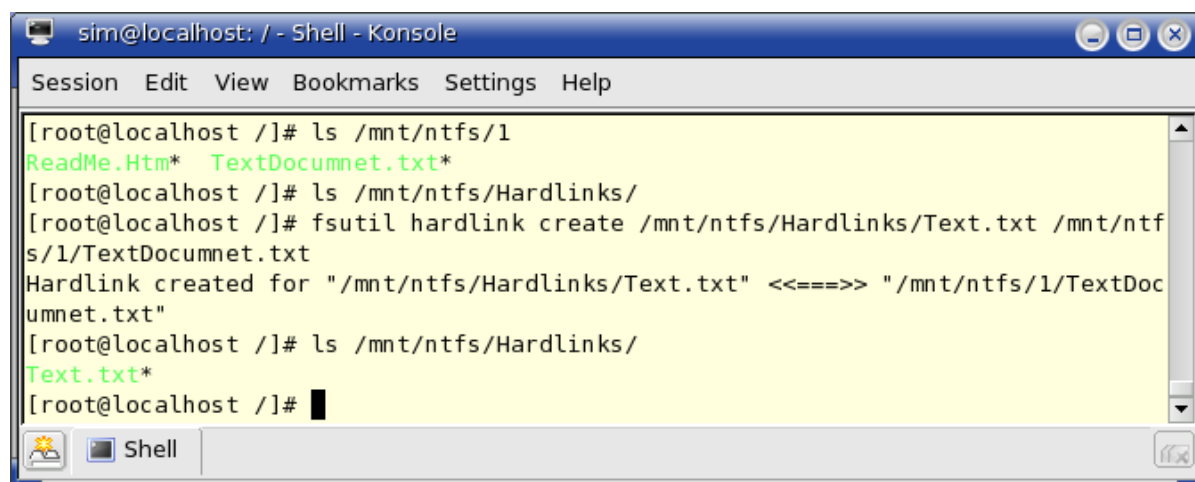
**fsutil hardlink <create> <new\_filename> <existing\_filename>** – Create a hardlink.

E.g.: **fsutil hardlink create /mnt/vol1/hi.txt /hello.txt**

## Options

<b>create</b>	Establishes an NTFS hard link between an existing file and a new file. An NTFS hard link is similar to a POSIX hard link.
<b>new_filename</b>	Specifies the file to which you want to create a hardlink.
<b>existing_filename</b>	Specifies the file from which you want to create a hardlink.

## Screenshots



```
sim@localhost: / - Shell - Konsole
Session Edit View Bookmarks Settings Help

[root@localhost /]# ls /mnt/ntfs/1
ReadMe.Htm* TextDocumnet.txt*
[root@localhost /]# ls /mnt/ntfs/Hardlinks/
[root@localhost /]# fsutil hardlink create /mnt/ntfs/Hardlinks/Text.txt /mnt/ntfs/1/TextDocumnet.txt
Hardlink created for "/mnt/ntfs/Hardlinks/Text.txt" <====> "/mnt/ntfs/1/TextDocumnet.txt"
[root@localhost /]# ls /mnt/ntfs/Hardlinks/
Text.txt*
[root@localhost /]#
```

## Fsutil: objectid

Typically used by professionals. Manages object identifiers, which are internal objects used by the Distributed Link Tracking (DLT) Client service and File Replication Service (FRS) to track other objects such as files, directories, and links. Object identifiers are invisible to most programs and should never be modified.

## Syntax

**fsutil objectid <query>** - Query the object identifier.

E.g.: **fsutil objectid query /mnt/vol1/hello.txt**

**fsutil objectid <set> <ObjectId> <BirthVolumeId> <BirthObjectId> <DomainId> <filename>** - Change the object identifier.

E.g.: **fsutil objectid set 7adcc02fc9b4d4118f120090273fa9fc  
dc6ad6 865fe8d21183913008c409d19e**

```
d2dff02fc9b4d4118f120090273fa9d2
```

```
00000000000000000000000000000000 /mnt/vol1/hello.txt
```

**fsutil objectid <delete> <filename>** - Delete the object identifier.

E.g.: **fsutil objected delete /mnt/vol1/hello.txt**

**fsutil objectid <create>** - Create the object identifier.

E.g.: **fsutil objected create /mnt/vol1/hello.txt**

## Options

<b>query</b>	Queries the object identifier.
<b>set</b>	Changes the object identifier.
<b>delete</b>	Deletes the object identifier.
<b>create</b>	Creates the object identifier if the file does not already have one, otherwise equivalent to query.

### ObjectID

A file-specific 16 byte hexadecimal identifier that is guaranteed to be unique within a volume. It is used by the Distributed Link Tracking (DLT) Client service and the File Replication Service (FRS) to identify files. Any file that has an **ObjectID**, also has a **BirthVolumeID**, a **BirthObjectID**, and a **DomainID**. When you move a file, the **ObjectID** may change, but **BirthVolumeID** and **BirthObjectID** remain the same, which enables Windows to always find a file, no matter where it has been moved.

### BirthVolumeID

A 16 byte hexadecimal identifier indicates the volume on which the file was located when it first obtained an **ObjectID**. This value is used by DLT Client service.

### BirthObjectID

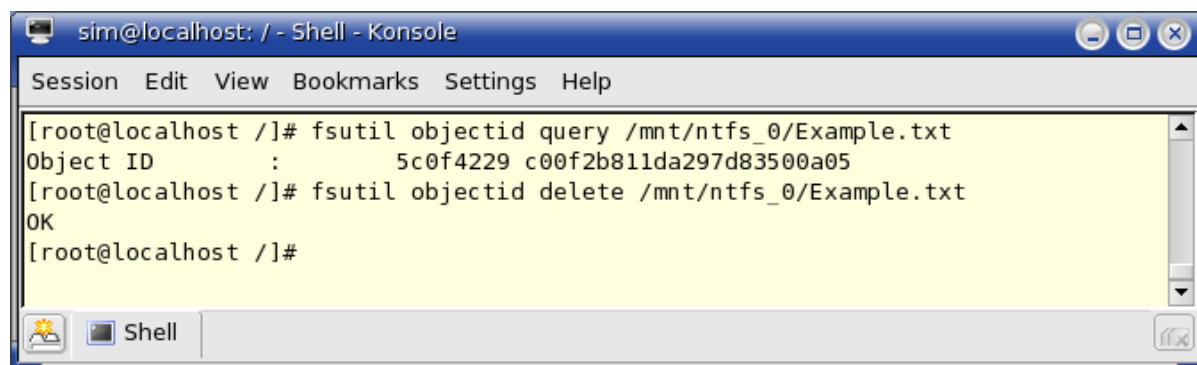
A 16 byte hexadecimal identifier indicates the file's original **ObjectID**. This value is used by DLT Client service.

### DomainID

16 byte hexadecimal domain identifier. This value is not currently used and must be set to all zeros.

Note: All values must be in Hex of the form 8a0cf02fc9b4d4118f120090273fa91a.

## Screenshots



```
sim@localhost: / - Shell - Konsole
Session Edit View Bookmarks Settings Help

[root@localhost /]# fsutil objectid query /mnt/ntfs_0/Example.txt
Object ID      :      5c0f4229 c00f2b811da297d83500a05
[root@localhost /]# fsutil objectid delete /mnt/ntfs_0/Example.txt
OK
[root@localhost /]#
```

## Fsutil: compress

Compressing files decreases their size and reduces the amount of space they use on your drives or removable storage media.

### Syntax

**fsutil compress queryflag <filename>** – Query compression flag.

E.g.: **fsutil compress queryflag /mnt/vol1/hello.txt**

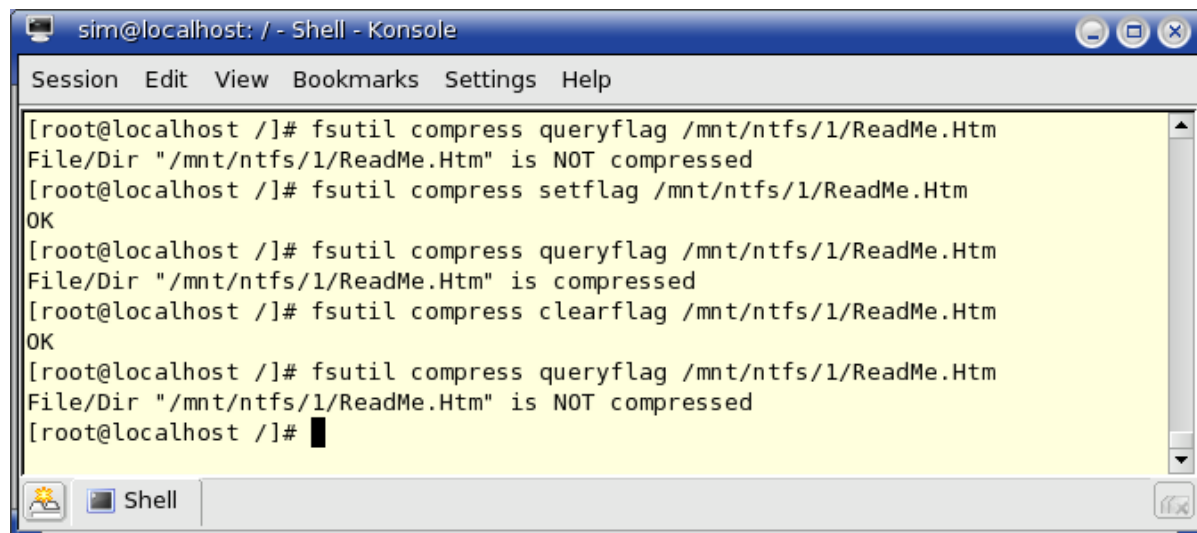
**fsutil compress setflag <filename> (<-r>)** – Set compression flag.

E.g.: **fsutil compress setflag /mnt/vol1/hello.txt -r**

**fsutil compress clearflag <filename> (<-r>)** – Clear compression flag.

E.g.: **fsutil compress clearflag /mnt/vol1/hello.txt -r**

### Screenshots



```
sim@localhost: / - Shell - Konsole
Session Edit View Bookmarks Settings Help

[root@localhost /]# fsutil compress queryflag /mnt/ntfs/1/ReadMe.Htm
File/Dir "/mnt/ntfs/1/ReadMe.Htm" is NOT compressed
[root@localhost /]# fsutil compress setflag /mnt/ntfs/1/ReadMe.Htm
OK
[root@localhost /]# fsutil compress queryflag /mnt/ntfs/1/ReadMe.Htm
File/Dir "/mnt/ntfs/1/ReadMe.Htm" is compressed
[root@localhost /]# fsutil compress clearflag /mnt/ntfs/1/ReadMe.Htm
OK
[root@localhost /]# fsutil compress queryflag /mnt/ntfs/1/ReadMe.Htm
File/Dir "/mnt/ntfs/1/ReadMe.Htm" is NOT compressed
[root@localhost /]#
```

## Fsutil: streams

This subcommand is intended for querying and dumping streams of a file. It shows Type, Id, Size and Name of all streams of the specified file. It also can show the raw data of the specified stream of a file as a dump.

## Syntax

**fsutil streams query <filename>** – Query the list of streams.

E.g.: **fsutil streams query /mnt/vol1/hello.txt**

**fsutil streams dump <filename> <type(:name)> (<Id>)** - Dump the contents of stream.

E.g.: **fsutil file streams dump /mnt/vol1/ hello.txt 10**

E.g.: **fsutil file streams dump /mnt/vol1/ hello.txt 30 1**

E.g.: **fsutil file streams dump /mnt/vol1/ hello.txt 90:\$I30 2**

## Screenshots



Type of stream:

10 - Standard Information

30 - Name (can be missing)

40 - Obj ID

80 - Data

Not Named Data is Default Data

Named Data - Alternate Data Stream

```

sim@localhost: / - Shell - Konsole
Session Edit View Bookmarks Settings Help

[root@localhost /]# fsutil streams query /mnt/ntfs/2/TextDocument2.txt
Streams of "/mnt/ntfs/2/TextDocument2.txt":
Type      Id      Size      (Name)
0x0010 0x0000 0x000000048
0x0030 0x0003 0x00000005a
0x0030 0x0000 0x000000064
0x0080 0x0001 0x000000035
0x0080 0x0002 0x000000090 (\5DocumentSummaryInformation)
0x0080 0x0003 0x00000009c (\5SebiesnrMkudrfcoIaamtykdDa)
0x0080 0x0004 0x000000160 (\5SummaryInformation)
0x0080 0x0005 0x000000024 (KAVICHS)
0x0080 0x0009 0x000000000 ({4c8cc155-6c1e-11d1-8e41-00c04fb9386d})

[root@localhost /]# fsutil streams query /mnt/ntfs/2/ReadMe2.htm
Streams of "/mnt/ntfs/2/ReadMe2.htm":
Type      Id      Size      (Name)
0x0010 0x0000 0x000000048
0x0030 0x0002 0x000000058
0x0080 0x0003 0x000000d4d
0x0080 0x0004 0x000000024 (KAVICHS)

[root@localhost /]# fsutil streams dump /mnt/ntfs/2/TextDocument2.txt 80 1
Dump stream 80: of file "/mnt/ntfs/2/TextDocument2.txt"
0000: 54 68 69 73 20 69 73 20 73 61 6d 70 6c 65 20 74 This is sample t
0010: 65 78 74 2e 20 28 53 68 6f 75 6c 64 20 62 65 20 ext. (Should be
0020: 72 65 73 69 64 65 6e 74 20 61 74 74 72 69 62 75 resident attribu
0030: 74 65 29 0d 0a te)..

[root@localhost /]# fsutil streams dump /mnt/ntfs/2/ReadMe2.htm 30 2
Dump stream 30: of file "/mnt/ntfs/2/ReadMe2.htm"
0000: 22 00 00 00 00 00 01 00 d2 02 66 d3 56 83 c5 01 ".....R.fSV.E.
0010: d2 02 66 d3 56 83 c5 01 d2 02 66 d3 56 83 c5 01 R.fSV.E.R.fSV.E.
0020: d2 02 66 d3 56 83 c5 01 00 00 00 00 00 00 00 00 R.fSV.E.....
0030: 00 00 00 00 00 00 00 20 00 00 00 00 00 00 00 00 .....
0040: 0b 03 52 00 65 00 61 00 64 00 4d 00 65 00 32 00 ..R.e.a.d.M.e.2.
0050: 2e 00 68 00 74 00 6d 00 ..h.t.m.

[root@localhost /]#
    
```

Note: Every stream has a unique pair of Type and Id.

## Fsutil: sparse

This subcommand manages sparse file. A sparse file is a file that is handled in a way that requires much less disk space than would otherwise be needed. Sparse support allows an application to create very large files without committing disk space for regions of the file that only contain zeros. For example, you can use sparse support to work with a 10GB file in which you need to write data only to the first 64 KB (the rest of the file is zeroed). In other words, all meaningful or nonzero data is allocated, whereas all not meaningful data (large strings of data composed of

zeros) is not allocated. When a sparse file is read, allocated data is returned as stored and unallocated data is returned, by default, as zeros.

## Syntax

**fsutil sparse <setflag> <filename> (<-r>) – Set sparse.**

E.g.: **fsutil sparse setflag /mnt/vol1/hello.txt -r**

**fsutil sparse <queryflag> <filename> – Query sparse.**

E.g.: **fsutil sparse queryflag /mnt/vol1/hello.txt**

**fsutil sparse <queryrange> <filename> – Query range.**

E.g.: **fsutil sparse queryrange /mnt/vol1/hello.txt**

**fsutil sparse <setrange> <filename> <beginning offset> <length> – Set sparse range.**

E.g.: **fsutil sparse setrange /mnt/vol1/hello.txt 65536 131072**

## Options

<b>setflag</b>	Marks the indicated file as sparse.
<b>queryflag</b>	Queries sparse.
<b>queryrange</b>	Scans a file looking for ranges that may contain nonzero data.
<b>setrange</b>	<p>Fills a specified range of a file with zeroes.</p> <p><b>beginning offset</b> - Offset within the file to mark as sparse.</p> <p><b>length</b> - Length of the region in the file to be marked as sparse, in bytes.</p>

## Remarks

- In a sparse file, large ranges of zeroes may not require disk allocation. Space for nonzero data will be allocated as needed as the file is written.
- Only compressed or sparse files can have zeroed ranges known to the operating system.
- If the file is sparse or compressed, NTFS may deallocate disk space within the file. This sets the range of bytes to zeroes without extending the file size.

## 6.1.9 junction

JUNCTION Utility – Reparse point viewer on a NTFS volume.

## Name

**junction** – display reparse point information.

## Synopsis

**junction device [options]**

E.g.: **junction /dev/hdb1 -o report.txt**

## Options

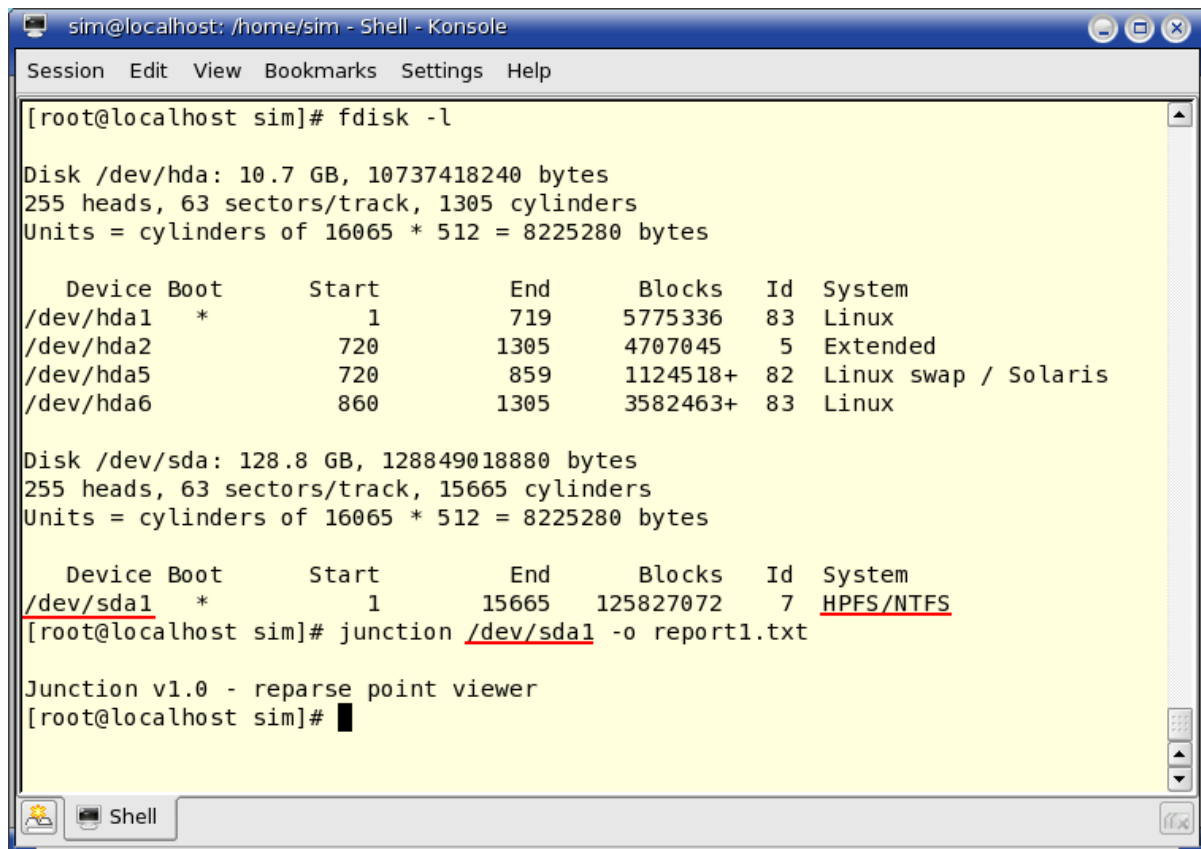
<b>-o</b>	A file name should be specified (where all reparse points must be enumerated). Stdout is by default.
<b>-v</b>	Explain what is being done.
<b>--help</b>	Display this help.
<b>--trace</b>	Turn on UFSD trace
<b>--version</b>	Show the version and exit.

## Description

Windows 2000 and higher supports junctions - directory symbolic links, where a directory used as a symbolic link to another directory on the computer. For example, if the directory "D:\symlink" specifies "C:\winnt\system32" as its target, then when an application accesses "D:\symlink\drivers", it actually accesses "C:\winnt\system32\drivers". Linux doesn't have any tools to manage junctions and we therefore decided to write this junction utility. It allows you to see if files or directories are actually reparse points. Reparse points are the mechanism on which NTFS junctions are based, and they are used by Windows' Remote Storage Service (RSS), as well as volume mount points.

## Screenshots

Let's enumerate all reparse points on a Vista NTFS partition. The list of reparse points must be written to a report1.txt file (the file doesn't exist).



```
sim@localhost: /home/sim - Shell - Konsole
Session Edit View Bookmarks Settings Help

[root@localhost sim]# fdisk -l

Disk /dev/hda: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1 *          1           719     5775336   83   Linux
/dev/hda2            720          1305     4707045    5   Extended
/dev/hda5            720           859     1124518+   82   Linux swap / Solaris
/dev/hda6            860          1305     3582463+   83   Linux

Disk /dev/sda: 128.8 GB, 128849018880 bytes
255 heads, 63 sectors/track, 15665 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1 *          1        15665    125827072    7   HPFS/NTFS

[root@localhost sim]# junction /dev/sda1 -o report1.txt

Junction v1.0 - repare point viewer
[root@localhost sim]#
```

report1.txt file:

## 6.2 HFS+ utilities

There are 2 additional utilities for HFS+:

- [mkhfs](#) <sup>[56]</sup> — format any partition as HFS+ under Linux;
- [chkhfs](#) <sup>[57]</sup> — check HFS+ partition for integrity and (optionally) fix errors.

### 6.2.1 mkhfs

MKHFS Utility - Create an HFS volume on a partition.

#### Name

**mkhfs** — create an HFS+ volume on specified (block) device under Linux OS.

#### Synopsis

**mkhfs** [options] device  
 E.g.: **mkhfs** /dev/hdb1

#### Options

<b>-q</b>	Perform a quick format.
<b>-v:label</b>	Specify the volume label.
<b>-a:size</b>	Override the default allocation unit size. Default settings are strongly recommended for general use.  512, 1024, 2048, 4096, 8192, 16K, 32K and 64K are supported.
<b>-f</b>	Force the format without confirmation.
<b>-j</b>	Make volume journaled.
<b>-c</b>	Make volume case-sensitive.
<b>--help</b>	Display this help.
<b>--trace</b>	Turn on UFSD trace.
<b>--verbose</b>	Explain what is being done.
<b>--version</b>	Show the version and exit.

## Description

**mkhfs** is a standalone utility that allows to format HFS+ partitions under Linux. It is used to create an HFS+ file system on a device (usually a disk partition).

### 6.2.2 chkhfs

CHKHFS Utility - Perform consistency checks on an HFS+ volume.

## Name

**chkhfs** — provide consistency checking of a HFS volume and fix errors.

## Synopsis

```
chkhfs device [options]
```

E.g.: **chkhfs /dev/hdb1**

## Options

<b>-f</b>	Fix errors on the disk.
<b>-a</b>	Perform checks only if 'dirty' flag is set.
<b>-h</b>	Display this help.

<b>--trace</b>	Turn on UFSD trace.
<b>--verbose</b>	Explain what is being done.
<b>--version</b>	Show the version and exit.

## Description

**chkhfs** creates and displays a status report about a HFS+ file system. **Chkhfs** also lists and corrects errors on the disk, if any (**-f** flag must be specified).

**Part**

---

**VII**

**Troubleshooting**



This section highlights troubleshooting processes.

## 7.1 Troubleshooting processes

### Step 1. Consult Documentation

Please consult documentation to make sure that encountered behavior is not by design, with special attention given to the part related to installation, testing and troubleshooting as well as to section on [System requirements](#)<sup>[6]</sup>. Please also review [Mount troubleshooting](#)<sup>[61]</sup> and [Using The Driver](#)<sup>[14]</sup> subsection.

### Step 2. Make sure the issue is not related to Linux itself

Now, make sure that root cause of the issue is not related to Linux itself. For example, if an issue is discovered while performing certain file system-related operation on a volume mounted with Paragon NTFS&HFS driver, make sure the same issue is not observed when the same operation is performed on 'native' file system like Ext2fs, Ext3fs or FAT (except, of course, for operations specific to NTFS or HFS+ file systems or to Paragon's driver itself, e.g. IOCTLs, additional utilities and so on).

### Step 3. Prepare to report the issue

After performing previous steps and making sure that the issue is related to Paragon NTFS driver, prepare to report the issue to Paragon.

#### Collect all information on the issue

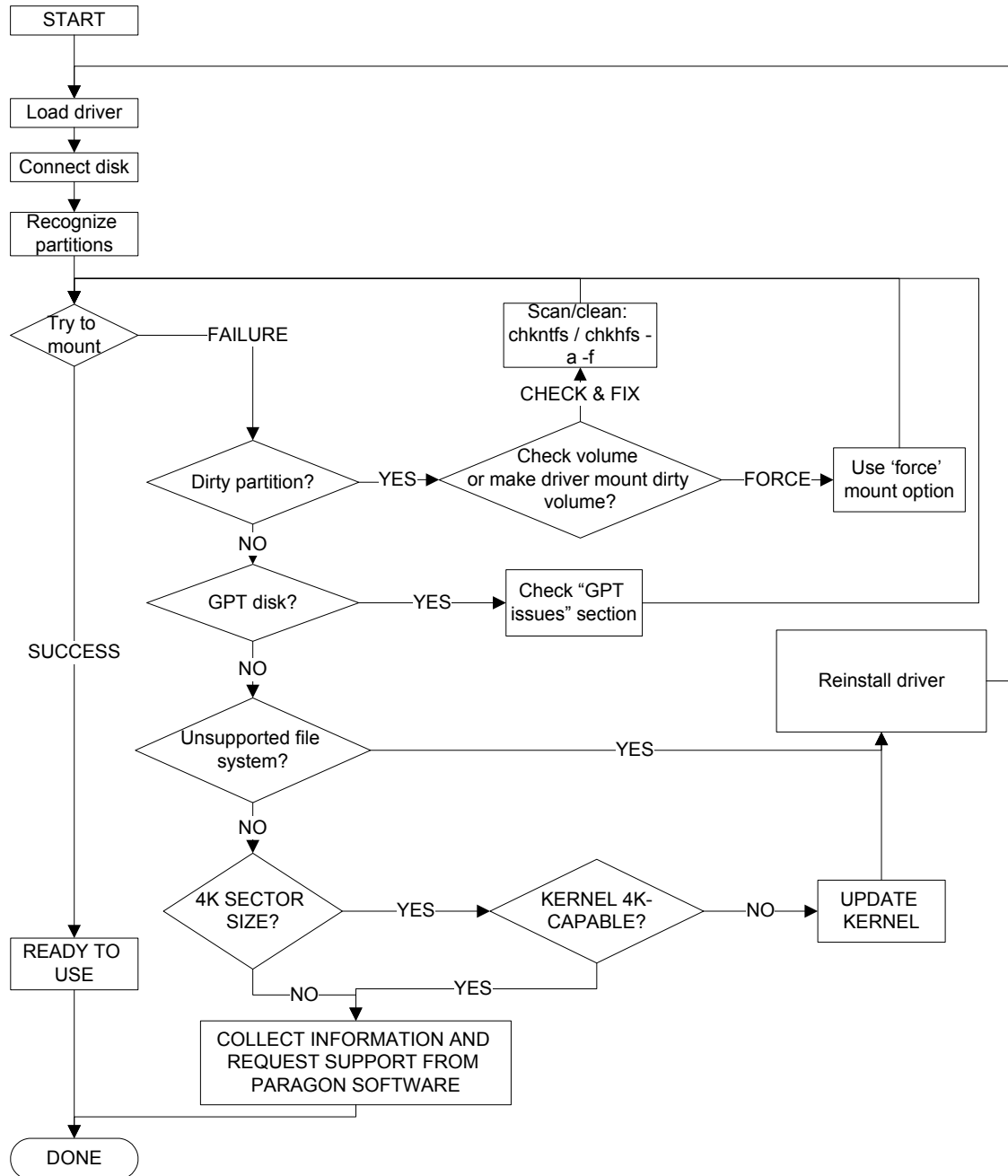
The most important point in issue resolution process is quickly obtaining all the information related to the issue. Quick collection of required information is the key to resolving an issue faster.

### Step 4. Assist Paragon engineers to resolve the issue quickly

Please provide any firmware updates required to reproduce the issue together with information on how to upgrade firmware in your hardware samples. Try to provide as detailed information on the issue, as possible.

## 7.2 Mount troubleshooting

Use our mount troubleshooting diagram for faster mount issue resolution.



## 7.3 The `install.sh` script can't find kernel sources

1. Read system requirements section, make sure all tools are functional. For more information, please read kernel documentation.
2. Linux kernel must be configured correctly.
3. Make sure that you have kernel sources, for example, in the `/usr/src/linux-x.x.xx` directory, where `x.x.xx` is your kernel version (for example, 2.6.10). Type `uname -r` in the command line to know your current kernel version.
4. Create a symbolic link from the `/usr/src/linux-x.x.xx` directory to `/usr/src/linux`. To create the link type `ln -s /usr/src/linux-$(uname -r) /usr/src/linux`.
5. Make sure that you have the `config-x.x.xx` file, for the booted Linux kernel, in the `/boot` directory. If you haven't the `config-x.x.xx` file then type `ln -s /usr/src/linux-$(uname -r)/.config /boot/config-$(uname -r)` to create a symbolic link to the config file.

Note: There are cases when the kernel sources may be located in other directories. In these cases you should create a symbolic link to `/usr/src/linux`, for example, `ln -s /lib/modules/$(uname -r)/build /usr/src/linux`.

If you still have the same problem i.e. the `install.sh` script can't find the kernel sources it is better to rebuild your kernel or download and build a stable kernel from the [www.kernel.org](http://www.kernel.org) site.

## 7.4 Can't compile the NTFS/HFS+ for Linux driver

1. Read System requirements section, make sure all tools are functional. For more information, please read kernel documents.
2. Linux kernel must be configured correctly.
3. The `/boot` directory must contain the `config-(kernel version)` file. If the file is missing you should execute the following command: `ln -s /usr/src/linux-$(uname -r)/.config /boot/config-$(uname -r)`.

## 7.5 "Can't load module" message at the end of installation

1. Make sure that you use the same version of GCC compiler that was used for kernel compilation.
2. Make sure that the **Makefile** of the kernel (you can find the **Makefile** in the directory where the kernel sources are located) have the correct kernel version at the beginning of the file. For example: if your loaded kernel version is `2.6.11-6mdksmp` then the following lines must be found at the beginning of the **Makefile**:

```
VERSION = 2
PATCHLEVEL = 6
SUBLEVEL = 11
```

**EXTRAVERSION** = -6mdksmp

## 7.6 ufsd Module: kernel-module version mismatch

That means kernel version mismatch.

1. Check kernel source version in `/usr/src/linux/include/linux/version.h`
2. Check the currently running kernel version: `uname -r`
3. Both version must match.
4. If they don't match, please restore Kernel configuration or recompile kernel (advanced).

## 7.7 ufsd Module: create\_module: operation is not permitted

That means you must have root privilege to load driver.

## 7.8 insmod: a module named as ufsd already exists

That means driver have been loaded. There is no need to load it again.

Driver status can be found by using the following command: `lsmod | grep ufsd`

## 7.9 When I run the “insmod ufsd.o” command, there are some error messages

1. Make sure you are trying to install a module for this kernel.
2. Generally the same `ufsd` binary module works with both smp and non-smp kernels, but there are exceptions to the rule, please note this.
3. Please note that `ufsd.o` is for 2.4.x kernels, while `ufsd.ko` is for 2.6.x kernel.

## 7.10 I can't mount NTFS/HFS+ volume

1. Make sure that the driver is activated (loaded into the Kernel): `lsmod | grep ufsd`
2. Make sure that the driver supports file system mounted partition is formatted with:

```
cat /proc/fs/ufsd/version
```

3. The volume is dirty. Use `chkntfs/chkhfs` utility with `-a -f` command line options to reset 'dirty' flag. Alternatively, use 'force' mount options to make the driver ignore 'dirty' flag.

**Part**

---



**NTFS & HFS  
Compatibility**

USER MANUAL

This section describes NTFS and HFS features supported by Paragon NTFS&HFS driver, respectively.

## 8.1 NTFS features

### Compressed files

Reading and writing compressed files is fully supported in both sequential and random orders.

### Encrypted files

Encrypted files are read encrypted. During copy operation, file data streams will be copied encrypted with loss of decryption capability. To make full archive preserving all NTFS-specific information `cpntfs` utility can be used (available in the Professional Edition).

### Alternate data streams

When copying from NTFS to Linux FS: all additional streams will not be copied, along with compression flag and security attributes (use the `cpntfs` utility to preserve this information).

### Hardlinks and symlinks

Any link will be copied as a full file with its body, losing link information.

### Maximum filename length

NTFS stores filenames in UTF-16 encoding. This may cause trouble when very long filenames containing non-latin characters are used and UTF-8 is selected as default Kernel codepage.

## 8.2 HFS features

This section describes features of HFS+ file system supported by the driver.

### Case sensitivity

Both case sensitive and case insensitive types of HFS+ file system are supported.

### Alternate data streams (forks)

During file copy operation (using `cp` command) on Linux only 'data' fork is copied.

**Part**

---

**IX**

**Frequently Asked  
Questions**

USER MANUAL

## 9.1 What are 'minor errors' reported by chknfts utility?

Most of information about files (times, sizes, attributes) in NTFS is duplicated and triplicated. Minor error means that copies does not match original. E.g. "latime" means last access time. The native chkdsk from Microsoft does not show these mismatches and fixes it silently (if /f is specified) — see <http://technet.microsoft.com/en-us/library/cc959914.aspx>. Paragon chknfts utility can also find the following minor errors:

mdtime — modification time

ctime — last change time

asize — data allocated size

dsize — data size

attrib — attributes

To see more verbose output on minor errors, use --showminors command line option when running chknfts. For an example output, please see the log below:

```
# chknfts --showminors /dev/sda2
WARNING! f parameter not specified.
Running chknfts in read-only mode.

Checking Volume /dev/sda2...
Verifying 1680 records ...
$UpCase file is formatted for use in Windows NT/2K/XP Verifying 161
folders ...
minor error " latime" in index 0x5 "." => "admin"
minor error " latime" in index 0xb "$Extend" => "$Reparse"
minor error " latime" in index 0x1f "public" => "EZ TALK.doc"
minor error " latime" in index 0x1f "public" => "Fedora-13-i686-Live-
KDE"
minor error " latime" in index 0x1f "public" => "Fedora-13-x86_64-Live"
minor error " latime" in index 0x1f "public" => "FEDORA~1"
minor error " latime" in index 0x1f "public" => "FEDORA~2"
minor error " latime" in index 0x1f "public" => "FTP_login_information.
doc"
minor error " latime" in index 0x1f "public" => "Reports"
minor error " latime" in index 0x1f "public" => "... 2 K.M..b."
minor error " latime" in index 0x1f "public" => "...~1"
minor error " mdtime ctime latime" in index 0x5bd
"_restore{FB5EFA8E-F7E1-4999-B498-21EEC0CF7124}" => "RP83"
minor error " latime" in index 0x676 "mungchacha_com .+LC Kung Fu Dunk"
=> "DISC2.DAT.bc!"
minor error " latime" in index 0x676 "mungchacha_com .+LC Kung Fu Dunk"
=> "DISC2D~1.BC!"
Verifying files security...
    4.83 Gb in 1492 files
    464 Kb in 163 directories
```



```
0 Kb in bad blocks in 0 fragments
90424 Kb in use by the system
65536 Kb occupied by the log file
4096 bytes in each allocation unit
182879943 total allocation units on volume
181590430 allocation units available on volume
The volume /dev/sda2 contains minor error(s).
```

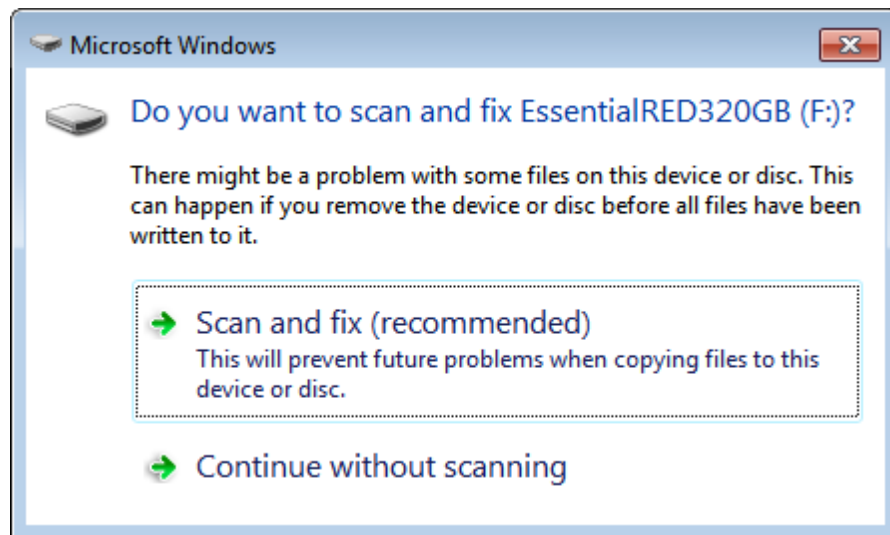
## 9.2 Warnings on Windows7/Vista when NTFS HDD is reconnected from Linux

After NTFS volume previously operated by Paragon NTFS&HFS driver is attached to Windows Vista/Windows 7 machine, warnings are displayed on the screen. Why?

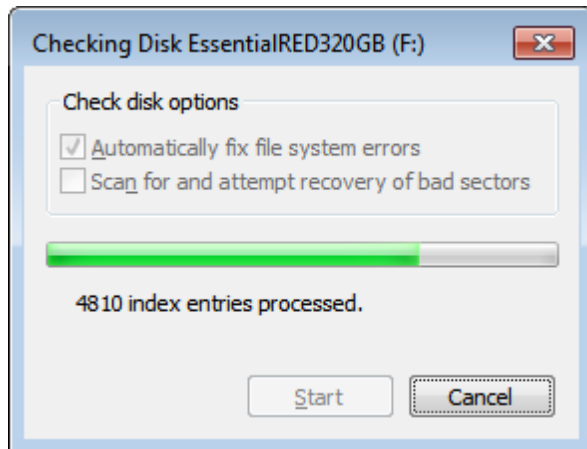
This is the case when volume was not unmounted correctly before it was detached from Linux system.

This section illustrates 'dirty' volumes handling as implemented in Windows 7. For more information on dirty flag and its support in Paragon file system drivers products see '[Dirty flag issues](#)'<sup>[15]</sup> subsection of [Using The Driver](#)'<sup>[14]</sup> section.

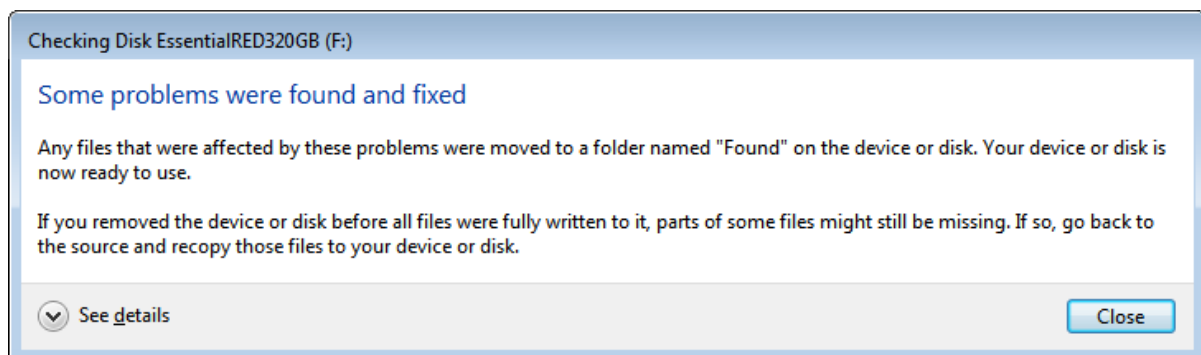
An USB HDD enclosure with 320 Gb SATA HDD with one NTFS partition was detached from system while file copy operation was in progress. After the enclosure was attached to Windows 7 PC again, the following dialog was displayed:



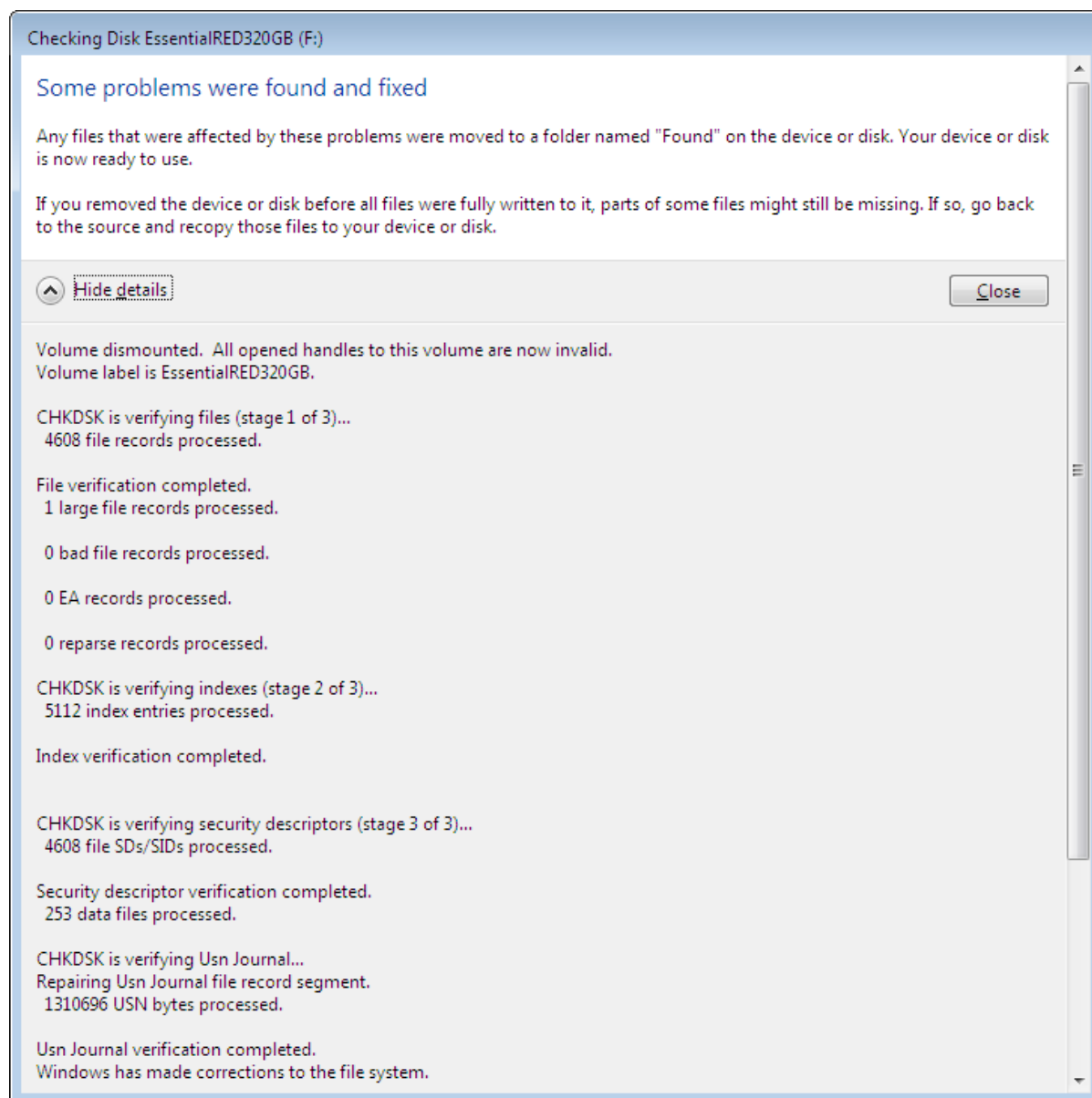
After user clicks 'Scan and fix (recommended)', scan process begins:



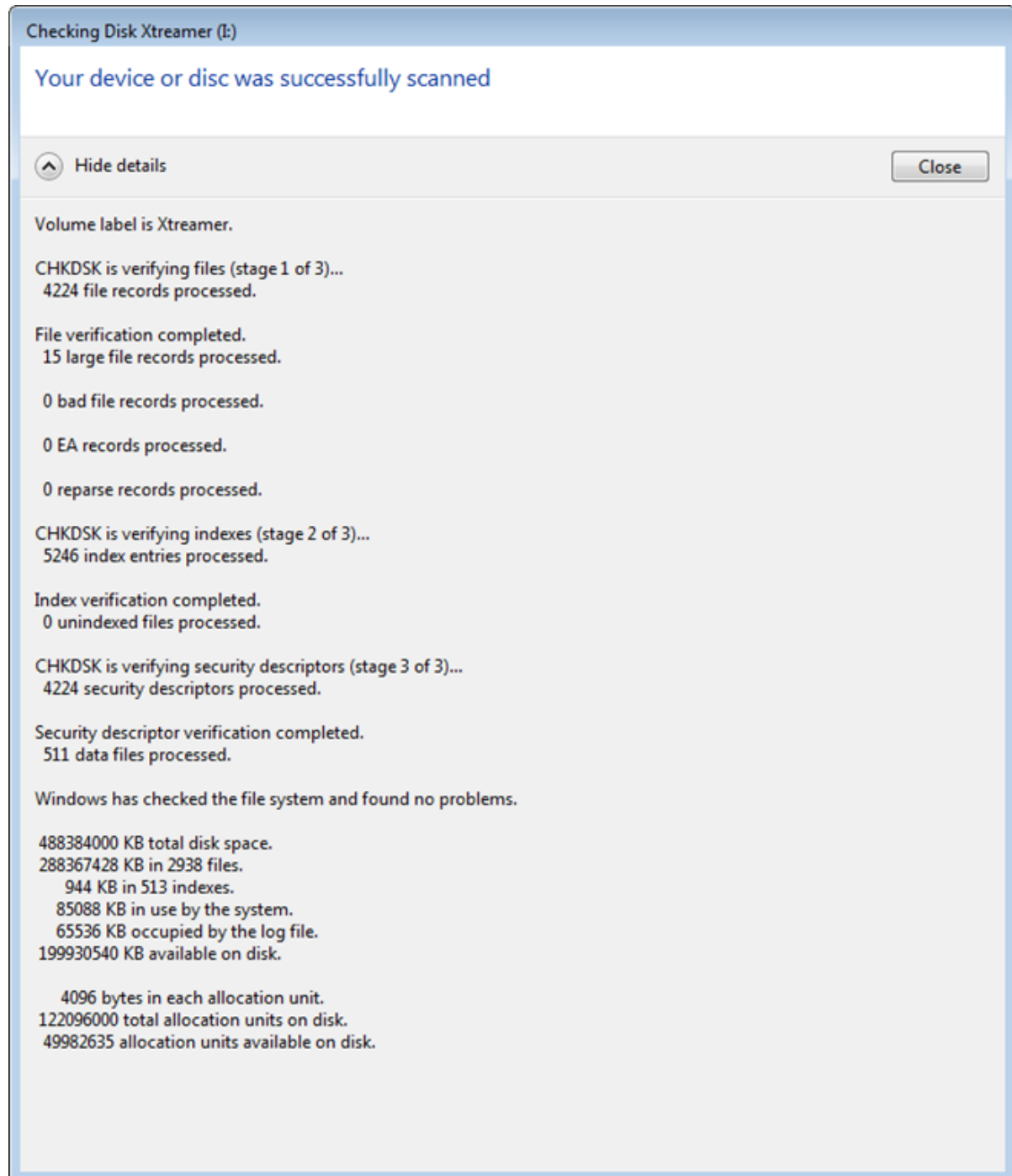
After checking is completed, the following summary window is displayed:



After user clicks 'Details', the window is expanded and more detailed information is displayed to the user:



In case there are no errors, the following information is displayed:



### 9.3 Recently changed file has its modification time a few hours ahead of or behind the current system time. Why?

This offset occurs due to the fact that NTFS stores file times as UTC time (in contrast to FAT that stores local time) and the system might not have time zone setting that can be read by C library and then used to convert file times reported by Kernel to local time.

Consequently, if a file is written to an NTFS volume on Windows with time zone set to, say, UTC+8, and then the volume is connected to the Linux system, C library reports values provided by Kernel 'as is' without converting them to local time. However, if a file is modified on the Linux system, its modification time is written to the file system as system's current time and then it is reported correctly. In the latter case, after the file modified on the Linux system is brought back to the Windows machine (with its local time zone set to UTC+8), the file's modification time will be reported 8 hours ahead of current time (assuming that current time is the same on the Linux system and Windows PC).

There is 'bias' mount option (see [Mount Options](#)<sup>[20]</sup> subsection) that allows to work around the issue on systems that do not have time zone setting readable by C standard library (first introduced in version 8.1.023). However, we recommend that time zone setting that can be used by C standard library to convert time values, is added to the Linux system.

**Part**

---



**X**

**Legal questions**

This section describes legal questions of using Paragon NTFS & HFS for Linux.

## 10.1 NTFS legal questions

Paragon NTFS&HFS for Linux driver is absolutely legal. It does not violate any patents and/or intellectual property rights. It is well known that originally NTFS was very close to the HPFS file system developed by IBM. HPFS was much more OPEN in terms of documentation support, data structure and so on. It helped us to gain a better understanding of its nature, architecture and ideology. The knowledge about NTFS we also have got has already been used for years inside our best-seller product – Paragon Partition Manager. We have sold several million copies of Paragon Partition Manager all over the world. The stability of the products as far as NTFS related operations are concerned says for itself about the stability of the NTFS technology at all. Thus, having a pretty good idea about what the HPFS file system is, we may understand the way NTFS functions.

Applying to the other sources of information like Linux drivers for NTFS and debugging Windows applications, we've documented NTFS structures from within and finally created the Universal File System Driver.

While developing Paragon NTFS&HFS for Linux driver we always stuck to the following rules:

- 1) We never applied to any confidential Microsoft NTFS stuff (docs, codes, etc.) and the reverse engineering approach for MS code.
- 2) Open sources are the only thing we used. E.g. from [www.ntfs.com](http://www.ntfs.com) we got the great part of our NTFS knowledge and understanding.
- 3) NTFS as a file system as well as on-disk layout is not patented and not documented.

## 10.2 HFS legal questions

Paragon NTFS&HFS+ for Linux driver is absolutely legal. It does not violate any patents and/or intellectual property rights. HFS+ specifications are openly published by Apple Corporation on <http://developer.apple.com/>.