Path Tool Long

André Burgaud

2016-11-13

Over the last couple of years, I faced a Windows issue preventing me to delete directories or files with paths exceeding a certain size. Using the default Windows tools such as `DEL`, `RMDIR`, or `Windows Explorer`, resulted in errors `The directory name ... is too long` or `The source file name(s) are larger than is supported by the file system`.

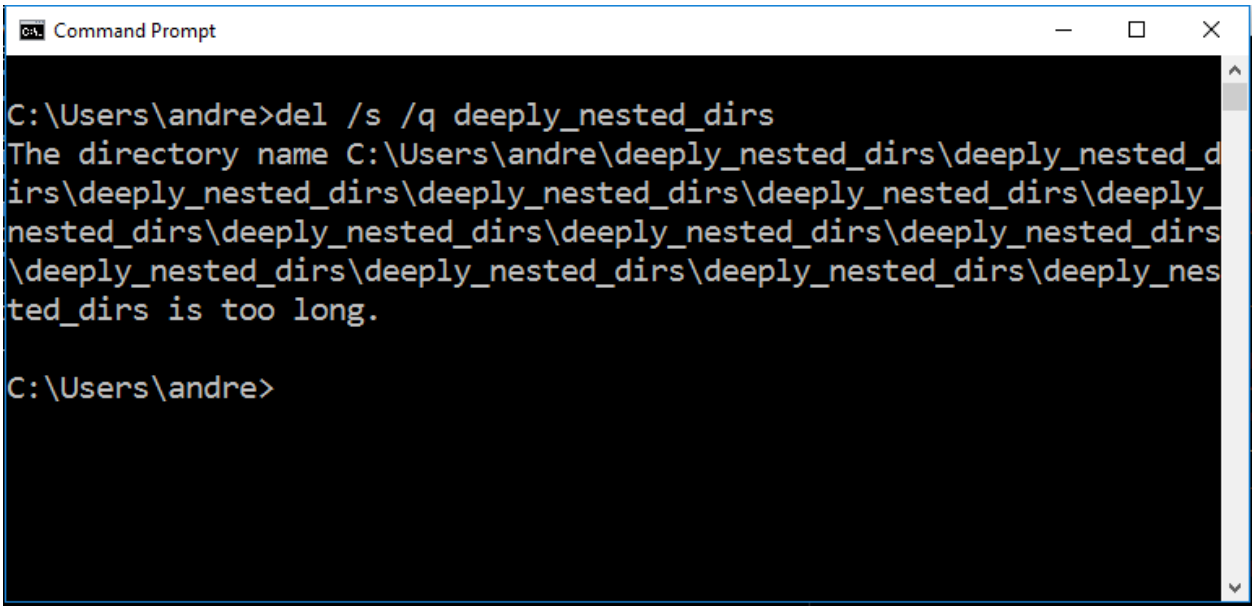

Figure 1: Path is too long

After some research, I noticed that a lot of people also encountered this problem. Various workarounds and tools are already available, but they did not work for me due to the depth of nested directories created on my Windows system. In this article I describe my findings and propose some solutions, including the convenient Winrmrf that I implemented.

Maximum Path Length Limitation

The issue described above is related to the Windows Maximum Path Length Limitation. As documented in the Microsoft documentation, the Windows API has a limitation of 260 characters for file and directory paths (`MAX_PATH`). By default, any time a path exceeds this limit, the Windows API functions *choke*. This directly impacts the common tools we interface with on Windows, either in a command prompt (`DEL`, `RMDIR`) or via graphical user interface applications like `Windows Explorer`.

Nevertheless, the Windows API has functions with Unicode versions. Those particular functions, such as `RemoveDirectory()`, are easy to identify as they expose the Unicode enabled version with an appended `W` (e.g. RemoveDirectoryW() for `RemoveDirectory`), whereas the ANSI version is suffixed with an `A` (e.g. RemoveDirectoryA(). The Unicode versions allow an extended path to a maximum of 32,767 ($2^{15} - 1$) characters.

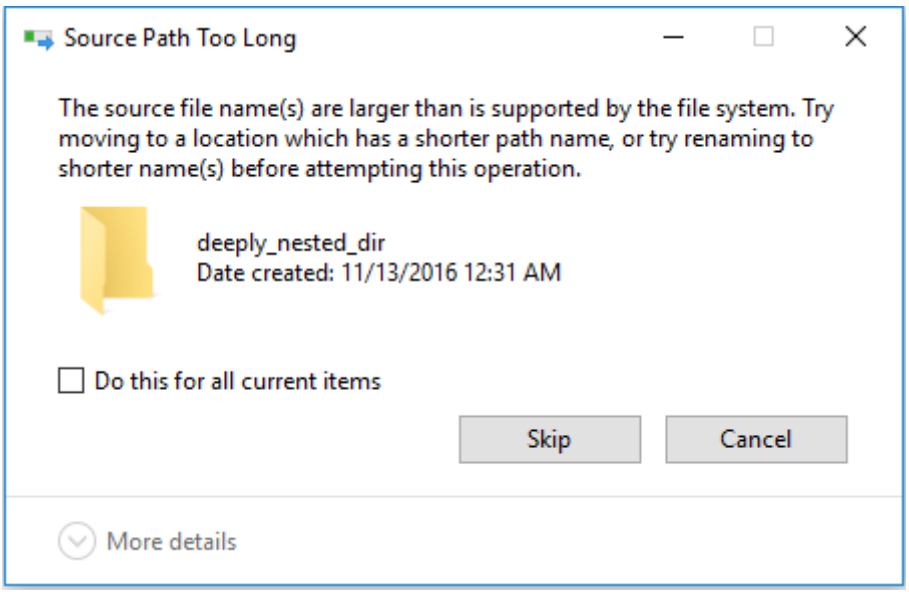

Figure 2: Source Path is too long

To specify an extended path, you need to use absolute paths and prefix them with `\\?\`. For example the extended path for a directory named `deeply_nested_dirs` in folder `C:\Users\andre` is:

```
\\?\C:\Users\andre\deeply_nested_dirs
```

Windows 10

Since Windows 10 version 1607 (Windows 10 Anniversary Update), the `MAX_PATH` limitations have been somehow removed, but it requires to modify a key in the registry. The following registry script updates the appropriate key:

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem]
"LongPathsEnabled"=dword:00000001
```

Note: This does not solve the problem of being able to delete long path files or directories using the default Windows tools like `DEL`, `RMDIR` or `Windows Explorer`. But, changing the value of `LongPathsEnabled` from `0` to `1`, enables the usage of the Windows API functions without having to prefix paths with `\\?\`.

Programmatic Solutions

With a little help from Python

On Windows, Python is using the Windows API functions. Therefore, armed with the knowledge of the `\\?\` prefix as documented in Maximum Path Length Limitation, it might be possible to overcome the `MAX_PATH` limit. Indeed, after some simple tests, I realized that `shutil.rmtree()` worked perfectly, provided that an absolute path prefixed with `\\?\` was passed as argument. Below is an example of a complete Python script building an extended path, and invoking `shutil.rmtree()` to successfully delete a directory with a path exceeding 260 characters:

```
import os, shutil, sys

def usage():
```

```
    print("Usage: python %s <dir_to_delete>" % sys.argv[0])

def remove_dir(path):
    abs_path = os.path.abspath(path)
    ext_path = r"\\?\%s" % abs_path
    try:
        shutil.rmtree(ext_path)
    except FileNotFoundError:
        print("The system cannot find the path specified: '%s'" % path)
        sys.exit(2)

if __name__ == '__main__':
    try:
        path = sys.argv[1]
    except IndexError:
        usage()
        sys.exit(1)
    else:
        remove_dir(path)
```

The following shows how to delete the directory `deeply_nested_dirs` with `rmrf.py`:

```
C:\>python rmrf.py deeply_nested_dirs
```

In order to test the different solutions explored to prepare this article, I also used the following script to create deeply nested directories resulting in paths longer than `MAX_PATH`:

```
import os, sys

def usage():
    print("Usage: python %s <root_dir> <iteration>" % sys.argv[0])
    print("Example: python %s some_dir 100" % sys.argv[0])

def makedirs(path, iterations):
    abs_path = os.path.abspath(path)
    dir_name = path.split("\\")[-1] # Last name if absolute path
    ext_path = r"\\?\%s" % abs_path
    for _ in range(iterations):
        ext_path += r"\%s" % dir_name
    try:
        os.makedirs(ext_path)
    except FileExistsError:
        print('Directory already exists')

if __name__ == '__main__':
    try:
        path, iterations = sys.argv[1:3]
    except(IndexError, ValueError):
        usage()
        sys.exit(1)
    else:
        makedirs(path, int(iterations))
```

To create deeply nested directories (100 levels deep), execute the following:

```
C:\>python makedirs.py deeply_nested_dirs 100
```

Note: The Python code examples were tested with Python 3.5.2 on Windows 10.

For Something 'Groovy'

If you are more leaning towards the Java ecosystem than Python, you may consider the following to delete files and directories the default Windows tools can't.

These examples are using Groovy, but any language targeting the JVM (Java Virtual Machine) would work.

Apache Commons IO Using [FileUtils.deleteQuitely] from the Apache Commons IO library and Groovy, you can delete directories (in this example `deeply_nested_dirs`) with the following script:

```
import org.apache.commons.io.FileUtils

FileUtils.deleteQuietly(new File('deeply_nested_dirs'))
```

Save the code above in a file named `rmrf.groovy` and execute with:

```
C:\>groovy rmrf.groovy
```

Command Line Script The Groovy `-e` option does not require any dependency on external libraries, and is directly executed at the command line:

```
C:\>groovy -e "new File('deeply_nested_dirs').deleteDir()"
```

Note: The Groovy functions don't require the *magic* prefix `\\?\` to prepend the path.

Winrmrf

If you don't want to program your own solutions, but want something small and easy to use in order to delete *stubborn* directories, Winrmrf might be worth a try. **Winrmrf**, implemented in Nim, results in a small standalone executable that you can copy in a directory included in your Windows `PATH`. It does not have any other dependency like a Python or Groovy solutions would have (respectively Python runtime and JVM with Groovy library). It can be invoked at the command line:

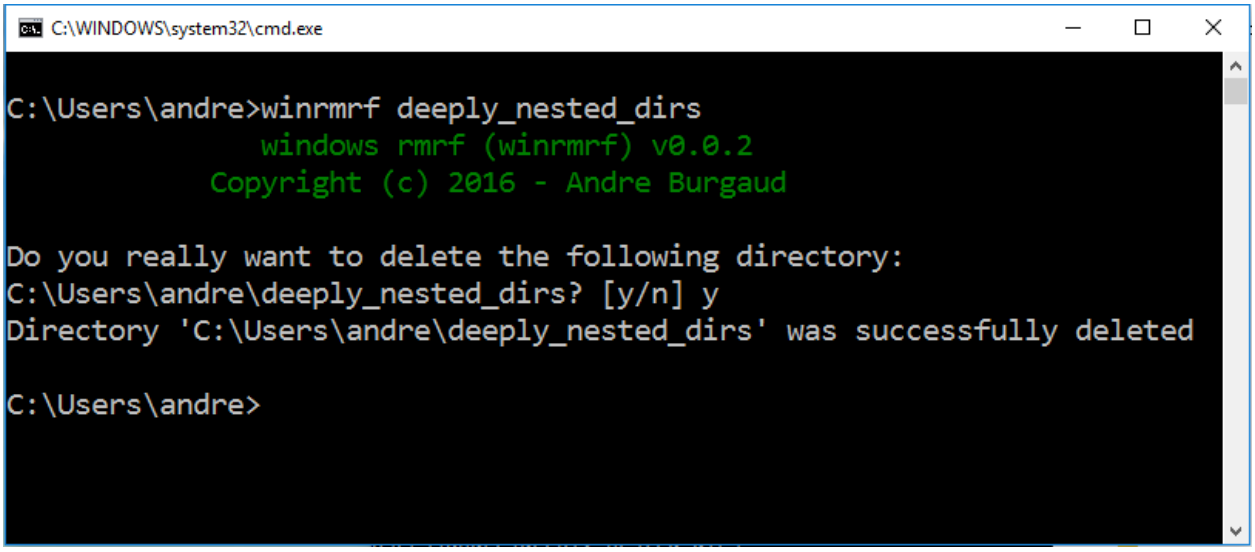

Figure 3: Winrmrf

Note: The source code of Winrmrf is available on Github. Binary versions are also available for download in the GitHub project releases. **Winrmrf** is released under the MIT License.

Other Solutions and Workarounds

Many creative workarounds and solutions are also available online. Although these solutions were not fully satisfying for my situation, some are worth checking out:

- How to overcome long file path issues with robocopy?
- How to delete directories with path/names too long for normal delete
- rimraf: A deep deletion module for node

Resources

- Naming Files, Paths, and Namespaces
- Maximum Path Length Limitation
- Python
- Groovy
- Nim
- Winrmrf
- Winrmrf binaries

Legal