Paragon Technologie GmbH
Leo-Wohleb-Straße 8 • 79098 Freiburg, Germany
Tel. +49-761-59018-201 • Fax +49-761-59018-130
Website: www.paragon-software.com
E-mail: sales@paragon.software.com

Paragon NTFS&HFS+ for Linux 9.6

User manual

Abstract

This document covers implementation of NTFS & HFS+ file system support in Linux operating systems using Paragon NTFS & HFS+ file system driver. 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 is also advanced troubleshooting section.

Information

Printed: March 2019 in Freiburg, Germany.

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.

We welcome your feedback
Please send your feedback to your Paragon contact or to sales@paragon-software.com.

Paragon Technologie GmbH
Leo-Wohleb-Straße 8, 79098 Freiburg, Germany

Contents

1 Introduction **4**
1.1 Historical review . . . 4
1.2 Paragon UFSD technology . . . 4
1.3 How UFSD works on Linux . . . 5
1.4 Key features . . . 5

2 System requirements **7**
2.1 Hardware requirements . . . 7
2.2 Software requirements . . . 7

3 Installation **9**
3.1 Shipment . . . 9
3.2 Components . . . 9
3.3 Installing driver . . . 9
3.4 Uninstalling driver . . . 11

4 Using The Driver **12**
4.1 Getting started . . . 12
4.2 Mounting and unmounting partitions . . . 12
4.3 Dirty flag issues . . . 13
4.4 GPT issues . . . 13
4.5 Issues with large HDDs . . . 14

5 Mount options **15**
5.1 Mount options . . . 15

6 Additional Utilities **19**
6.1 ufsd utility . . . 19
6.2 chkufsd . . . 19
6.3 NTFS utilities . . . 20
6.3.1 mkntfs . . . 20
6.3.2 chkntfs . . . 22
6.4 HFS+ utilities . . . 23
6.4.1 mkhfs . . . 23
6.4.2 chkhfs . . . 24

7 Troubleshooting **26**
7.1 Troubleshooting processes . . . 26
7.2 Mount troubleshooting . . . 27
7.3 The install.sh script can’t find kernel sources . . . 27
7.4 Can’t compile the NTFS/HFS+ for Linux driver . . . 28
7.5 “Can’t load module” message at the end of installation . . . 28
7.6 ufsd module: kernel-module version mismatch . . . 28
7.7 ufsd module: create_module: operation is not permitted . . . 29
7.8 insmod: a module named as ufsd already exists . . . 29
7.9 insmod: Unknown symbol jnl_op (err0) . . . 29
7.10 Can’t mount NTFS/HFS+ volume . . . 29
7.11 Hardware issues . . . 30

8 Sysdump utility **31**
8.1 Introduction . . . 31

8.2 Main functions . . . 31
8.3 Using the sysdump utility to collect both platform system information and metadata . . . 31
8.4 Changing the name of output archive . . . 32
8.5 Output file description . . . 32
8.6 Delivering collected data to Paragon . . . 32
8.7 Privacy policy . . . 32

9 UFSD driver compatibility 34
9.1 NTFS features . . . 34
9.2 HFS+ features . . . 34

10 Frequently Asked Questions 35
10.1 What are 'minor errors' reported by chkntfs utility? . . . 35
10.2 Warnings on Windows7/Vista when NTFS HDD is reconnected from Linux . . . 36
10.3 Recently changed file has its modification time a few hours ahead of or behind the current system time. Why? . . . 38
10.4 Why does mount option A make driver ignore mount option B? . . . 39
10.5 Does the driver have an optimization for avoiding data fragmentation? . . . 39
10.6 Why a lot of memory is used for volume mounting? . . . 40
10.7 Why the disk can't be dismounted? . . . 40

1 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+ for Linux 9.6 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 NTFS&HFS+ for Linux 9.6 is a 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 9.6 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, exFAT, HFS+, APFS 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, HFS+, APFS and exFAT partitions under Linux, thus getting access to its contents, just the way it is implemented in the NTFS&HFS+ or APFS 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.

This software product contains components, which are partly subject to the license terms of the GNU General Public License or GNU Lesser General Public License ("LGPL"). You can request the modified source code of this software via a contact request: https://www.paragonsoftware.com/contact.html. The offer is valid for at least 3 years from the date of the publication of the corresponding software product. We deliver the software on CD/DVD or USB stick, whose production costs we claim in return.

Note: NTFS, HFS+, APFS and exFAT 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 9.6 includes NTFS & HFS+ driver for Linux environment. Once appropriate components of Paragon NTFS&HFS+ for Linux 9.6 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 9.6 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 iSCSI and SSD storages;
- File sharing over network via SAMBA;
- Low CPU load during data transfers;
- Unlimited file and volume size (within NTFS/HFS+ and Kernel limitations).

What's new in Paragon NTFS&HFS+ for Linux 9.6

- Support for Kernel versions from 2.6.36 up to 4.20.x;
- Improved "sparse" mount option for NTFS;
- Overall stability improvements;
- Simultaneous workability for NTFS/HFS+ and APFS drivers.

NTFS specific features:

- NTFS versions 1.2, 3.0 and 3.1 (Windows NT 4.0, 2000, XP, 2003, Vista, 7, 8.1, 10, Server 2016, Server 2019);
- 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/7/8.1/10 and Server 2016/2019

Additional features of the Professional Edition:

- Full support of the native HFS+ journal;
- Automatic driver rebuild for newer supported Kernels (support for the DKMS library);
- Automatic NTFS/HFS+ volume mounting;
- Additional utilities:
  - ufsd utility - single binary for all file system utilities;
  - chkufsd - single binary for checking every supported file system;
  - mkntfs - format any partition as NTFS under Linux;
  - chkntfs - check NTFS partition integrity and fix errors;
  - mkhfs- format any partition as HFS+ under Linux;
  - chkhfs - check HFS+ partition for integrity and fix errors;
  - Sysdump utility - debug utility for collecting volume metadata image and debug system information.

2 System requirements

This topic highlights requirements to hardware and software that may be used to run Paragon NTFS&HFS+ for Linux 9.6.

2.1 Hardware requirements

Minimum hardware requirements:

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

Due to unique technology NTFS&HFS+ for Linux drivers have low system requirements. For example, it is enough for our driver to have 650KB 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:

- 200 KB while executing 5 commands like `'dd if=/dev/zero of=/mnt/sda1/test bs=1M count=1000&'` in background.
- 516 KB while executing `'rsync -r /home /mnt/sda1'` command.
- 17 MB 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 Kernel versions

- Linux with kernel versions 2.6.36 and newer;
- Linux with kernel versions up to 4.20.x (NTFS & HFS+ driver was tested with Kernels up to 4.20.10).

Linux distributions the products were tested with:

- Ubuntu 18.04;
- Debian 9.8;
- Fedora 29;
- OpenSuse Leap 15.0;
- CentOS 7.6.

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:

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

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

- GNU glibc-static (recommended for Professional version only);
- DKMS library (for Professional version only).

```
#dkms --version
```

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

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

- GNU C compiler;

```
#gcc --version
```

- GNU Make;

```
#make --version
```

- GNU ld (binutils);

```
#ld --version
```

- Modutils (module-init tools).

```
#insmod -V
```

Limitations

- GNU C compiler (gcc) version 4.9 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).

3 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);
- 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 driver

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 Paragon NTFS&HFS+ for Linux 9.6 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 & HFS+ partitions.

The steps 1-3 should be made only once while 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 a separate folder:

```
$ mkdir /usr/tmp/ufsd
```

- change the current directory to the new one

```
$ cd /usr/tmp/ufsd
```

- use tar utility to unpack the 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 can reconfigure OS so that driver module is automatically rebuilt for another supported Kernel version (Professional edition only)and NTFS & HFS+ volumes are automatically mounted with the UFSD driver(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 driver binary modules (`jnl.ko` and `ufsd.ko`);
5. Install the driver;
6. Add automatic mounting settings for NTFS/HFS+ volumes (Professional edition only);
7. Build and install additional utilities (Professional edition only);

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:

```
$ sudo mount -t ufsd <device> <mount_point>
```

3.4 Uninstalling driver

To completely remove the driversand utilitiesfrom 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/utilitiesuninstall automation.
The assistant script `uninstall.sh` completely removes the drivers/utilitiesfrom the system.

Using the UNINSTALL.SH Assistant Script

The assistant script `uninstall.sh` provides the extremely easy and flexible way to deactivate and remove the driversand utilitiesfrom the system. The script performs correct deactivation, uninstallation and the complete removal of driver'sand utilities' files.

Uninstalling

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

```
$ sudo ./uninstall.sh
```

The assistant script will automatically perform the following actions:

1. Deactivate the driver modules. If the driver is still in use, the further script execution is aborted;
2. Uninstall the drivers;
3. Remove all binary and source files of the driver
4. Uninstall utilities (for Professional version only).

4 Using The Driver

After building and installing Paragon NTFS&HFS+ for Linux 9.6 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 users to 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 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 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 subsections);
     - Use ‘force’ mount option (see Dirty flag issues subsection).
  2. The disk is a GPT-partitioned disk — check GPT issues subsection for more information.
  3. Follow other steps on the Mount troubleshooting diagram to find the cause of the issue. Update/-fix Kernel source code (if necessary) and request new driver for the new Kernel from Paragon Software.

Analyze returned status and check output of (dmesg | tail). In case of failure, follow the Mount troubleshooting diagram to find possible causes and try to mount the partition again using the same or different mount options, if needed (see Mount options 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 and unmounting partitions

Mounting NTFS and HFS+ volumes

To mount volume using UFSD driver, use mount command with FS type set to ufsd, e.g.:

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

Sometimes volumes cannot be mounted using `'mount'` command, for example:

```
# mount -t ufsd /dev/sdd1 /mnt/sdd1
mount: wrong fs type, bad option, bad superblock on /dev/sdd1,
       missing codepage or helper program, or other error
```

```
       In some cases useful info is found in syslog - try
       dmesg | tail or so
```

In that case more information can be obtained by using the following command:

```
# dmesg | tail | grep ufsd
[ 369.844741] ufsd: volume is dirty and "force" flag is not set
```

Lines related to UFSD driver start with 'ufsd:' prefix. In case Kernel log is redirected to console, messages may be output to terminal window right after mount command is issued:

```
# mount -t ufsd /dev/sdd1 /mnt/sdd1
ufsd: volume is dirty and "force" flag is not set
mount: wrong fs type, bad option, bad superblock on /dev/sdd1,
       missing codepage or helper program, or other error
       In some cases useful info is found in syslog - try
       dmesg | tail or so
```

In this example, dmesg output contains information that the volume could not be mounted due to 'dirty' flag. See Dirty flag issues subsection for more information on 'dirty' flag and how to resolve the issue.

Dismount volumes
To dismount volumes mounted by UFSD driver issue, use umount command:

```
# umount /dev/sdd1
```

See Why the disk can't be dismounted? subsection for more information, if the volume can't be dismounted.

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 from 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). Run with `-a` command line option, the utilities check dirty flag state and in case it is set, they perform 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. 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)
```

4.5 Issues with large HDDs

Though our driver supports partitions larger than 2 TB (tested on 20 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).

5 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	+	+	`-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:
			• `nls=utf8`
			• `iocharset=utf8`

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.

Option	NTFS	HFS+	Expected behavior and examples
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.
uid	+	+	`-o uid=USERID` By default all existed files on a mounted NTFS/HFS+ volume are owned by root, while created files are owned by the user. 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. File's owner can be changed by the chown command, but this change will be effective until volume unmount.
gid	+	+	`-o gid=GROUPID` By default all existed files on a mounted NTFS/HFS+ volume are owned by group root, while created files are owned by the user's group. 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. File's group can be changed by the `chown` command, but this change will be effective until volume unmount.

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	+	+	`-o ro` To mount an NTFS/HFS+ volume in read-only mode.

Option	NTFS	HFS+	Expected behavior and examples
bestcompr	+	-	`-o bestcompr` Instructs the driver to use highest compression level when writing compressed files. High CPU-load.
sparse	+	-	`-o 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	+	+	`-o 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.
nohidden	+	-	`-o nohidden` Files with the Windows-specific `HIDDEN` attribute will not be shown under Linux.
sys_immutable	+	-	`-o sys_immutable` Files with the Windows-specific `SYSTEM` attribute will be marked as system immutable files. This option is not supported for symlink/fifo/socket files.
acl	+	+	`-o acl` Support POSIX ACLs (Access Control Lists). Effective if supported by Kernel. Not to be confused with NTFS ACLs. The option specified as acl enables support for POSIX ACLs. Supported if driver is built with support for ext2-like handling of file/directory permissions. Recommended for use with the HFS+ volumes.

Option	NTFS	HFS+	Expected behavior and examples
sync	+	+	`-o sync` All file system operations will be done synchronously. When used, UFSD driver performance may be decreased. This option is recommended for use, when disk can be detached without proper unmount, as it helps to minimize the possibility of the file system errors, when proper unmount for the volume is not performed.
discard	+	+	`-o discard` `discard` is the mount option in the UFSD driver, which is recommended for use with the solid-state drives (SSD) to enable support of the TRIM command for improved performance on delete operations..

6 Additional Utilities

Additional utilities for Paragon NTFS&HFS+ for Linux 9.6 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 9.6 were developed with Paragon UFSD SDK.

6.1 ufsd utility

ufsd utility is the single file system utility binary, which is included into the package since Paragon NTFS&HFS+ for Linux version 9.4.3.

Synopsis

```
ufsd [utility [arguments]...]
ufsd [--version]
```

E.g.: `'ufsd chkntfs -f /dev/sdb1'`

Utilities

- chkntfs - check and fix NTFS volume
- chkhfs - check and fix HFS+ volume
- chkufsd - check and fix NTFS/HFS+ volume (universal utility)
- mkntfs - format volume into NTFS
- mkhfs - format volume into HFS+

Description

`ufsd` is the multi-call binary that combines Paragon file system formatting and checking utilities into a single executable.

All utilities are actually hardlinks to the `ufsd` utility. In order to make sure that the files are not taking unnecessary space, it is advised to unpack provided archive on the target platform, or create hardlinks to the `ufsd` utility manually. You may also use symlinks to achieve the identical functionality or pass utility name as an argument to the `ufsd` binary, similar to how `busybox` binary operates.

6.2 chkufsd

chkufsd utility is the combined utility for checking and fixing all file systems, supported by Paragon NTFS&HFS+ for Linux 9.6.

Synopsis

```
chkufsd [option] device
```

E.g.: `'chkufsd -f /dev/sdb1'`

Options

`-fs:ntfs`	Treat target volume as NTFS
`-fs:hfs`	Treat target volume as HFS+
`-f`	Fix errors on the disk
`-a`	Perform checks only if 'dirty' flag is set
`-b:size`	Override default block (sector) size. Usage of default settings are strongly recommended. The `'-b'` parameter can use following values: 512, 1024, 2048, 4096.
`-m:size`	Set a limit for memory usage during file system check. Usage of this parameter may lower file system check performance. This option only limits amount of memory allocated for volume bitmap.
`-h`	Display this help
`--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

`chkufsd` is a combined utility that allows to check and fix file systems supported by the Paragon driver.

Note: Since version 9.4.3 this tool is actually a hardlink to the ufsd utility. In order to make sure that the files are not taking unnecessary space, it is advised to unpack provided archive on the target platform, or create hardlinks to the `'ufsd'` utility manually.

6.3 NTFS utilities

There are 2 basic utilities for NTFS file system:

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

6.3.1 mkntfs

`mkntfs` utility creates NTFS volumes (1.2, 3.0, 3.1 (Windows NT 4.0/2000/XP/2003/Vista/7/8.1/10) file system) on user-specified (block) device (disk partition) under Linux OS.

Synopsis

```
mkntfs [options] device
```

E.g.: 'mkntfs -f /dev/sdb1'

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.
-j:size	Set journal size for Paragon journal on NTFS volume. Supported sizes: from 1MB to 512MB. By default Paragon journal for NTFS is disabled.
-f	Force the format without confirmation.
-L	Use large FRS. The same as '-UseLargeFRS' or '/L' option on Windows. Equivalent of the '-m:4096' option in Paragon's mkntfs.
-s:start	Specify "hidden" sectors in the boot area.
-g:tracks:sectors tracks sectors	Specify disk geometry that should be written in the boot area. specifies number of tracks per disk side. 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. Generally 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
-h	Display this help
--help	Display this help
--bcfs	Sets System ID in boot sector to 'BCFS', making file system unrecognizable for non-Paragon drivers and utilities
--nodiscard	Do not discard volume before formatting
--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/7/8.1/10) file system on adevice (usually a disk partition).

Notes:

1. 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.

2. Since version 9.4.3 this tool is actually a hardlink to the ufsd utility . In order to make sure that the files are not taking unnecessary space, it is advised to unpack provided archive on the target platform, or create hardlinks to the ufsd utility manually.

6.3.2 chkntfs

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

Synopsis

```
chkntfs [options] device
```

E.g.: 'chkntfs -f /dev/sdb1'

Options

-f	Fix errors on the disk.
-a	Perform checks only if 'dirty' flag is set.
-b:size	Override default block (sector) size. Usage of default settings are strongly recommended. The '-b' parameter can use following values: 512, 1024, 2048, 4096.
-m:size	Set a limit for memory usage during file system check. Usage of this parameter may lower file system check performance.
-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).

Notes:

1. when `--no-orphans` option is used, real orphan files will be deleted. Without this option chkntfs restores real orphan files to found.XXX folders.
2. Since version 9.4.3 this tool is actually a hardlink to the ufsd utility. In order to make sure that the files are not taking unnecessary space, it is advised to unpack provided archive on the target platform, or create hardlinks to the ufsd utility manually.

6.4 HFS+ utilities

There are 2 basic utilities for HFS+ file system:

- mkhfs — format any partition as HFS+ under Linux
- chkhfs — check HFS+ partition for integrity and (optionally) fix errors

6.4.1 mkhfs

`mkhfs` utility creates an HFS+ volume on specified (block) device (disk partition) under Linux OS.

Synopsis

```
mkhfs [options] device
```

E.g.: `'mkhfs -j /dev/sdb1'`

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.
`-na:size`	Specify attributes b-tree node size: 4K-32K
`-f`	Force the format without confirmation.
`-j`	Make volume journalized with default journal size
`-j:size`	Make volume journalized, manually setting size of the journal

-c	Make volume case-sensitive.
-h	Display this help
--help	Display this help.
--nodiscard	Do not discard volume before formatting
--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).

Note: Since version 9.4.3 this tool is actually a hardlink to the ufsd utility. In order to make sure that the files are not taking unnecessary space, it is advised to unpack provided archive on the target platform, or create hardlinks to the ufsd utility manually.

6.4.2 chkhfs

chkhfs utility provides consistency checking of a HFS+ volume and fix errors.

Synopsis

```
chkhfs [options] device
```

E.g.: 'chkhfs -f /dev/sdb1'

Options

-f	Fix errors on the disk.
-a	Perform checks only if 'dirty' flag is set.
-b:size	Override default block (sector) size. Usage of default settings are strongly recommended. The '-b' parameter can use following values: 512, 1024, 2048, 4096.
-m:size	Set a limit for memory usage during file system check. Usage of this parameter may lower file system check performance.
-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

`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).

Note: Since version 9.4.3 this tool is actually a hardlink to the ufsd utility. In order to make sure that the files are not taking unnecessary space, it is advised to unpack provided archive on the target platform, or create hardlinks to the `ufsd` utility manually.

7 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 binary module installation, testing and troubleshooting as well as to section on Hardware requirements and Software requirements. Please also review Mount troubleshooting and Using The Driver 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+ for Linux 9.6 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/HFS+ file systems or to Paragon's driver itself, e.g. IOCTLs, additional utilities and so on).

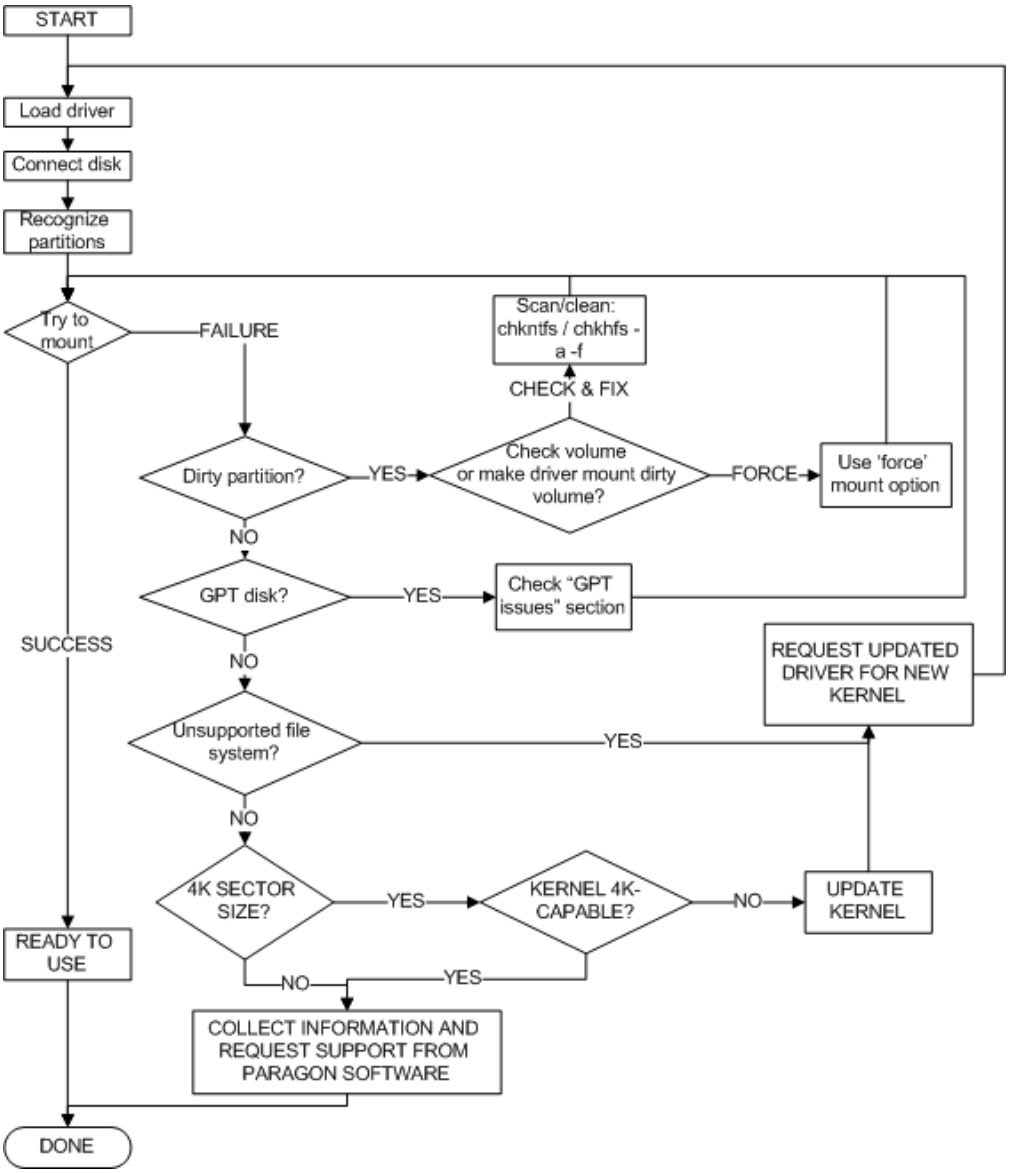
Step 3. Prepare to report the issue
After performing previous steps and making sure that the issue is related to Paragon NTFS&HFS+ for Linux 9.6 driver, prepare to report the issue to Paragon.

COLLECT ALL INFORMATION ON THE ISSUE
The most important point in issue resolution process is quickly obtaining all the information related to the issue. A quick collection of required information is the key to resolving an issue faster.

Step 4. Assist Paragon engineers to resolve the issue quickly
Try to provide as detailed information on the issue, as possible.

7.2 Mount troubleshooting

Use our mount troubleshooting diagram for faster mount issue resolution.



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

1. Read Software 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, 4.4.9). 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

```
ls -s /usr/src/linux-$(uname-) /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

```
ls -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,

```
ls -s /lib/modules/$(uname-r)/build /usr/src/linux
```

If you still have the same problem i.e. `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 Software 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:

```
ls -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 `Makefile` of the kernel (you can find `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 `4.4.9-300.fc23.x86_64` then the following lines must be found at the beginning of the Makefile:

```
VERSION = 4
PATCHLEVEL = 4
SUBLEVEL = 9
EXTRAVERSION = -300.fc23.x86_64
```

7.6 ufsd module: kernel-module version mismatch

That means kernel version mismatch.

1. Check kernel source version in `/usr/include/linux/version.h`
2. Check the currently running kernel version: `uname -r`
3. Both versions must match.

4. If they don't match, please restore Kernel configuration or recompile kernel anew (advanced).

7.7 ufsd module: create_module: operation is not permitted

That means you must have root privilege to load the driver.

7.8 insmod: a module named as ufsd already exists

1. Make sure that you use the same version of GCC compiler that was used for kernel compilation.
2. Make sure that `Makefile` of the kernel (you can find `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 `4.4.9-300.fc23.x86_64` then the following lines must be found at the beginning of the Makefile:

```
VERSION = 4
PATCHLEVEL = 4
SUBLEVEL = 9
EXTRAVERSION = -300.fc23.x86_64
```

7.9 insmod: Unknown symbol jnl_op (err0)

That means one is trying to load `ufsd.ko` module into the Linux Kernel before `jnl.ko` module has been loaded:

```
# insmod ufsd.ko
insmod: ERROR: could not insert module ufsd.ko: Unknown symbol in module
# dmesg | tail
[84199.673358] ufsd: Unknown symbol jnl_op (err 0)
```

Load `jnl.ko` module first and `ufsd.ko` afterwards:

```
# insmod jnl.ko
# insmod ufsd.ko
# lsmod | grep ufsd
ufsd 614400 0
jnl 36864 1 ufsd
```

7.10 Can't mount NTFS/HFS+ volume

1. Make sure that the driver is activated (loaded into the Kernel) via following command:

```
lsmod | grep ufsd
```

2. Make sure that the driver supports file system mounted partition is formatted with checking its version:

```
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.

7.11 Hardware issues

Sometimes storage issues are caused by a faulty hardware. Hardware issues can be classified by their place of origin:

1. storage device (HDD, SSD, USB flash drive, SD card, etc.)
2. target platform (hardware board which utilizes the UFSD driver)
3. data medium (e.g. USB cable)

Often this kind of issues can be identified based on the system log output (e.g. dmesg). The basic principle is to look for an error messages from the underlying subsystem on top of which the UFSD driver operates (e.g. USB controller). Some examples:

» `usb *-*: USB disconnect, device number ***`
(in case the USB device was not actually disconnected)
» `end_request: I/O error, dev sdX, sector ***`
» `Buffer I/O error on device sdX, logical block ***`
» `mmcblk*: error -110 transferring data, sector ***`
» `sd *:*:*:*: [sdX] Unhandled error code`
» `sd *:*:*:*: [sdX] Spinning up disk........not responding...`
» `sd *:*:*:*: [sdX] Device not ready`
» `xhci_hcd *:*:*: WARN: Stalled endpoint`

Identification of the faulty hardware cannot be done solely on the UFSD driver messages because the driver operates on a higher level than HW drivers. Still UFSD messages can be helpful for obtaining a general picture of the issue. Here are some typical messages in case of hardware issues:

» `ufsd: failed to read block 0xNNNN`
» `ufsd: failed to map block 0xNNNN`
» `ufsd: failed to write block 0xNNNN`
» `ufsd: bio read I/O error`

A hardware problem cannot be solved at the file system driver level. Thereby that kind of issues is out of the Paragon's scope of responsibility. The best solution is to contact the faulty hardware's manufacturer technical support. One should contact us only if the hardware fault is followed by a system hang or kernel panic caused by the UFSD driver.

8 Sysdump utility

8.1 Introduction

In case Paragon team requires additional information on the test storage device and/or test platform itself for the issue troubleshooting with specific NTFS/HFS+ volume(s), the sysdump utility can be used to capture very compact images of volumes with file system inconsistencies. Additionally it could be used to collect test platform system information: storage device summary, Kernel version, loaded UFSD driver version and so on.

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

8.2 Main functions

Main functions of the sysdump utility are:

- Capturing metadata image of the test volume
- Collecting HW sample system information.

Note that sysdump utility requires root privileges as they are needed for working with storage devices on Linux platform.

When utility is run without additional options basic help is displayed

```
# ./sysdump
Dump NTFS/HFS/Ext volumes (metafiles only) and collect system information
    ↪.
Paragon's Sysdump utility doesn't collect any personal or user-sensitive
    ↪information.
Please refer to the 'Privacy policy' described in the Sysdump utility
    ↪guide for more information.
Usage: sysdump [-s] [device] [-o <filename>]
-s      collect system information
-o      new file name (if name without suffix ".tar", then suffix ".tar"
    ↪will be added automatically)
E.g. sysdump /dev/hdb1
```

8.3 Using the sysdump utility to collect both platform system information and metadata

1. Extract sysdump utility to a folder with read/write access.
2. In terminal change to the folder with the extracted sysdump utility. Use the following command to capture the metadata image and to collect platform system information:

   ```
   # ./sysdump -s /path/to/partition
   ```

3. This will create an archive named, based on platform parameters `<Architecture>.<Kernel version>.<date>.tar` with storage device metadata image, its md5 sum and sample system information:

```
# ./sysdump -s /dev/sdc1
Scanning NTFS...
Added 13 files/dirs from $Extend                                    .
Recognized as NTFS.
Dumping "/dev/sdc1" (465.76 GB) ...
Dump finished. File size 1248334848 bytes (including tail)
Collecting system information...
System information was saved to /home/UFSD_utilites/i686
   ↪.3.13.0.20141105.tar
```

8.4 Changing the name of output archive

Name of the output archive can be changed by using the additional '-o' parameter and adding new file name. For example:

```
# ./sysdump -s /dev/sdc1 -o test.tar
Scanning NTFS...
Added 13 files/dirs from $Extend                                    .
Recognized as NTFS.
Dumping "/dev/sdc1" (465.76 GB) ...
Dump finished. File size 1248334848 bytes (including tail)
Collecting system information...
System information was saved to /home/UFSD_utilites/test.tar
```

8.5 Output file description

The Sysdump utility creates a single tar archive in the working folder with its' work results for easier data transfer to Paragon team. This archive can be easily uncompressed with the tar utility, e.g.:

```
# tar -xf ./x86_64.3.11.9.20131211.tar
```

There are several files inside archive:

- dmesg - contents of the 'dmesg' command output
- dumpbin.gz - compressed metadata image of the test volume
- dumpbin.md5 -md5 checksum for the collected image itself, not the archive file
- modules - list of all Kernel modules (*.ko files) found on the platform
- sysinfo - file with platform system information: list of storage devices, Linux Kernel version, version of the loaded UFSD driver, CPU, memory information, etc.

8.6 Delivering collected data to Paragon

Please send the tar file, generated by the 'Sysdump' utility to your support contact at Paragon Software via e-mail, via 'My account' page at Paragon Software Support Portal.

8.7 Privacy policy

Paragon's Sysdump utility doesn't collect any personal or user-sensitive information. Platform data is obtained via usage of the native Linux utilities and standard system files (e.g. fdisk, parted,

/proc/cpuinfo, /proc/meminfo and others). The resulted volume metadata image includes only file system metadata and risk of leaking sensitive data when using sysdump utility is minimized. Nevertheless, collected information is also covered by the mutual Non-disclosure agreement and couldn't be forwarded to any third party.

9 UFSD driver compatibility

This section describes file system features supported by Paragon NTFS&HFS+ for Linux 9.6 driver, respectively.

9.1 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.

9.2 HFS+ features

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.

10 Frequently Asked Questions

10.1 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-LiveKDE"
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).
```

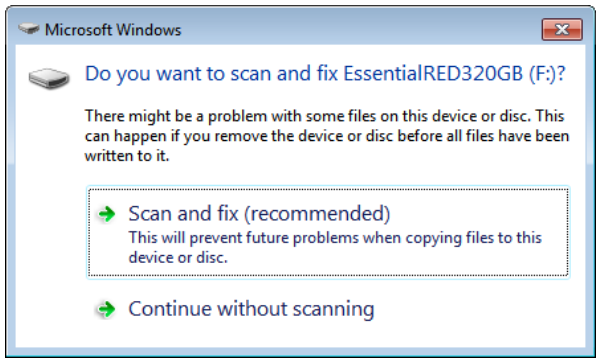
10.2 Warnings on Windows7/Vista when NTFS HDD is reconnected from Linux

After NTFS volume previously operated by Paragon NTFS&HFS+ for Linux 9.6 driver is attached to Windows Vista/Windows 7 machine, warnings are displayed on the screen. Why?

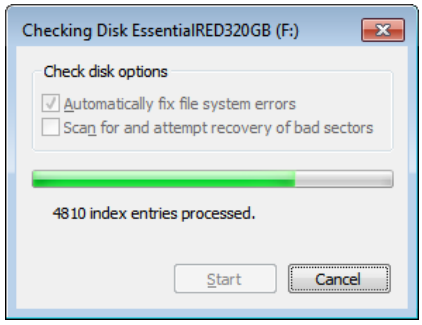
This is the case when volume was not unmounted correctly before it was detached from Linux system.

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.

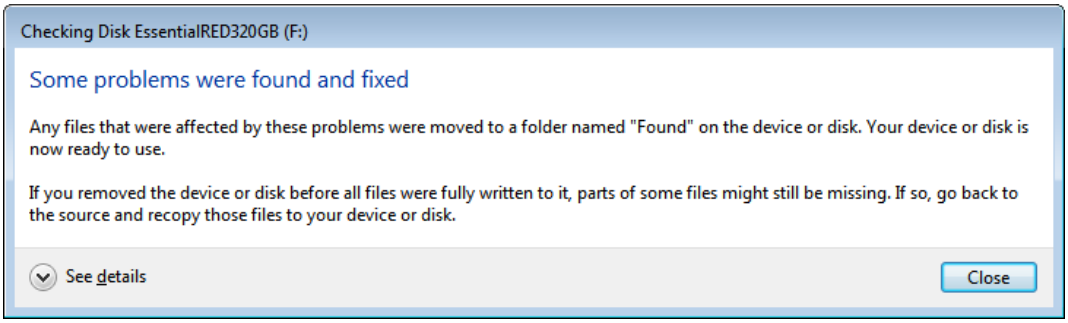
An USB HDD enclosure with 320 GB SATA HDD with one NTFS partition was detached from system while file copy operation was in progress. After the enclosure was attached to Windows 7 PC again, the following dialog was displayed:



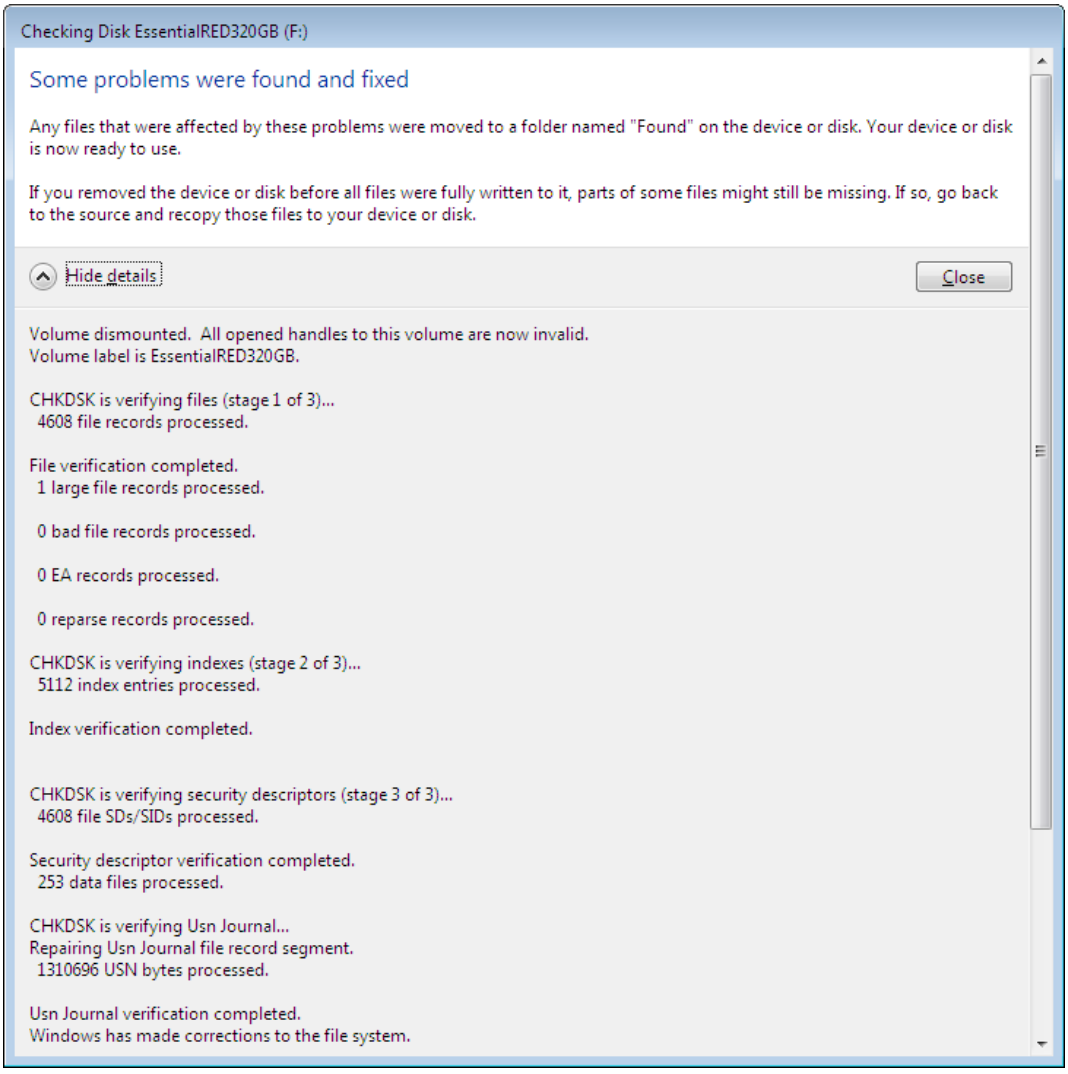
After user clicks 'Scan and fix (recommended)', scan process begins:



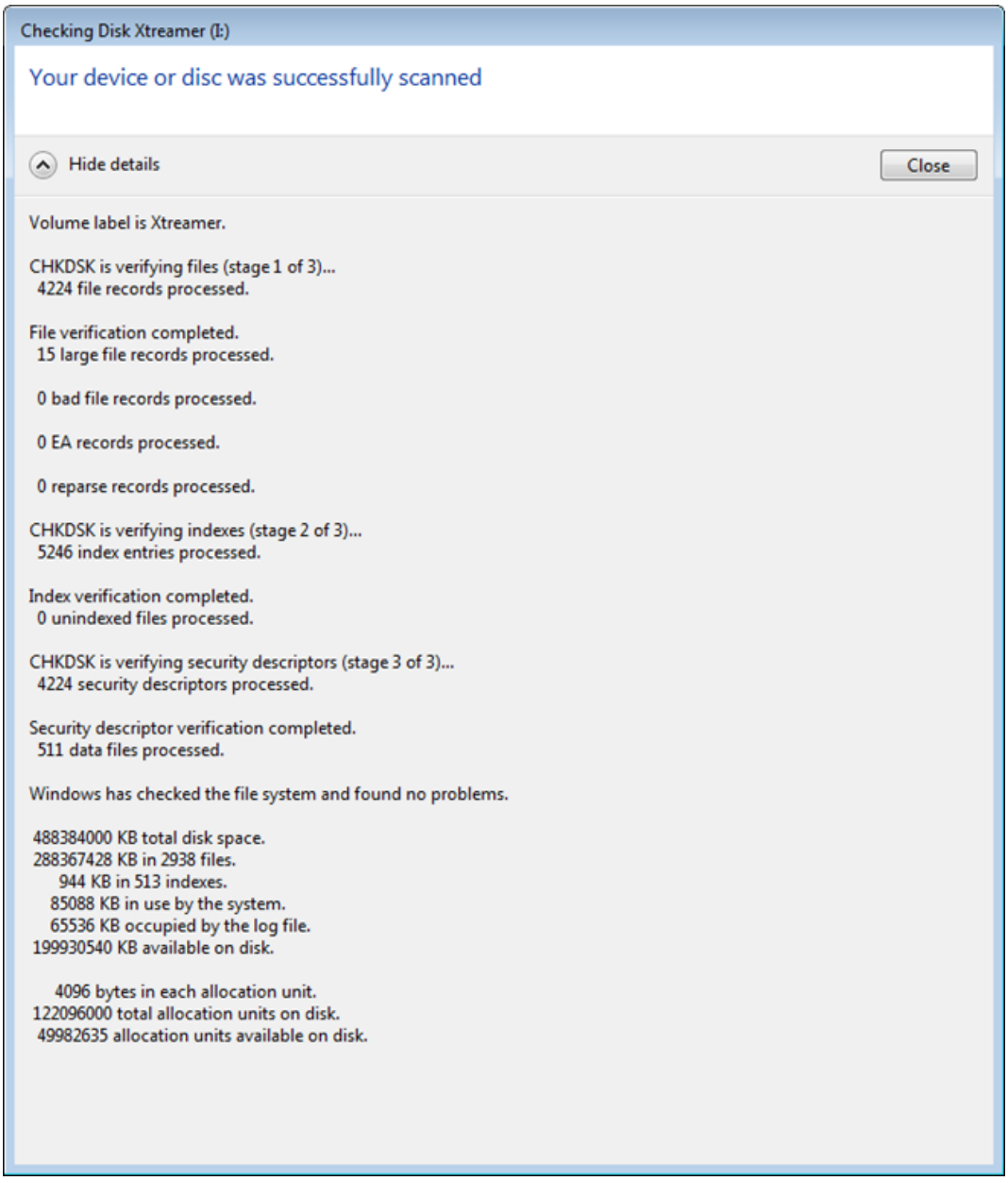
After checking is completed, the following summary window is displayed:



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:



10.3 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 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.

10.4 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.

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
```

10.5 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 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.

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

Let us 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.

10.7 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.

Note: 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:

```
# sync
```

It is recommended also to add `'sync'` command before `'umount'` to the unmounting web-interface script, as it helps to avoid file system corruption on the storage medium, if unmount script was unsuccessful and the drive was detached manually.