PARAGON Technologie GmbH
Heinrich-von-Stephan-Str.5c • 79100 Freiburg/Germany
Tel. +49-761-59018-202 • Fax +49-761-59018-130
Website: www.paragon-software.com • E-mail: sales@paragon.software.com

Paragon ReFS for Linux

Getting started guide

Contents

1 Introduction

The goal of this guide is to help users quickly find out how to use the **urefs** driver. It describes a general approach to mounting partitions using the Paragon ReFS for Linux driver and helps to avoid common issues. We strongly recommend reading this document before using our drivers.

Paragon ReFS for Linux has the following main features:

- Transparent read-write access to ReFS (1.x) volumes;
- Easy installation and uninstallation (assistant scripts);
- Support for SMP kernels;
- Support for Kernel versions from 2.6.36 up to 4.10.x;
- ReFS volumes mounted by the '**urefs**' driver could be easily shared over network (e.g. via SAMBA/FTP/... protocols);
- Paragon ReFS for Linux can be installed in parallel with the Paragon NTFS&HFS+ for Linux driver on the same platform.

Paragon ReFS for Linux specific limitations:

- Only x86/x86-64 platforms are supported;
- Only systems with THREAD_SIZE set to 16KB or larger are supported;
- Only ReFS (1.x) volumes from Windows Server 2012/Windows 8.1/Windows 10 are supported;
- ReFS (3.x) volumes from Windows Server 2016 can't be mounted;
- ReFS file system utilities are not provided.

2 System requirements

2.1 Hardware requirements

Minimum hardware requirements:

- CPU: Intel Pentium 300 MHz and higher, or compatible;
- Only x86/x86-64 CPUs are supported;
- 32MB of RAM.

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 4.10.x.

Linux distributions the product was tested with:

- Ubuntu 16.04 LTS;
- Debian 8.7;
- Fedora 25;
- CenOS 7;
- OpenSuse Leap 42.2.

Development Environment.

A development environment is required to compile Linux driver. Please verify these tools are all functional. The easiest way is to choose the developer tool kit when installing Linux.

What has to be installed:

- Kernel source code (recommended) or Kernel header files;
  `#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`

Development environment limitations:

- GNU C compiler (gcc) must have the version 4.9 or higher;
- The user should login as root to install the driver;
- Correct operation is not guaranteed for customized Linux kernels.

3 Paragon ReFS for Linux driver installation

Paragon ReFS for Linux driver package includes the following components:

- The source files for the Paragon ReFS for Linux driver;
- Assistant script files for easy installation and uninstallation.

3.1 ReFS driver installation

To start using the 'Paragon ReFS for Linux' on the target platform it need to be built and installed first. Steps to install the **urefs** driver are as follows:

1. Log in as root. This step is obligatory;
2. Build and install the '**urefs**' driver using the '`install.sh`' script. Alternatively, driver binary module may be built manually using '`configure`' and '`make driver`' commands.
3. Activating (loading) the driver.

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

The installation procedure is described in more details below.

Unpacking Setup Files

Setup files of the Linux-based version of the 'Paragon ReFS for Linux' are provided in the form of a gzip archive. The archive should be copied to the hard disk and decompressed.
For example:

- Create separate folder (e.g. `mkdir /usr/tmp/urefs`);
- Change the current directory to the new one (e.g. `cd /usr/tmp/urefs`);
- Use tar utility to unpack initial archive (e.g. `tar -xf /path/to/the/archive/urefs_*.tar.gz`).

Using the INSTALL.SH assistant Script

The assistant script '`install.sh`' provides the extremely easy and flexible way to make the '**urefs**' driver and install driver module in the system.

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.

To start, simply run the '`install.sh`' script with root privileges e.g.:

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

The assistant script will automatically perform following actions:

- Detect the Linux Kernel version;
- Find kernel header files and libraries needed for building the drivers;
- Build the driver and binary module;
- Install the driver.

3.2 ReFS driver uninstallation

To completely remove the '**urefs**' driver from the current Kernel, one should dismount all ReFS volumes, uninstall the driver and optionally remove the binary files. 'Paragon ReFS for Linux' provides tools for the drivers uninstall automation.

The assistant script '`uninstall.sh`' allows to completely remove the driver from the current system Kernel, correctly uninstalling it from the computer.

Uninstalling the driver:

Unmount all currently mounted ReFS partitions and then run the '`uninstall.sh`' script:

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

The assistant script will automatically perform the following actions:

- Deactivate driver module;
- Uninstall the driver;
- Remove all binary and source files of the driver.

4 Paragon ReFS for Linux driver usage

4.1 Urefs driver version information

Version information of the '**urefs**' driver can be received using the following command:

```
cat /proc/fs/urefs/version
```

```
ReFs 1.x support included
Built_for__Retail_Package_ufsd_f_107660_refs_rw_r305723_b1226
$Id: ufsdvfs.c 305692 2017-04-07 14:43:16Z fedorenchik $
driver (UFSD_HEAD ufsd_f_107660_refs_rw_r305723_b1226, acl, ioctl, sd2(5)) loaded at ffffffffa0581000,
Kernel .config hash: original 0xbad3d38, current can't check.
```

4.2 Mounting volumes

ReFS partition(s) should be mounted before testing driver operation on the corresponding system by executing the following command:

```
# mount -t urefs [-o options] device mount_point
```

Please execute the '`mount | grep urefs`' command to make sure the device is mounted using the **urefs** driver.

```
$ sudo mount -t urefs /dev/sdf1 /mnt/refs
$ mount | grep refs
/dev/sdf1 on /mnt/refs type urefs (rw,relatime,nls=utf8)
```

4.3 Mount options

Example on mount options usage:

```
# mount -t urefs -o nls=utf8,fmask=000,dmask=000,umask=000 /device /mount_point
```

Below you can see the list of available options.

Mount option	Description
nls=<code_page>	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. E.g. `nls=utf8`
noatime	All files and directories will not update their last access time attribute if a ReFS partition is mounted with this parameter.
nocase	If this option is used all file and directory operations (open, find, rename...) are case insensitive. Casing is preserved in the names of existing files and directories. **This mount option is enabled by default.**
ro	Mounts ReFS volume in read-only mode.
umask=<value>	Changes the permissions for new created files and directories.
dmask=<value>	Changes the permissions for directories that already exist on a mounted volume. If 'dmask' is not defined, it uses the same values, as 'umask'

Mount option	Description
fmask=<value>	Changes the permissions for files that already exist on a mounted volume. If 'fmask' is not defined, it uses the same values, as 'umask'. To mount Samba, FTP or NFS shares the combination of `umask=000,fmask=000,dmask=000` is usually specified.
gid=<groupid>	By specifying the 'gid' parameter you can set an owner group of the files. The groupid can be any name from /etc/group, or any number representing a group id. By default all existed files on a mounted ReFS volume are owned by group root, while created files are owned by the user's group.
uid=<userid>	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. By default all existed files on a mounted ReFS volume are owned by root, while created files are owned by the user.

4.4 Driver usage after Linux Kernel update

After each Linux Kernel update on target platform, the '`urefs.ko`' driver module needs to be recompiled anew. Please refer to the ReFS driver installation subsection for more information.

5 Troubleshooting

1. **Consult Documentation**
   Please consult documentation to make sure that the encountered behaviour is not by design, with special attention given to the sections related to the Troubleshooting.

2. **Make sure the issue is not related to Linux itself**
   Make sure that the root cause of the issue is not related to Linux itself. For example, if an issue is discovered while performing a certain file system-related operation on a volume mounted with Paragon's '**urefs**' driver, please verify that the same issue is not observed when the same operation is performed on a 'native' file system like Ext2fs, Ext3fs or FAT (except, of course, for operations specific to ReFS file systems or to Paragon's driver itself).

3. **Prepare to report the issue**
   After performing previous steps and making sure that the issue is related to Paragon's UFSD driver, prepare to report the issue to Paragon, based on the issue type:

   - Urefs driver compilation issue;
   - Loading urefs driver module issue;
   - Urefs driver mount volume issue;

4. **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 assist Paragon engineers, try to provide as detailed information on the issue, as possible.

5.1 Urefs driver compilation issue

When Linux system doesn't have `THREAD_SIZE` set to 16 KB or larger one will receive the following error message during the `urefs.ko` module compilation:

```
#error "Refs requires 16K+ stack"
```

In this case one will need to update Linux system anew so the `THREAD_SIZE` is set to 16KB or larger as it is a mandatory requirement for the 'Paragon ReFS for Linux'.

In other cases please verify that the '`make clean`' command was used before manually rebuilding the '**urefs**' driver from the package. After you have tried to recompile driver module, please collect and send us the following information:

- Linux Kernel version;
- Console output of compilation process;
- The '`paragon-urefs-install.log`' file generated by the '`install.sh`' script
- The '`Config.log`' file generated during compilation process.

5.2 Loading urefs driver module issue

Verify that another **urefs** driver module is not already loaded into the platform. Use the package to rebuild driver module on your side. If the issue still exists, please collect and send the following information to Paragon:

- Linux Kernel version;
- Console output of the '**urefs**' driver loading;
- Dmesg output after `urefs.ko` module wasn't loaded;
- `urefs.ko` module used on your side.

5.3 Urefs driver mount volume issue

Please verify that the test volume is formatted into the ReFS file system supported by the current Paragon ReFS for Linux driver release (please refer to the Introduction for more information on Paragon ReFS for Linux driver limitations).

If the issue still exists, please collect and send the following information to Paragon:

- Linux Kernel version;
- Storage device information: device type, partitioning type, number of partitions, file system for the test volume, size of the test volume, etc;
- Mount command used on your side;
- dmesg output after '**urefs**' was used to mount test volume;
- Metadata image of the test volume collected with the `Sysdump` utility (please ask Paragon support team to provide it).