Paragon NTFS&HFS+ for Linux 8.9

User Manual

Generated 10/17/2013

Paragon NTFS&HFS+ for Linux 8.9

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. List of supported NTFS/HFS+ features is given with limitations imposed by Linux. There are also advanced troubleshooting section.

PARAGON Technologie GmbH, Systemprogrammierung
Heinrich-von-Stephan-Str. 5c • 79100 Freiburg, Germany
Tel. +49 (0) 761 59018201 • Fax +49 (0) 761 59018130
www.paragon-software.com • sales@paragon-software.com

Paragon NTFS&HFS+ for Linux 8.9

User Guide

Printed: October 2013 in Freiburg, Germany.

USER MANUAL

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

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 for Linux Professional also includes useful additional utilities that provide the ability to check integrity and create NTFS/HFS+ volumes.

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 8.9 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 Ext4 FS);
- Easy installation and uninstallation (assistant scripts);
- Support for the latest Linux Kernels and distributions;
- Support for SMP kernels;
- Support for iSCSI and SSD storages;
- No system degradation during data transfers;
- All NTFS versions supported;
- Unlimited file and volume size (within NTFS/HFS+ and Kernel limitations).

What's new in Paragon NTFS&HFS for Linux 8.9:

- Support for modern Linux Kernels (up to 3.11.x);
- Support for bd_discard/TRIM feature (for iSCSI and SSD storage);
- Improved HFS+ journal support;
- Improved installation script;
- Support DKMS library usage.

NTFS-specific features:

- NTFS versions 1.2, 3.0 and 3.1 (Windows NT 4.0, 2000, XP, 2003, Vista, 7, Windows 8);
- Support for compressed files (random access for reading and writing with no limitations);
- Sparse files;

HFS+ specific features:

- Both case sensitive and case instensitive types of HFS+ file system are supported;
- During file copy operation (using cp command) on Linux only ‘data’ fork is copied.

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/7 and 8

Additional features of the Professional Edition:

- Full support of the native HFS+ journal;
- Support for the DKMS library;
- Additional NTFS utilities:
  - mkntfs [23] utility - format any partition as NTFS under Linux;
  - chkntfs [24] utility - check NTFS partition integrity and fix errors;
- Additional HFS+ utilities:
  - mkhtfs [26] utility- format any partition as HFS+ under Linux;
  - chkhfs [27] utility - check HFS+ partition for integrity and fix errors;
- Debug utility:
  - Fsdump [34] utility - debug utility for collecting volume metadata image.

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. NTFS&HFS Kernel modules occupy around 700 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 2.6.15 and newer;
- Linux with kernel versions up to 3.11.x (NTFS&HFS driver was tested with Kernels up to 3.11.3).

Linux distributions the products were tested with:

- Ubuntu 13.04;
- Debian 7.1;
- Fedora 19;
- CenOS 6.4;
- OpenSuse 12.3;
- ArchLinux 2013.10.01;

- Slackware 14.0;
- Linux Mint 15.4;
- PCLinux OS 2013.10.

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-devel` (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`
- DKMS library

  `#dkms --version`

Limitations

- GNU C compiler (gcc) version 3.3 or higher is required.
- The user should login as root to install the drivers and utilities.
- 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).

Part III

Installation

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:

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

Paragon NTFS&HFS Linux driver and utilities must be compiled on the end user's system for correct configuration. By installing the software you accept the terms of End User License Agreement listed in License file.

3.3 Installing the Drivers

First, NTFS&HFS 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 driver using `install.sh` script. Alternatively, driver binary module may be built manually using '`configure`' '`make driver `' commands.
3. Install the NTFS & HFS driver (this step will make the modules available for use);
4. Activating (loading) the driver. After building and installing, the NTFS & HFS driver can be referenced as “universal file system driver” (ufsd) 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 driver and utilities. The sections below describe the installation procedure in details.

Unpacking Setup Files

The setup files of the Linux-based version of the NTFS & HFS for Linux are provided in the form of a gzip archive. The archive should be copied to the hard disk and decompressed.

For example:

For the NTFS & HFS for Linux driver and utilities:

- create separate folder

```
# mkdir /usr/tmp/ufsd
```

- change the current directory to the new one

```
# cd /usr/tmp/ufsd
```

- use tar utility to unpack initial archive

```
# tar -xf /path/to/the/initial/archive/ufsd_*.tar.gz
```

Next step is to build and install the NTFS & HFS for Linux driver..

Using the INSTALL.SH Assistant Script

The assistant script "`install.sh`" provides the extremely easy and flexible way to make the NTFS & HFS for Linux and install driver module in the system. Additionally, the script reconfigures OS so that driver module is automatically rebuilt for another supported Kernel version (Professional edition only).

Please note that development tools and kernel sources are required to present on the system and stay in the default locations to build and install the drivers.

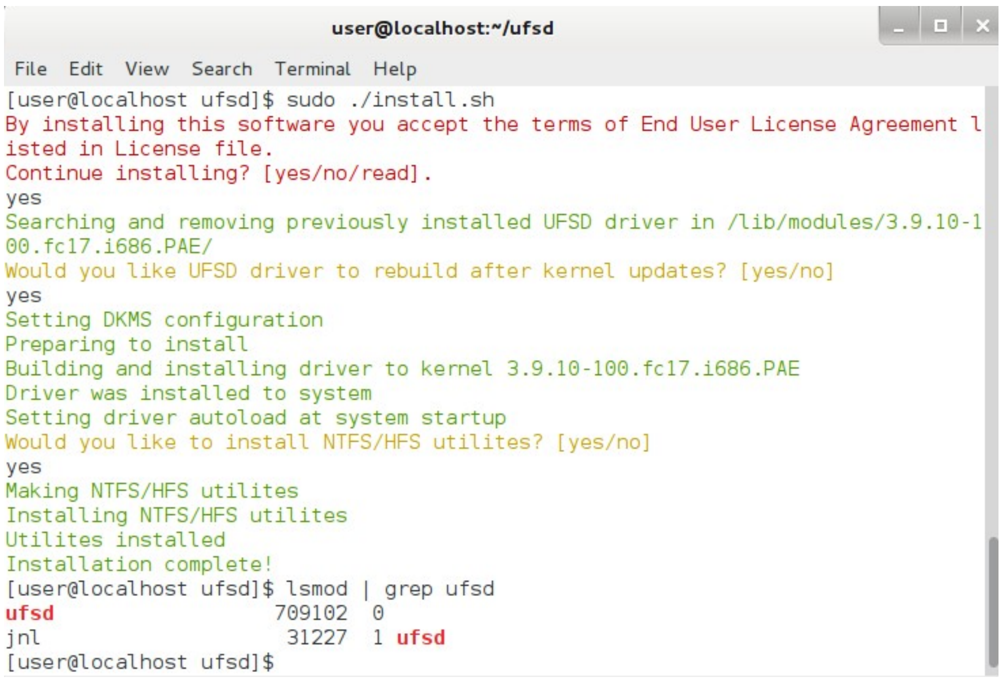
Installation

Just run the `install.sh` script with root privileges:

`# ./install.sh` or `$ sudo ./install.sh`

The assistant script will automatically perform the following actions:

1) Detect the Linux Kernel version;
2) Find kernel header files and libraries needed for building the drivers;
3) Add service for rebuilding driver module for supported Kernels via the DKMS library (Professional edition only).
4) Build the driver and binary modules;
5) Install the driver;
6) Build and install additional utilities (Professional edition only);



```
[user@localhost ufsd]$ sudo ./install.sh
By installing this software you accept the terms of End User License Agreement l
isted in License file.
Continue installing? [yes/no/read].
yes
Searching and removing previously installed UFSD driver in /lib/modules/3.9.10-1
00.fc17.i686.PAE/
Would you like UFSD driver to rebuild after kernel updates? [yes/no]
yes
Setting DKMS configuration
Preparing to install
Building and installing driver to kernel 3.9.10-100.fc17.i686.PAE
Driver was installed to system
Setting driver autoload at system startup
Would you like to install NTFS/HFS utilites? [yes/no]
yes
Making NTFS/HFS utilites
Installing NTFS/HFS utilites
Utilites installed
Installation complete!
[user@localhost ufsd]$ lsmod | grep ufsd
ufsd                  709102  0
jnl                    31227  1 ufsd
[user@localhost ufsd]$
```

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);

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

3.4 Uninstalling the Drivers

To completely remove the drivers and the utilities from the current Kernel, one should dismount all NTFS/HFS+ partitions mounted with the driver, uninstall the drivers and unload binary modules from the Kernel.

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. If some NTFS/HFS+ partitions are in use, the script (for the NTFS&HFS for Linux 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
5. Uninstall utilities (for Professional version only).

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 supported partitions and provides access to their whole 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 toubleshooting 30 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 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 or chkhfs 27 subsections);
     - Use ‘force’ mount option (see Dirty flag issues subsection).
  2. The disk is a GPT-partitioned disk – check GPT issues 15 subsection for more information.
  3. Follow other steps on the Mount toubleshooting 30 diagram to find the cause of the issue.

Analyze returned status and check output of (**dmesg | tail**). In case of failure, follow the Mount toubleshooting 30 diagram to find possible causes and and try to mount the partition again using the same or different mount options, if needed (see Mount options 19 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/sdb1 /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&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 23 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` succeedes:

```
/ # 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,iocharset=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 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).

4.6 Unmounting NTFS/HFS+ Partitions

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

```
umount /dev/hdb1
```

4.7 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/nls`** 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 **`codepage`**/**`charset/nls`** parameter is used for this purpose.

Examples:

1. Mounting a partition:

```
mkdir /mnt/test
mount -t ufsd /dev/sda6 /mnt/test
```

2. Dismounting a partition:

```
umount /mnt/test
```

3. Mounting partition in read-only mode:

```
mount -t ufsd -o ro /dev/sda6 /mnt/test
```

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

```
mount -t ufsd -o nls=utf8 /dev/sdb1 /mnt/test
```

For more information on mount options please refer to the Mount options 19 sub-section.

Part

V

Mount options

This section describes mount options for mounting supported file system partitions.

5.1 Mount options

SYNOPSYS

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

Option	NTFS	HFS+	Expected behavior and examples

`iocharset` or `nls` or `codepage`	+	+	`-o iocharset={NAME1}[,iocharset={NAME2}]`
			`-o nls={NAME1}[,nls={NAME2}]`
			`-o codepage={NAME1}[,codepage={NAME2}]`
			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.
			Note:
			▪ Paragon driver uses extended UTF-8 for Unicode number U+10000 characters support when '=utf8' is specified.
			▪ Codepage, nls and iocharset mount option must be used in the form: codepage=... nls=cp... iocharset=cp...
			Examples:
			• `codepage=950`
			• `nls=cp950`
			• `iocharset=cp950`

`nocase`	+		`-o nocase` All file and directory operations (open, find, rename) are case insensitive. Casing is preserved in the names of existing files and directories.
`showmeta`	+	+	`-o showmeta` Use this parameter to show all meta-files (System Files) on a mounted NTFS/HFS+ partition. By default, all meta-files are hidden.
`noatime`	+	+	`-o noatime` 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.

Option	NTFS	HFS+	Expected behavior and examples
`uid`	+	+	**`-o uid={USERID}`** 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.
`gid`	+	+	**`-o gid={GROUPID}`** 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.

`umask`	+	+	**`-o umask={VALUE}`**
			The default permissions given to a mounted NTFS/HFS+ volume are **`rwx------`** (for security reasons). The umask option controls these permissions for files/directories created after the volume is mounted.
			`mount -t ufsd /dev/hda1 /mnt/ntfs_0 -o umask=0222`

`fmask`
`dmask`
	+	+	**`-o fmask={VALUE}`**
			`-o dmask={VALUE}`
			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 **`umask=000,fmask=000,dmask=000`** is usually specified.

`ro`	+	+	To mount an NTFS/HFS+ volume in read-only mode.
`bestcompr`	+		Instructs the driver to use highest compression level when writing compressed files. High CPU-load.
`nobuf`	+	+	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
`sparse`	+		Create new files as “sparse”. This feature allows creating holes inside new created files (avoids filling unwritten space with zeroes). This option is not recommended in case NTFS partition is used for BitTorrent downloads.
`force`	+	+	Not recommended for use. 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. Note that if ‘dirty’ volume was mounted with ‘force’ mount option, dirty flag will not be cleared when volume is unmounted using **`umount`** command.

Option	NTFS	HFS+	Expected behavior and examples
`nohidden`	+		Files with the Windows-specific HIDDEN attribute will not be shown under Linux.
`sys_immutable`	+		Files with the Windows-specific SYSTEM attribute will be marked as system immutable files.

`acl`
`acl={1|0}`
	+	+	Support POSIX ACLs (Access Control Lists). Effective if supported by Kernel.
			The option specified as **`acl`** or **`acl=1`** enables support for POSIX ACLs; **`acl=0`** disables it.

`user_xattr`
`user_xattr={1|0}`
	+	+	Support user.* extended attributes. Effective if supported by Kernel.
			The options specified as **`user_xattr`** or **`user_xattr=1`** enables support for user.* extended attrubutes; **`user_xattr=0`** disables it.

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 from your Linux OS. Additional utilities for Paragon NTFS&HFS for Linux were developed with Paragon UFSD SDK.

6.1 NTFS utilities

There are 2 basic utilities for NTFS file system:

- mkntfs 23 — format any partition as NTFS under Linux;
- chkntfs 24 — check NTFS partition for integrity and (optionally) fix errors;

6.1.1 mkntfs

MKNTFS utility creates NTFS volumes (1.2, 3.0, 3.1 (Windows NT 4.0/2000/XP/2003/Vista/7) file system) on user specified (block) device (disk partition) under Linux OS.

Synopsis

```
mkntfs [options] device
```

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

Options

`-v:label`	Specify volume label.
`-c`	Files created on the new volume will be compressed by default.
`-a:size`	Override the default allocation unit size. Default settings are strongly recommended for general use. NTFS supports 512, 1024, 2048, 4096, 8192, 16K, 32K, 64K. File compression is not supported for allocation unit size above 4096.
`-b:size`	Override the default block (sector) size. Default settings are strongly recommended for general use. One can use 512, 1024, 2048, 4096.
`-m:size`	Override default MFT record size. Default settings are strongly recommended for general use. One can use 512, 1024, 2048, 4096.
`-f`	Force the format without confirmation.
`-s:start`	Specify “hidden” sectors in the boot area.
`-g:tracks:sectors`	Specify disk geometry that should be written in the boot area.
`tracks`	specifies number of tracks per disk side.
`sectors`	specifies number of sectors per track.
	The most popular 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 (`-g:255:63`). If –g is not specified, the utility obtains geometry from OS.
`-winxp`	Create NTFS compatible with Windows XP (default)
`-winvista`	Create NTFS compatible with Windows Vista
`-win7`	Create NTFS compatible with Windows 7
`--help`	Display this help.
`--trace`	Turn on UFSD trace.
`--verbose`	Explain what is being done.
`--nopercents`	Do not print percents during format process.
`--version`	Show the version and exit.

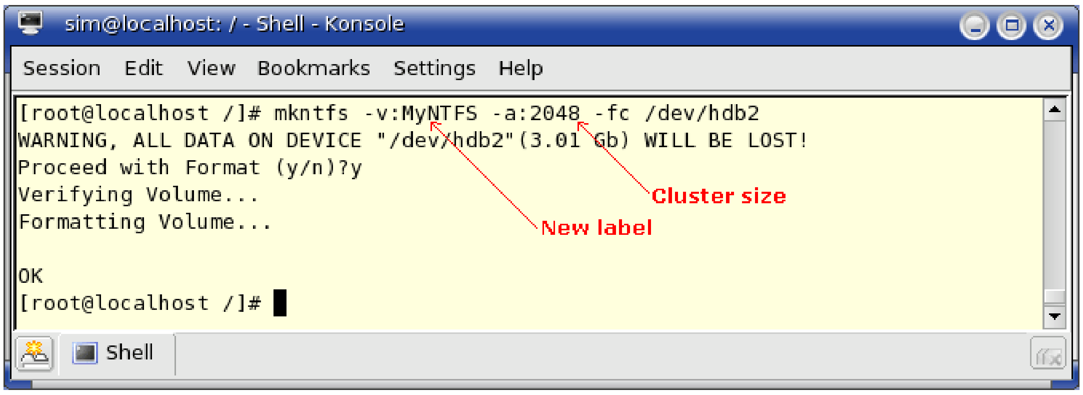
Description

`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/7/8) file system on a device (usually a disk partition).

Note: mkntfs doesn't change the MBR (Master Boot Record) when formatting a partition. Therefore, most of Linux commands (like fdisk -l) will be unable to determine that partition's files system was changed to NTFS.

Screenshots

Making NTFS partition


sim@localhost: / - Shell - Konsole
Session Edit View Bookmarks Settings Help
Cluster size
New label
Shell

```
[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 /]#
```

6.1.2 chkntfs

CHKNTFS utility performs consistency checking of NTFS volumes and (optionally) fixes errors.

Synopsis

```
chkntfs [options] device
```

E.g.: `chkntfs -f /dev/hdb1`

Options

`-f`	Fix errors on the disk.
`-a`	Perform checks only if 'dirty' flag is set.
`-h`	Display this help.
`-m:size`	Memory limit used by the utility
`--short`	Minimum file system check
`--safe`	Errors are not fixed, dirty flag is cleared if no errors found
`--showminors`	Show minor errors.
`--no-orphans`	Do not restore real orphan files
`--trace`	Turn on UFSD trace.
`--verbose`	Explain what is being done.
`--nopercents`	Do not print percents during checking process.
`--version`	Show version and exit.

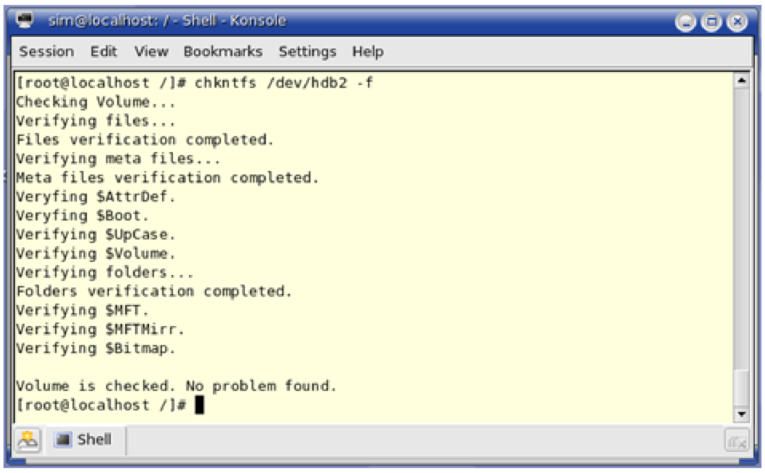
Description

`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).

Note: when `--no-orphans` option is used, real orphan files will be deleted. Without this option chkntfs restores real orphan files to found.XXX folders.

Screenshots

Verifying and fixing errors on the specified partition:


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

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

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

6.2 HFS+ utilities

There are 2 additional utilities for HFS+ file system:

- **mkhtfs** 26 — format any partition as HFS+ under Linux;
- **chkhfs** 27 — check HFS+ partition for integrity and (optionally) fix errors.

6.2.1 mkhtfs

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 -j /dev/hdb1`

Options

-v:label	Specify the volume label.
-a:size	Override the default allocation unit size. Default settings are strongly recommended for general use. HFS+ supports 512, 1024, 2048, 4096, 8192, 16K, 32K and 64K.
-ne:size	Specify extents b-tree node size: 512-32K.
-nc:size	Specify catalog b-tree node size: 4K-32K.
-f	Force the format without confirmation.
-j	Make volume journalized.
-c	Make volume case-sensitive.
-h	Display this help
--help	Display this help.
--trace	Turn on UFSD trace.
--verbose	Explain what is being done.
--nopercents	Do not print percents during format process.
--version	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 [options] device
```

E.g.: **`chkhfs -f /dev/hdb1`**

Options

`-f`	Fix errors on the disk.
`-a`	Perform checks only if 'dirty' flag is set.
`-m:size`	Memory limit used by the utility
`-h`	Display this help.
`--safe`	Errors are not fixed, dirty flag is cleared if no errors found.
`--showminors`	Show minor errors.
`--trace`	Turn on UFSD trace.
`--verbose`	Explain what is being done.
`--nopercents`	Do not print percents during checking process.
`--version`	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 6. Please also review Mount toubleshooting 30 and Using The Driver 14 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&HFS driver, prepare to report the issue to Paragon.

Collect all information on the issue

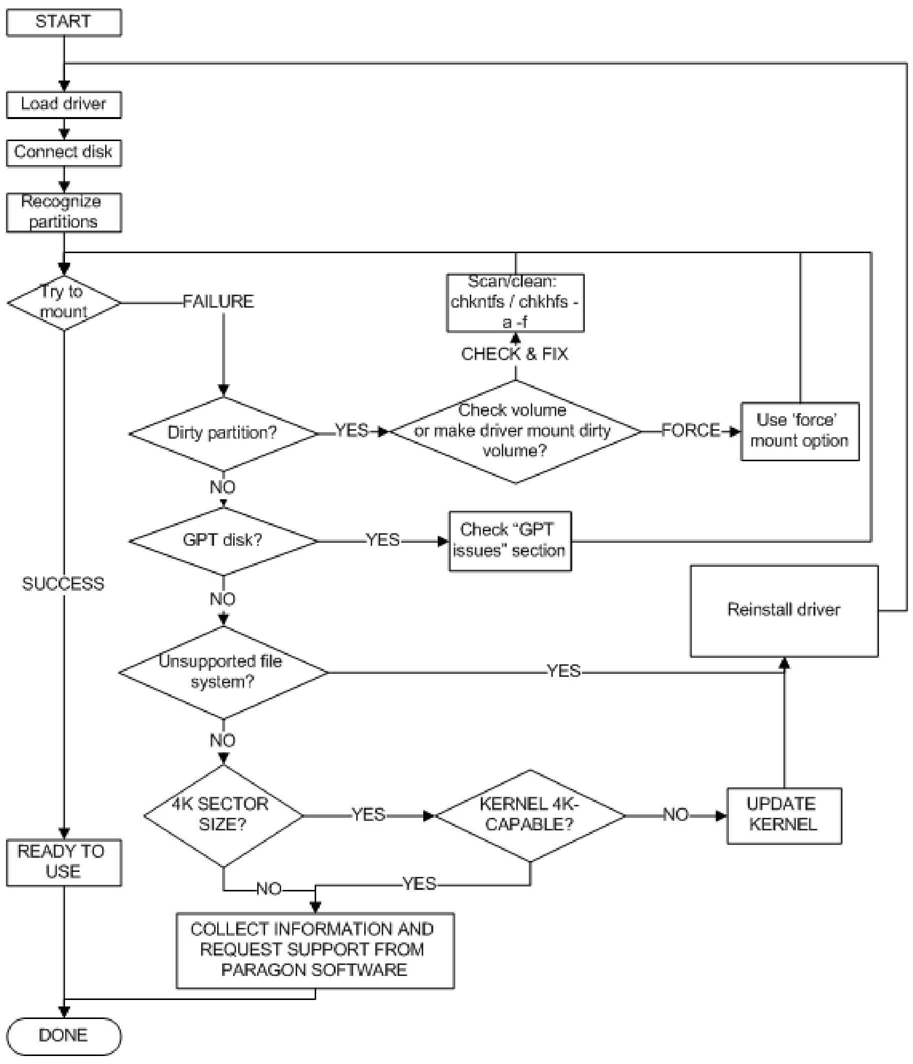
The most important point in issue resolution process is quickly obtaining all the information related to the issue. Quick collection of required information is the key to resolving an issue faster.

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 toubleshooting

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 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 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.

7.10 Collecting logs using release build of the driver user

Additional logs in the NTFS&HFS for Linux driver are used to collect information on driver's internal operations. This information is very important for faster issue resolution. In some cases, with log files available reproduction of the issue at Paragon's lab is not needed at all, that will save a lot of time. Please collect logs and send them to Paragon together with issue report.

For easier logs collection on customer's side, we have added trace functionality to the release **`ufsd.ko`** driver module.

Driver settings related to logging

Logs collection can be set up when driver is loaded into the platform (with the **`insmod`** command), during volume mount (similar to the debug driver usage) or manually via the **`/proc`** files usage:

- `# insmod ufsd.ko trace=all log=/tmp/ufsd.log`
- `# mount -t ufsd -o trace=all,log=/tmp/ufsd_trace.log`
- `# echo "/tmp/ufsd_trace.log" > /proc/fs/ufsd/log`

  `# echo "all" > /proc/fs/ufsd/trace`

where **`trace`** - level of tracing messages (possible values are: **`all, vfs`**, or hex value can be used, e.g.: **`0xffffffff`**)

`log` - path to log file

`Trace` and **`log`** parameters can be verified and changed on the fly, while NTFS&HFS for Linux driver is in use, via **`/proc`** interface:

- **`Trace`** option

To verify current settings for the trace option use the **`/proc/fs/ufsd/trace`** metafile:

```
# cat /proc/fs/ufsd/trace
all
```

To change trace settings use the **`echo`** command to provide new value:

```
# echo "vfs" > /proc/fs/ufsd/trace
# cat /proc/fs/ufsd/trace
vfs
```

- **`Log`** option is altered the same way

To verify current settings for the **`log`** option use the **`/proc/fs/ufsd/log`** metafile:

```
# cat /proc/fs/ufsd/log
/tmp/ufsd_trace.log
```

To change log settings use the **`echo`** command to provide new value:

```
# echo "/root/ufsd_trace.log" > /proc/fs/ufsd/log
# cat /proc/fs/ufsd/log
/root/ufsd_trace.log
```

Collecting driver logs

Please note that log file must not be stored on any volume that is mounted using Paragon driver because that will cause infinite recursion and then stack overflow.

If NTFS&HFS for Linux driver module (**`ufsd.ko`**) is loaded into the platform manually, logs collection can be enabled, when driver module is loaded into the Kernel:

```
# insmod ufsd.ko trace=all log=/tmp/ufsd.log
```

If NTFS&HFS for Linux driver is already loaded into the system, logs collection can be enabled:

- using the **`/proc`** file interface:

```
# echo "/tmp/ufsd_trace.log" > /proc/fs/ufsd/log
# echo "all" > /proc/fs/ufsd/trace
```

- using **`mount`** command options:

mount the test volume adding the following options to the list of mount options used on your system:

```
-o trace=all,log=/path-to-log-file/on-non-ufsd-mounted-volume
```

After NTFS&HFS for Linux driver logs are enabled, perform reproduction steps for the issue, so driver operations are written into the log file.

After the issue is reproduced, unmount the volume, unload **`ufsd.ko`** and **`jnl.ko`** driver modules (to flush and close log file), compress driver's log file and send it to Paragon along with the issue report and log of reproduction steps.

Use case examples:

- When issue reproduction takes a lot of time, but platform is halted after it happens (e.g. in case of the Kernel panic issues) additional **`'cycle=<log file size>'`** mount option are recommended for usage, so UFSD logs are added to the file of the fixed size. New logs would be written instead of older ones, when logs file size is reached, e.g.:

```
# mount -t ufsd -o trace=all,cycle=100M,log=/tmp/ufsd_trace.log
```

In this case file limit for the UFSD logs file is 100 MB.

- If there are several operation need to be made, before reproducing the issue, additional logging could be turned on/off on the fly via **`/proc/fs/ufsd/trace`** parameters.

For example when logging is already activated and is needed to be temporary turned off:

```
# cat /proc/fs/ufsd/trace
all
# echo "0" > /proc/fs/ufsd/trace
# cat /proc/fs/ufsd/trace
0
# echo "all" > /proc/fs/ufsd/trace
# cat /proc/fs/ufsd/trace
all
```

In the UFSD driver log file you will see trace level information:

`trace mask set to 5fffbf7f` (when trace level is set to 'all')
`trace mask set to 00000000` (when trace logs are turned off)
`trace mask set to 5fffbf7f` (when trace level is returned to 'all')

7.11 Fsdump utility

Collecting images of volumes with file system inconsistencies

In case Paragon chk*fs utilities cannot fix metadata inconsistencies on NTFS/HFS+ volumes, or report that volume is OK while it is obviously not so (e.g. content of NTFS volume cannot be read under MS Windows or HFS+ volume cannot be read under MacOS), **`fsdump`** utility can be used to capture very compact images of volumes with file system inconsistencies. The images then may be forwarded to Paragon via support service for analysis. This helps us to improve our products.

As fsdump images only include file system metadata, the risk of leaking sensitive data when using fsdump is minimized.

Using Fsdump utility to capture metadata from NTFS/HFS+ volumes

Using Fsdump on Linux

1. Extract fsdump utility to a folder on the drive with at least 5 Gb free space.
2. In terminal, change to a folder where fsdump was extracted. Type the following command:

```
#./fsdump /path/to/partition > volume_metadata.bin
```

3. This will create file named volume_metadata.bin containing file system structures that may be used to duplicate file system structure on another volume.
4. Fsdump utility needs to access the partition directly, so root permissions are required for this operation. The partition must be unmounted before dumping metadata to file.

5. After binary image (volume_metadata.bin) is ready, calculate it's md5 check sum.
6. Compress binary image to save time and traffic when uploading the file to our FTP server. Image of average 100-Gb volume's metadata is around 145 Mb. It gets compressed into ~8 Mb file using gzip compression utility (this information is for estimation only). Output of fsdump utility may be redirected to gzip (see example #2 below) to compress the image 'on the fly'.
7. Send the image to our support service.

Using Fsdump to restore volume's state from fsdump dump file on Linux

1. Use `info` option of Fsdump utility to report information about the image: number of used sectors, sector size and full volume size.

```
#./fsdump volume_metadata.bin -info
<FS_NAME>, <BBBBBBBBB> (<CCCCC> x <SSS>), <VVVVV> Gb
```

where:

`<FS_NAME>` file system name (NTFS or HFS+)
`<BBBBBBBBBBB>` – volume size in bytes (hexadecimal)
`<CCCCC>` – volume size in sectors (hexadecimal)
`<SSS>` – sector size in bytes (hexadecimal)
`<VVVVV>` – volume size in Gbytes

2. Convert volume size and sector size to decimal numeration (`<CCCCC_d>` and `<SSS_d>`)
3. Prepare a disk with MBR partitioning scheme.

```
#parted -s /dev/sda mklabel msdos
```

4. Prepare partition of the needed size. It is not necessary to format the partition.

```
#parted -s /dev/sda mkpart primary 64s (63+<CCCCC_d>)s
```

5. Use fsdump utility on the partition, redirecting standard input from fsdump dump file (opposite to capturing file system's metadata):

```
#./fsdump /dev/sda1 < volume_metadata.bin
```

Notes:

- Volume size (`<BBB>`) calculates as a multiply of volume size in blocks (`<CCC>`) and block size (`<SSS>`). Volume size in Gbytes (`<VVV>`) could be even or less then the calculated volume size in bytes (`<BBB>`), depending on the file system.
- Any partitioning utility can be used for disk partition preparation. '`Parted`' utility for Linux is used only as an example.

Troubleshooting

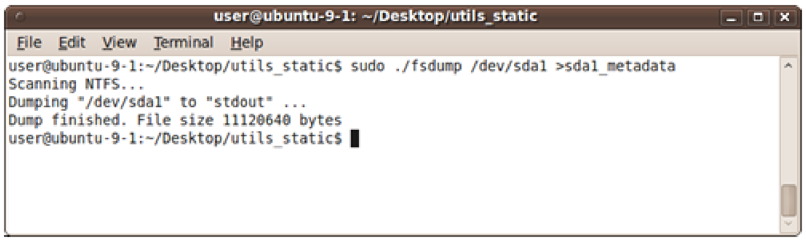
1. Check the size of the received file and compare it with the sent file size.
2. Check md5 sum of the received binary file.

3. Fsdump utility doesn't parse partition scheme of the disk, which is used for collecting metadata information. The utility only operates on a given partition. For fsdump to access necessary partition(s) on the storage medium, partitioning scheme used on the storage medium must be supported by the host operating system (most popular partitioning schemes are Master boot record (MBR), Apple partition map (APM), GUID Partition Table (GPT)).
4. The same partition on the same storage medium may appear to have different number of partitions when used with different operating systems (e.g. Windows/Linux). Therefore, one may need to use different partition numbers e.g. sdX3 or sdX4 instead of (seemingly correct) sdX2. When on Linux, it is recommended to check size of partitions using parted or gparted utility before using fsdump.

Linux Examples

Example 1

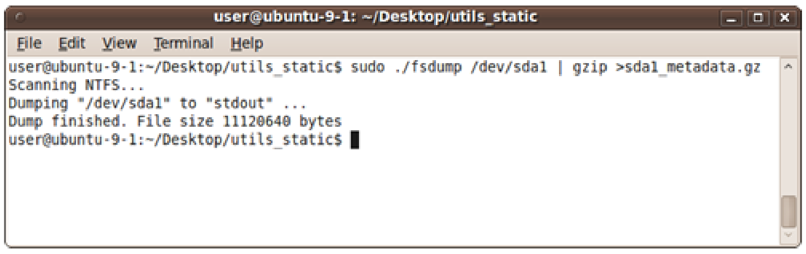
Dumping metadata image to file:


user@ubuntu-9-1: ~/Desktop/utils_static
File Edit View Terminal Help

```
user@ubuntu-9-1:~/Desktop/utils_static$ sudo ./fsdump /dev/sda1 >sda1_metadata
Scanning NTFS...
Dumping "/dev/sda1" to "stdout" ...
Dump finished. File size 11120640 bytes
user@ubuntu-9-1:~/Desktop/utils_static$
```

Example 2

Dumping metadata image through gzip to compress it ‘on the fly’:


user@ubuntu-9-1: ~/Desktop/utils_static
File Edit View Terminal Help

```
user@ubuntu-9-1:~/Desktop/utils_static$ sudo ./fsdump /dev/sda1 | gzip >sda1_metadata.gz
Scanning NTFS...
Dumping "/dev/sda1" to "stdout" ...
Dump finished. File size 11120640 bytes
user@ubuntu-9-1:~/Desktop/utils_static$
```

Part VIII

UFSD driver compatibility

This section describes file system 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.

Alternate data streams

When copying from NTFS to Linux FS: all additional streams will not be copied, along with compression flag and security attributes.

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 instensitive 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

9.1 What are 'minor errors' reported by chkntfs 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 chkntfs utility can also find the following minor errors:

mdtime — modification time

chtime — 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 chkntfs. For an example output, please see the log below:

```
# chkntfs --showminors /dev/sda2
WARNING!  f parameter not specified.
Running chkntfs 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 chtime 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).
```

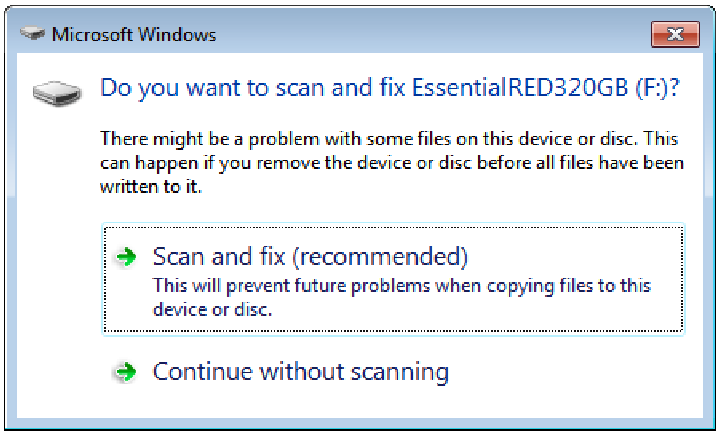
9.2 Warnings on Windows7/Vista when NTFS HDD is reconnected from Linux

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' subsection of Using The Driver 14 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:


Microsoft Windows

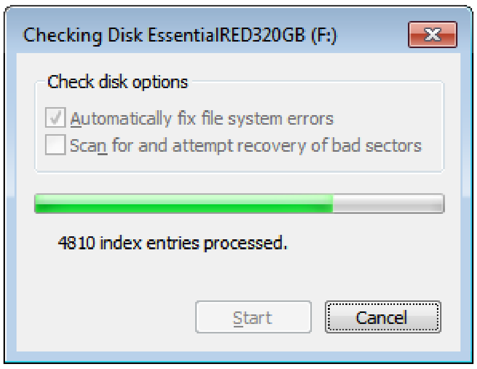
Do you want to scan and fix EssentialRED320GB (F:)?

There might be a problem with some files on this device or disc. This can happen if you remove the device or disc before all files have been written to it.

Scan and fix (recommended)
This will prevent future problems when copying files to this device or disc.

Continue without scanning

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

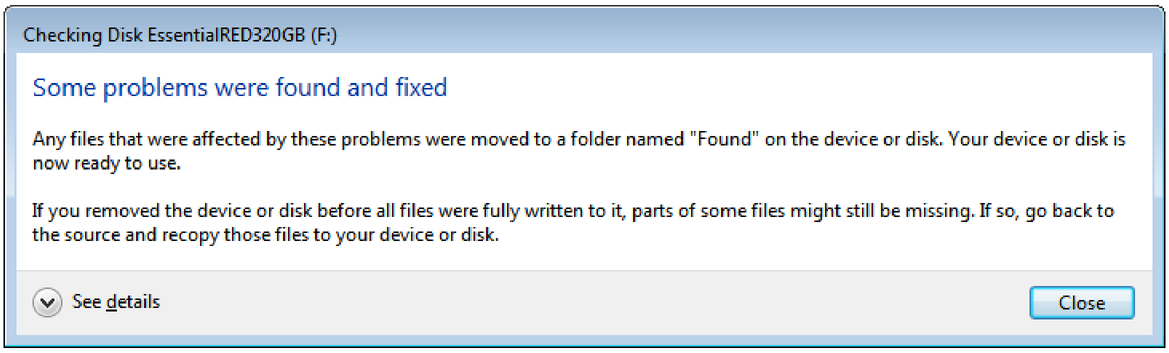

Checking Disk EssentialRED320GB (F:)

Check disk options

☑ Automatically fix file system errors
☐ Scan for and attempt recovery of bad sectors

4810 index entries processed.

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



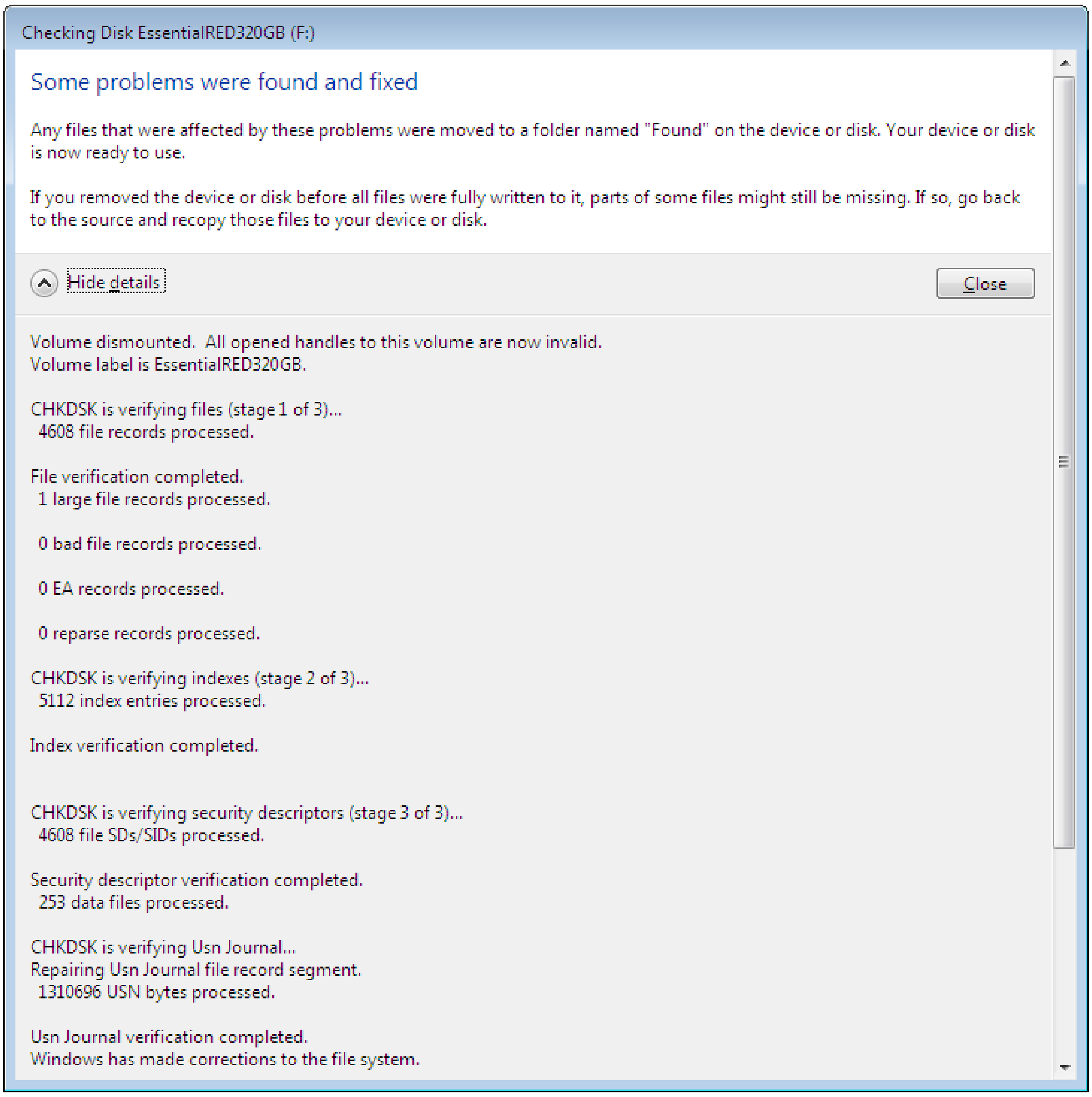
Checking Disk EssentialRED320GB (F:)

Some problems were found and fixed

Any files that were affected by these problems were moved to a folder named "Found" on the device or disk. Your device or disk is now ready to use.

If you removed the device or disk before all files were fully written to it, parts of some files might still be missing. If so, go back to the source and recopy those files to your device or disk.

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



Checking Disk EssentialRED320GB (F:)

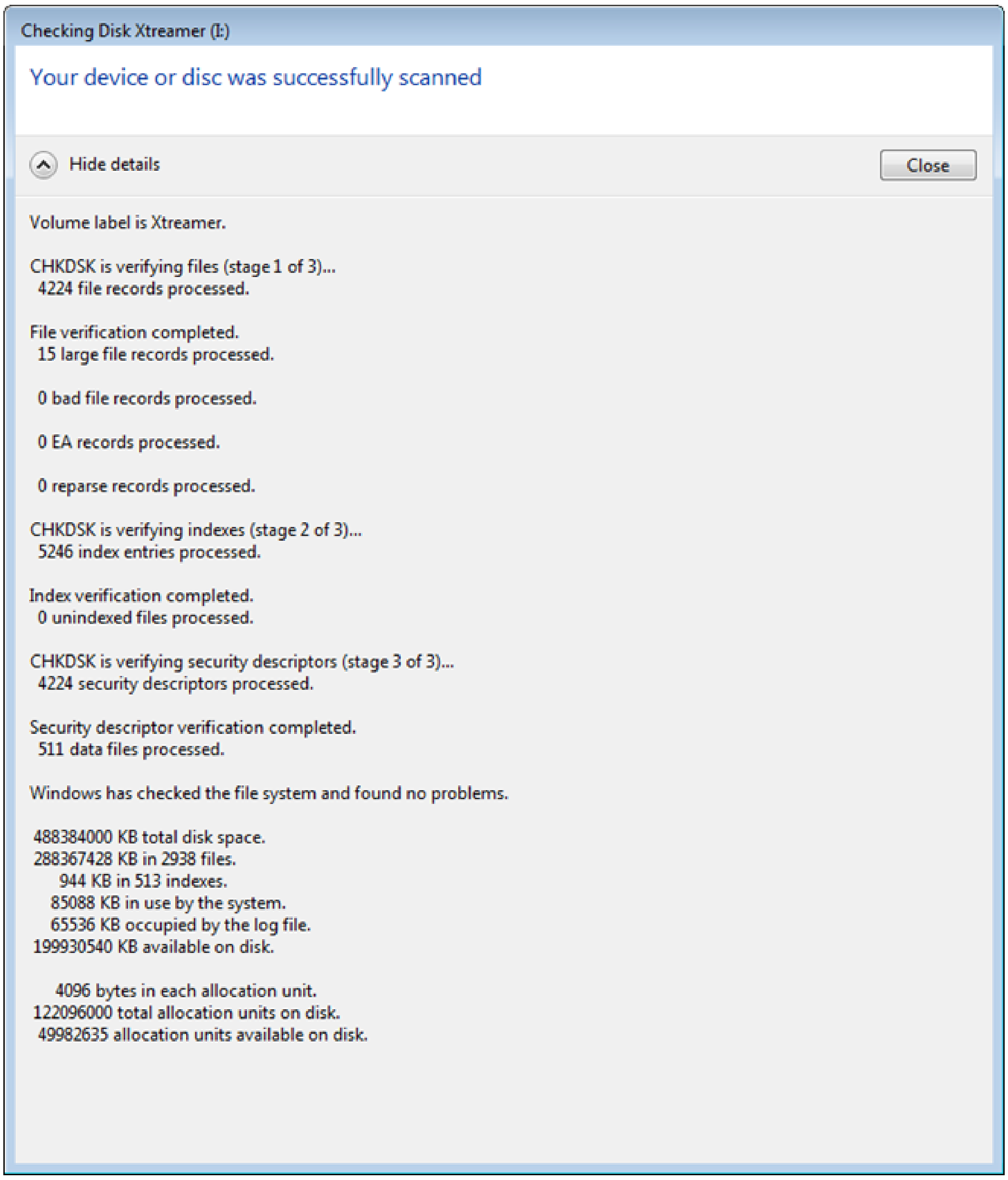
Some problems were found and fixed

Any files that were affected by these problems were moved to a folder named "Found" on the device or disk. Your device or disk is now ready to use.

If you removed the device or disk before all files were fully written to it, parts of some files might still be missing. If so, go back to the source and recopy those files to your device or disk.

```
Volume dismounted.  All opened handles to this volume are now invalid.
Volume label is EssentialRED320GB.

CHKDSK is verifying files (stage 1 of 3)...
 4608 file records processed.

File verification completed.
 1 large file records processed.

 0 bad file records processed.

 0 EA records processed.

 0 reparse records processed.

CHKDSK is verifying indexes (stage 2 of 3)...
 5112 index entries processed.

Index verification completed.


CHKDSK is verifying security descriptors (stage 3 of 3)...
 4608 file SDs/SIDs processed.

Security descriptor verification completed.
 253 data files processed.

CHKDSK is verifying Usn Journal...
Repairing Usn Journal file record segment.
 1310696 USN bytes processed.

Usn Journal verification completed.
Windows has made corrections to the file system.
```

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


Checking Disk Xtreamer (I:)
Your device or disc was successfully scanned
Hide details
Close

```
Volume label is Xtreamer.

CHKDSK is verifying files (stage 1 of 3)...
 4224 file records processed.

File verification completed.
 15 large file records processed.

 0 bad file records processed.

 0 EA records processed.

 0 reparse records processed.

CHKDSK is verifying indexes (stage 2 of 3)...
 5246 index entries processed.

Index verification completed.
 0 unindexed files processed.

CHKDSK is verifying security descriptors (stage 3 of 3)...
 4224 security descriptors processed.

Security descriptor verification completed.
 511 data files processed.

Windows has checked the file system and found no problems.

 488384000 KB total disk space.
 288367428 KB in 2938 files.
       944 KB in 513 indexes.
     85088 KB in use by the system.
     65536 KB occupied by the log file.
 199930540 KB available on disk.

      4096 bytes in each allocation unit.
 122096000 total allocation units on disk.
  49982635 allocation units available on disk.
```

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 19 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.

9.4 Why does mount option A make driver ignore mount option B?

When you mount disk with several mount options driver may ignore some of them. Why?

This issue can happen when mount command is used with several options coded like:

```
mount -t ufsd -o option A,option B -o option C device mount point
```

In this case driver may ignore options A and B when mounting disk with option C.

To prevent this possibility it is recommended to write your commands with several options like:

```
mount -t ufsd -o option A,option B,option C device mount point
```

9.5 Does the driver have an optimization for avoiding data fragmentation?

We use the driver for recording video data. This writes a huge amount of small data packs to the hard drive. Can you tell me if the driver has an optimization for avoiding data fragmentation? Or will this cause a strong fragmentation to the hard disk?

As of UFSD version 8.4, level of fragmentation mostly depends on number of files that are written to a volume simultaneously. In case only one file is written with small chunks of data, no fragmentation will occur beyond fragmentation caused by fragmentation of free space already existing on the volume. In case several files are written simultaneously, fragmentation will occur if no counter measures are taken.

One of possible countermeasures is to allocate large continuous chunks of disk space for files, by using:

```
fallocate (fd, FALLOC_FL_KEEP_SIZE, ...).
```

This routine allocates disk space for file, but does not change file size. Before closing the file, one could call `ftruncate()` to reclaim disk space that was allocated but was not written to, but that is up to developer to decide whether it is needed to reclaim the unwritten space (maybe the allocated space will be used in future write sessions, e.g. as in P2P network client software when downloading large files during several sessions). This approach also reduces CPU load thus improving performance, as there's no need for FS driver to invoke disk space allocation routines each time small block should be appended to a file.

UFSD version 8.5 has transparent feature called **`delayed allocation`** (available on 2.6.X Kernel versions with **`writeback_inodes_sb_if_idle`** routine, see **`delalloc`** 19 mount option) to prevent fragmentation even in case writes are performed in small chunks. For this feature to work, the system must have enough memory for disk/file cache and write operations must be performed in buffered mode.

9.6 Why a lot of memory is used for volume mounting?

Let me describe what's going on when UFSD driver mounts volumes.

First of all, the driver must read file system boot record, and after verifying it, it must also read some metadata from the mounted volume, namely, parts of **`$Mft`** and the entire **`$Bitmap`** metafile. For example, if the volume has **`$Bitmap`** of 74 MB, the driver has to read not less than 74MB to mount it. When our driver reads data from disk, Kernel keeps the data cached in memory. The amount of memory that Kernel allocates for I/O buffers is printed in line #3 of **`/proc/meminfo`** file (**`Buffers: XXXX kB`**). It's up to Kernel to decide how much memory to allocate for I/O buffers (this can be tuned via Kernel metafiles – please see the link below for more information). The memory (allocated for 'Buffers') is normally reclaimed by kernel when Kernel or an application needs to allocate some memory for 'private' use.

The real-time report on memory allocated by our driver for operations on specific partition that is currently mounted, is available in line #2 of file **`/proc/fs/ufsd/<block_device_name>/volinfo`**. Peak amount of memory allocated by UFSD driver when mounting NTFS volume is around 250 KB. The amount then reduced to ~40 KB after mount operation is completed.

For more information about Kernel memory consumption, please see the article: http://www.rt-embedded.com/blog/archives/linux-memory-consumption/ , where more details can be founded on the content of **`/proc/meminfo`** file and also on Kernel approach to RAM utilization.

9.7 Why the disk can't be dismounted?

When you try to dismount the disk with the '**`umount`**' command the volume is reported 'busy' and can't be unmounted. Why?

This issue can happen when there are working processes, that are still using the volume.

Therefore, there are several options to remove the conditions that prevent the storage medium from being unmounted:

1. Check if it is possible to safely unmount the storage using the system's web interface.
2. Check if support for external storage can be disabled from the system's web interface. Disable the services and retry unmounting the storage medium.
3. Check if the various file/media sharing services (like multimedia, SAMBA (SMB), AFP, etc) can be disabled from the system's web interface. Disable the services and retry unmounting the storage medium.

Notes:

- Windows system keeps SAMBA connection to the storage for several minutes. Disable the connection and retry to unmount the storage medium. For example in Windows XP it can be done from **`'Disconnect network drive'`** menu (e.g. '**`My computer`**' -> '**`Tools`**' -> '**`Disconnect Network Drive`**' menu).

If the volume couldn't be unmount with these steps, use '**`sync`**' command to flush buffers to the storage medium before detaching it manually from the device.

Part X

Legal questions

USER MANUAL

This section describes legal questions of using Paragon UFSD driver.

10.1 NTFS legal questions

Paragon UFSD 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 UFSD 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 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 UFSD 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/.

GPL statement.

Paragon journal-writing source code for the HFS+ file system is based on Journaling Block Device 2 (JBD2) implementation in Linux Kernel and released under the terms of the GNU General Public License, version 2 (http://www.gnu.org/licenses/gpl-2.0.html).

Paragon journal-writing implementation for the HFS+ file system is built as a single kernel module `jnl.ko`. Customers and other people can get Paragon journal-writing source code by sending letter to e-mail: support@paragon-software.com