

Paragon NTFS&HFS for Linux 9.0

User Manual

© 2014 Paragon Software Group

Generated 8/18/2014

Paragon NTFS&HFS for Linux 9.0

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 NTFS&HFS for Linux 9.0

User Guide

© 2014 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: August 2014 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

| | |
|---|-----------|
| Part I Introduction | 2 |
| 1 Historical review..... | 2 |
| 2 Paragon UFSD technology | 2 |
| 3 How UFSD works on Linux..... | 3 |
| 4 Key Features..... | 3 |
| Part II System requirements | 6 |
| 1 Hardware requirements..... | 6 |
| 2 Software requirements..... | 6 |
| Part III Installation | 9 |
| 1 Shipment..... | 9 |
| 2 Components..... | 9 |
| 3 Installing the Drivers..... | 9 |
| 4 Uninstalling the Drivers..... | 11 |
| Part IV Using The Driver | 14 |
| 1 Getting started..... | 14 |
| 2 Mounting NTFS/HFS+ Partitions..... | 14 |
| 3 Dirty flag issues..... | 14 |
| 4 GPT issues..... | 15 |
| 5 Issues with large HDDs..... | 16 |
| 6 Unmounting NTFS/HFS+ Partitions..... | 16 |
| 7 Choosing the codepage/charset for NTFS/HFS+ Partitions..... | 16 |
| Part V Mount options | 19 |
| 1 Mount options..... | 19 |
| Part VI Additional Utilities | 23 |
| 1 NTFS utilities..... | 23 |
| mkntfs..... | 23 |
| chkntfs..... | 24 |
| 2 HFS+ utilities..... | 25 |
| mkhfts..... | 25 |
| chkhfs..... | 26 |
| Part VII Troubleshooting | 29 |

| | |
|---|-----------|
| 1 Troubleshooting processes..... | 29 |
| 2 Mount troubleshooting..... | 30 |
| 3 The install.sh script can't find kernel sources..... | 30 |
| 4 Can't compile the NTFS/HFS+ for Linux driver..... | 31 |
| 5 "Can't load module" message at the end of installation..... | 31 |
| 6 ufsd Module: kernel-module version mismatch..... | 31 |
| 7 ufsd Module: create_module: operation is not permitted..... | 32 |
| 8 insmod: a module named as ufsd already exists..... | 32 |
| 9 I can't mount NTFS/HFS+ volume..... | 32 |
| Part VIII UFSd driver compatibility | 34 |
| 1 NTFS features..... | 34 |
| 2 HFS+ features..... | 34 |
| Part IX Frequently Asked Questions | 36 |
| 1 What are 'minor errors' reported by chkntfs utility?..... | 36 |
| 2 Warnings on Windows7/Vista when NTFS HDD is reconnected from Linux..... | 37 |
| 3 Recently changed file has its modification time a few hours ahead of or behind the current system time. Why?..... | 39 |
| 4 Why does mount option A make driver ignore mount option B?..... | 40 |
| 5 Why a lot of memory is used for volume mounting?..... | 40 |
| 6 Why the disk can't be dismounted?..... | 40 |
| Part X Legal questions | 43 |
| 1 NTFS legal questions..... | 43 |
| 2 HFS+ legal questions..... | 43 |

Part



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/HFS+. For example, open-source `ntfs-3g` NTFS driver does not support random write access to compressed files and Kernel `hfsplus` driver by default doesn't provide write support on HFS+ journalized volumes.

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, insert into `fstab` or use automounting scripts, 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 a 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 9.0 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;
- 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;
- 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 9.0:

- Support for modern Linux Kernels from 2.6.36 to 3.14.x;
- Driver performance improvements (reduced CPU load and memory consumption);
- Improved installation script;
- Auto mounting support.

NTFS-specific features:

- NTFS versions 1.2, 3.0 and 3.1 (Windows NT 4.0, 2000, XP, 2003, Vista, 7, Windows 8.1);
- Support for compressed files (random access for reading and writing with no limitations);
- Sparse files support.

HFS+ specific features:

- Both case sensitive and case insensitive 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/7/8 and 8.1 |

Additional features of the Professional Edition:

- Full support of the native HFS+ journal;
- Support for the DKMS library;
- Auto mounting support;
- 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:
 - [mkhfs](#) ^[25] utility- format any partition as HFS+ under Linux;
 - [chkhfs](#) ^[26] utility - check HFS+ partition for integrity and fix errors;

Part



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 450KB of free RAM to work with NTFS partitions larger than 250 GB. NTFS&HFS Kernel modules occupy around 800 Kb of RAM.

Real-life values

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

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

16 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 2.6.36 and newer;
- Linux with kernel versions up to 3.14.x (NTFS&HFS driver was tested with Kernels up to 3.14.4).

Linux distributions the products were tested with:

- Ubuntu 14.04;
- Debian 7.5;
- Fedora 20;
- OpenSuse 13.1;
- CentOS 7.

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 (depending on the Linux distribution commands for checking installed components may vary):

- 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 — for Professional version only;

```
#dkms --version
```

Limitations

- GNU C compiler (gcc) version 4.4 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



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:

- 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);
- Source files for automatic mounting integration (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`' and '`make driver`' commands. After kernel module is built, install/load driver modules:

```
# insmod jnl.ko
# insmod ufsd.ko
```

3. Activating 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 (refer to the [Mounting NTFS/HFS+ Partitions](#)^[14] subsection).

The steps 1-2 should be made only once while the step 3 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);
- 7) Add automatic mounting support (Professional edition only).

```
user@ubuntu: ~/ufsd
user@ubuntu:~$ sudo ./install.sh
[sudo] password for user:
By installing this software you accept the terms of End User License Agreement listed in License file.
Continue installing? [yes/no/read].
y
Searching and removing previously installed UFSD driver in /lib/modules/3.13.0-24-generic/
Would you like to mount NTFS/HFS volumes with UFSD driver automatically? [yes/no]
y
Automount configured
Would you like UFSD driver to rebuild after kernel updates? [yes/no]
y
Setting DKMS configuration
Preparing to install
Building and installing driver to kernel 3.13.0-24-generic

Good news! Module version for jnl.ko
exactly matches what is already found in kernel 3.13.0-24-generic.
DKMS will not replace this module.
You may override by specifying --force.
Driver was installed to system
Would you like to install NTFS/HFS utilities? [yes/no]
y
Making NTFS/HFS utilities
Installing NTFS/HFS utilities
Utilities installed
Installation complete!
user@ubuntu:~/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.

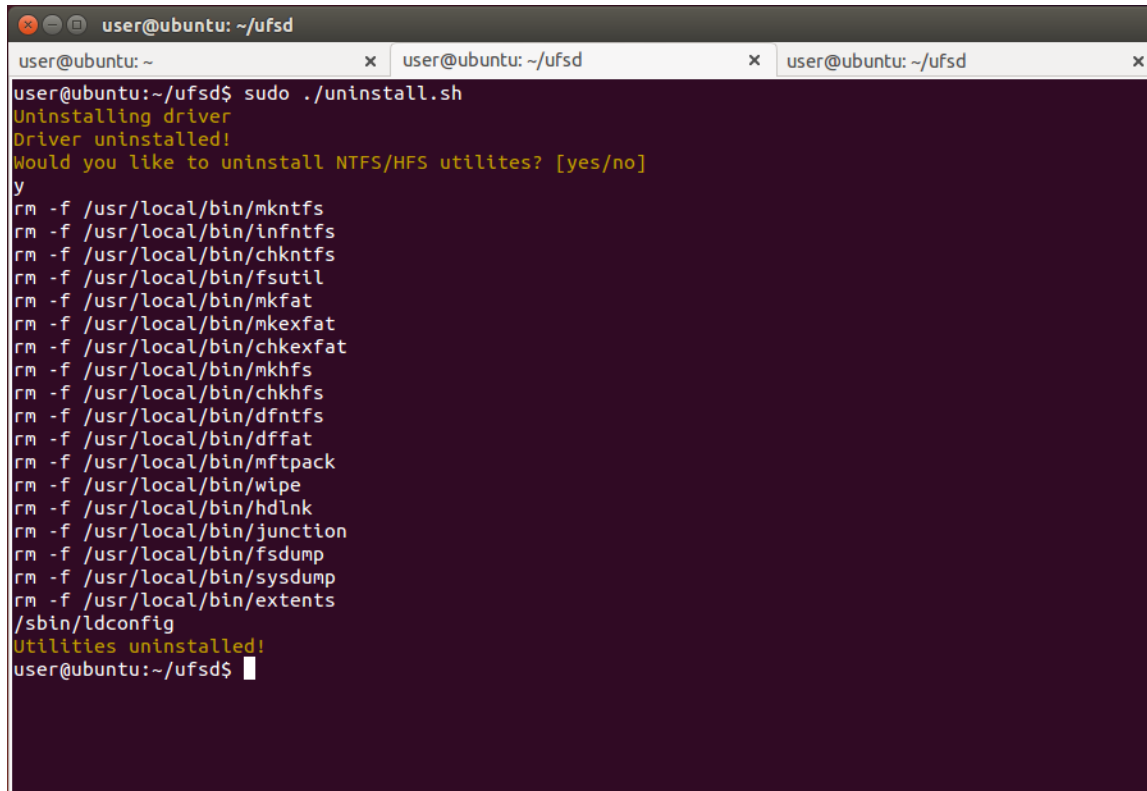
Uninstalling

Unmount all currently mounted NTFS/HFS+ partitions and run the `uninstall.sh` script:

```
# ./uninstall.sh
```


The assistant script will automatically perform the following actions:

1. Deactivate the driver modules.
2. Uninstall the drivers;
3. Remove all binary and source files of the driver;
4. Uninstall utilities (for Professional version only);
5. Remove automatic mounting settings configured for the UFSD driver.



```
user@ubuntu: ~/ufsd
user@ubuntu: ~
user@ubuntu: ~/ufsd$ sudo ./uninstall.sh
Uninstalling driver
Driver uninstalled!
Would you like to uninstall NTFS/HFS utilities? [yes/no]
y
rm -f /usr/local/bin/mkntfs
rm -f /usr/local/bin/infntfs
rm -f /usr/local/bin/chkntfs
rm -f /usr/local/bin/fsutil
rm -f /usr/local/bin/mkfat
rm -f /usr/local/bin/mkexfat
rm -f /usr/local/bin/chkexfat
rm -f /usr/local/bin/mkhfs
rm -f /usr/local/bin/chkhfs
rm -f /usr/local/bin/dfntfs
rm -f /usr/local/bin/dfdfat
rm -f /usr/local/bin/mftpack
rm -f /usr/local/bin/wipe
rm -f /usr/local/bin/hdlnk
rm -f /usr/local/bin/junction
rm -f /usr/local/bin/fsdump
rm -f /usr/local/bin/sysdump
rm -f /usr/local/bin/extents
/sbin/ldconfig
Utilities uninstalled!
user@ubuntu: ~/ufsd$
```

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 troubleshooting](#)^[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`^[26] 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 troubleshooting](#)^[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 troubleshooting](#)^[30] diagram to find possible causes 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 9 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` 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,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)
```

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/sdb1
```

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



Mount options

USER MANUAL

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

5.1 Mount options

SYNOPSIS

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

| Option | NTFS | HFS+ | Expected behavior and examples |
|--|------|------|--|
| iocharset or nls or codepage | + | + | <p>-o iocharset={NAME1} [,iocharset={NAME2}]</p> <p>-o nls={NAME1} [,nls={NAME2}]</p> <p>-o codepage={NAME1} [,codepage={NAME2}]</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"> 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... <p>Examples:</p> <ul style="list-style-type: none"> codepage=utf8 nls=utf8 iocharset=utf8 |
| nocase | + | | <p>-o nocase</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> |
| showmeta | + | + | <p>-o showmeta</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> |
| noatime | + | + | <p>-o noatime</p> <p>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.</p> |

| Option | NTFS | HFS+ | Expected behavior and examples |
|------------------------------|------|------|--|
| uid | + | + | <code>-o uid={USERID}</code> 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 | + | + | <code>-o gid={GROUPID}</code> 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 | + | + | <code>-o umask={VALUE}</code> The default permissions given to a mounted NTFS/HFS+ volume are <code>rwX-----</code> (for security reasons). The umask option controls these permissions for files/directories created after the volume is mounted. <code>mount -t ufsd /dev/hda1 /mnt/ntfs_0 -o umask=0222</code> |
| fmask dmask | + | + | <code>-o fmask={VALUE}</code> <code>-o dmask={VALUE}</code> 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 <code>umask=000</code> , <code>fmask=000</code> , <code>dmask=000</code> 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. |
| 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 <code>umount</code> command. |
| 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. |

| Option | NTFS | HFS+ | Expected behavior and examples |
|-------------------------|------|------|--|
| <code>acl</code> | + | + | Support POSIX ACLs (Access Control Lists). Effective if supported by Kernel. The option specified as <code>acl</code> enables support for POSIX ACLs; <code>acl=0</code> disables it. |
| <code>user_xattr</code> | + | + | Support user.* extended attributes. Effective if supported by Kernel. The options specified as <code>user_xattr</code> enables support for user.* extended attributes; <code>user_xattr=0</code> 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: <ul style="list-style-type: none"> • 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/ -h | 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.

6.1.2 chkntrfs

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

Synopsis

```
chkntrfs [options] device
E.g.: chkntrfs -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 |

| | |
|---------------------------|--|
| <code>--trace</code> | Turn on UFSD trace. |
| <code>--verbose</code> | Explain what is being done. |
| <code>--nopercents</code> | Do not print percents during checking process. |
| <code>--version</code> | 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.

6.2 HFS+ utilities

There are 2 additional utilities for HFS+ file system:

- [mkhtfs](#)^[25] — format any partition as HFS+ under Linux;
- [chkhfs](#)^[26] — 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

| | |
|-----------------------|---|
| <code>-v:label</code> | Specify the volume label. |
| <code>-a:size</code> | 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. |
| <code>-ne:size</code> | Specify extents b-tree node size: 512-32K. |
| <code>-nc:size</code> | Specify catalog b-tree node size: 4K-32K. |
| <code>-f</code> | Force the format without confirmation. |
| <code>-j</code> | Make volume journalized. |
| <code>-c</code> | Make volume case-sensitive. |

| | |
|---------------------|--|
| --help / -h | 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 |
| --help / -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

USER MANUAL

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 troubleshooting](#)^[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.

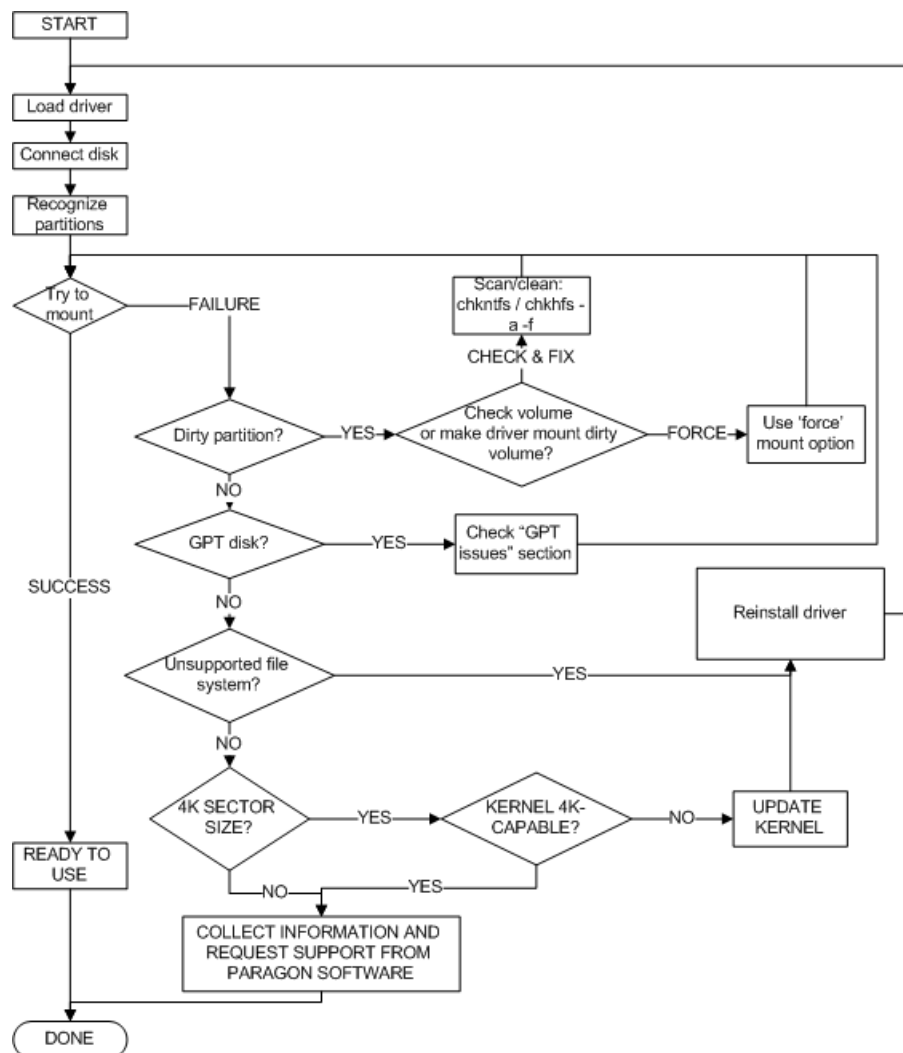
Please collect '`paragon-ufsd-install.log`' file generated by the '`install.sh`' script or '`config.log`' and console output of the building process, if modules are built by '`configure`' and '`make driver`' commands.

Step 4. Assist Paragon engineers to resolve the issue quickly

Provide logs, collected on step 3 to the Paragon engineer. Nevertheless, try to share as much additional information on the issue with our engineer, 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, 2.6.36). 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 `2.6.36` then the following lines must be found at the beginning of the **Makefile**:

```
VERSION = 2
PATCHLEVEL = 6
SUBLEVEL = 36
```

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 the kernel.

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/ufs/version
```

3. The volume is dirty. Use `chkntfs/chkhfs` utility with `-a -f` command line options to reset 'dirty' flag (refer to the [Additional Utilities](#)^[23] subsection). Alternatively, use 'force' mount options to make the driver ignore 'dirty' flag - not recommended (refer to the [Mount options](#)^[19] subsection).

Part



**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 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

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

mtime — 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 chkntrfs. For an example output, please see the log below:

```
# chkntrfs --showminors /dev/sda2
WARNING! f parameter not specified.
Running chkntrfs 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 " mtime 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).
```

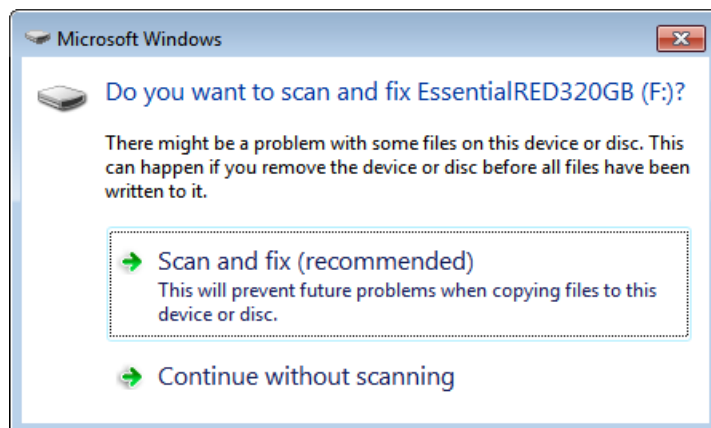
9.2 Warnings on Windows7/Vista when NTFS HDD is reconnected from Linux

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?

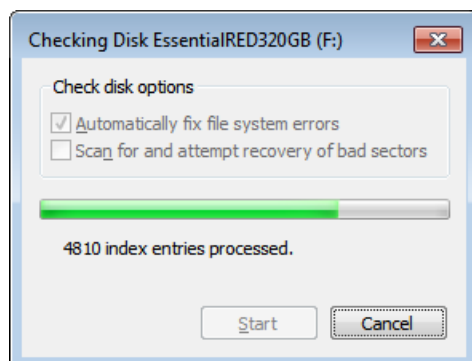
This is the case when volume was not unmounted correctly before it was detached from Linux system.

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' subsection of [Using The Driver](#)^[14] section.

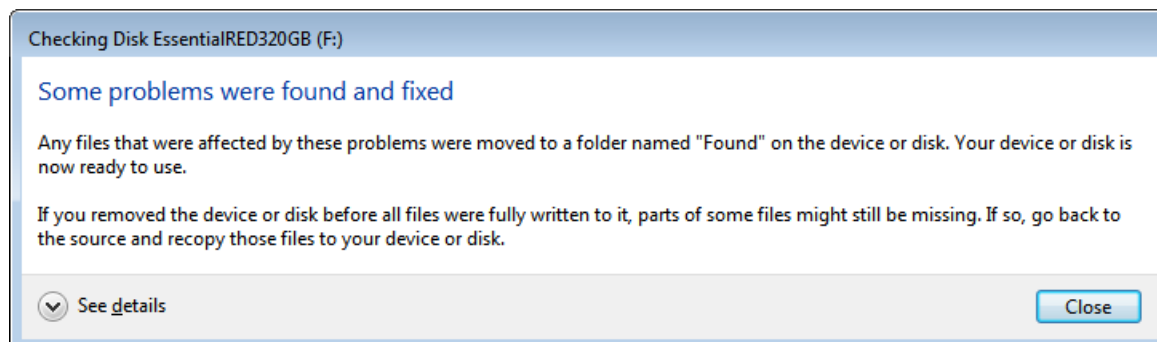
An USB HDD enclosure with 320 Gb SATA HDD with one NTFS partition was detached from system while file copy operation was in progress. After the enclosure was attached to Windows 7 PC again, the following dialog was displayed:



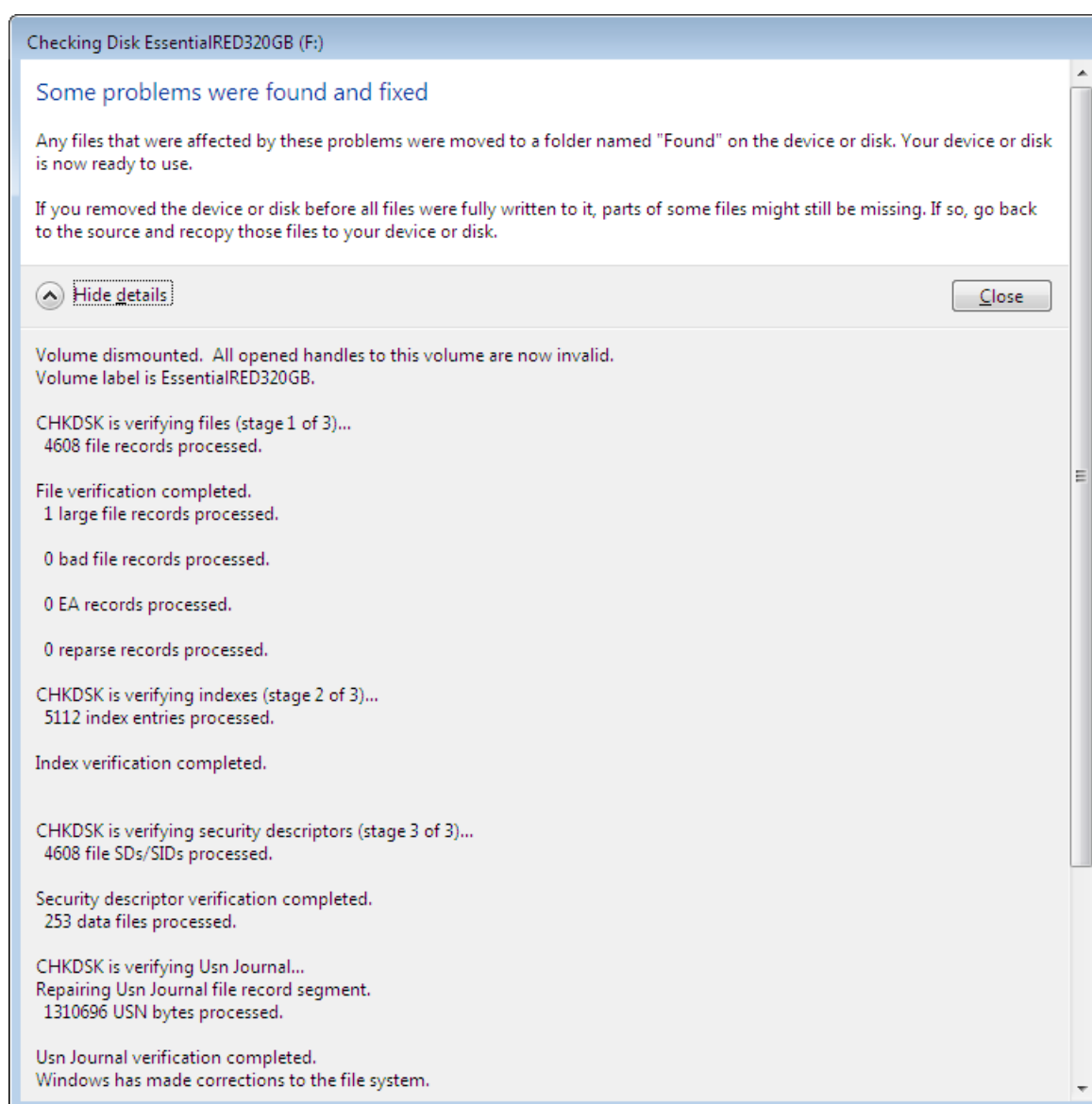
After user clicks 'Scan and fix (recommended)', scan process begins:



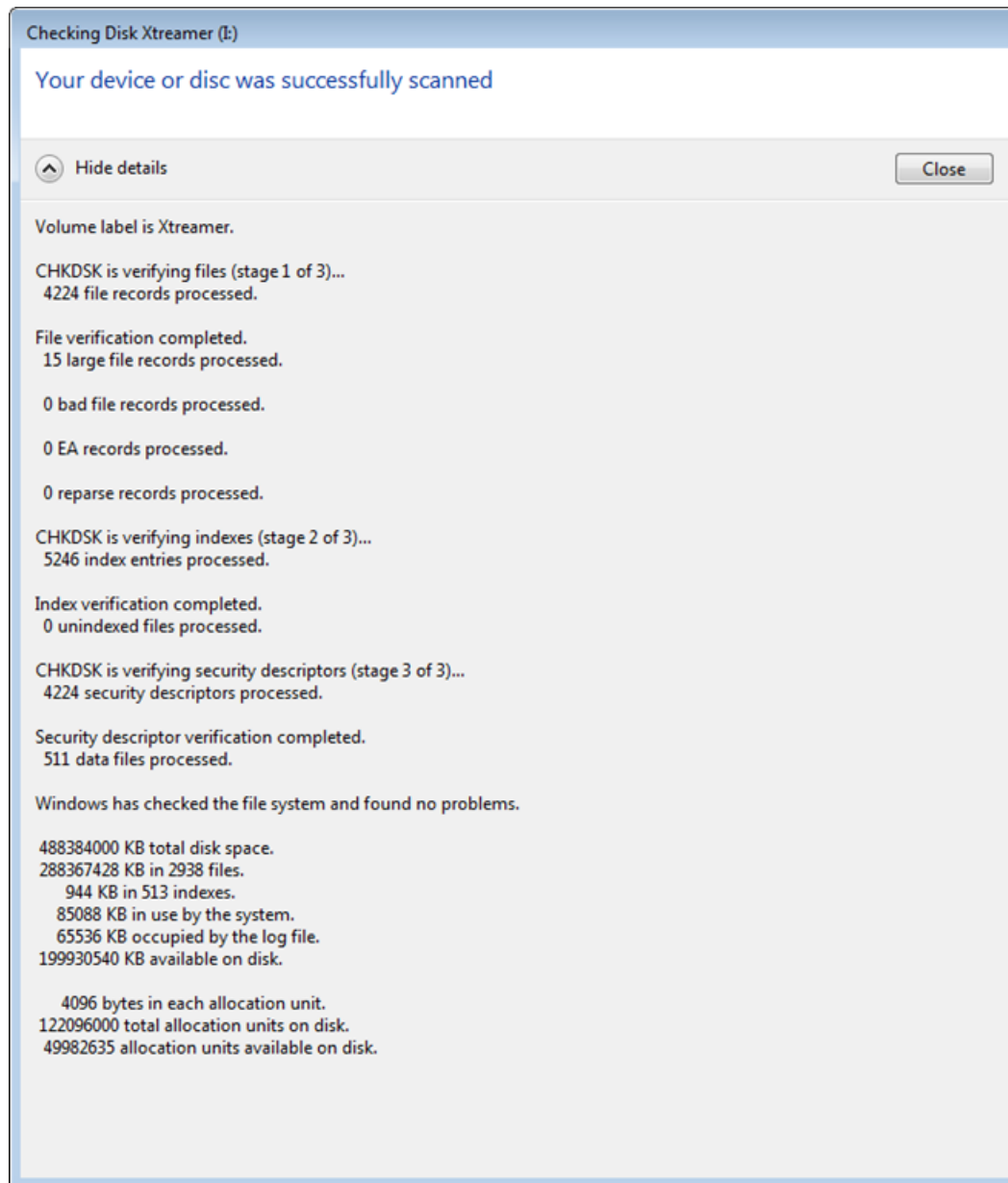
After checking is completed, the following summary window is displayed:



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).

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 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.

9.6 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. Verify that the `'umount'` command is not run from the mount point of the target volume;

2. Check if it is possible to safely unmount the storage using the system's GUI interface;
3. Check if the various file/media sharing services (like SAMBA (SMB), AFP, etc) can be disabled. 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 dismount with steps above, use the '**sync**' command to flush buffers to the storage medium, before detaching it manually from the platform.

Part



Legal questions

USER MANUAL

This section describes common legal questions of using Paragon UFSD driver.

10.1 NTFS legal questions

Paragon's 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-selling 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 speaks for itself about the stability of the NTFS technology.

After applying other sources of information like Linux drivers for NTFS and debugging Windows applications, we documented NTFS structures from within and created the Universal File System Driver.

While developing Paragon UFSD driver we always stuck to the following rules:

- 1) We never applied any confidential Microsoft NTFS materials (docs, codes, etc.) nor did we reverse engineer the MS code.
- 2) Open sources are the only thing we used. E.g. from www.ntfs.com we obtained a the great part of our NTFS knowledge and understanding.
- 3) NTFS as a file system as well as its's on-disk layout are not patented and not documented.

10.2 HFS+ legal questions

Paragon's 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 `jbd2.ko`. Customers and other interested parties can obtain access to the Paragon journal-writing source code by sending a request via e-mail to support@paragon-software.com