Commit

✕ [3.10] gh-123270: Replaced SanitizedNames with a more surgical fix. (G...

Browse files

...H-123354) (#123426)

Applies changes from zipp 3.20.1 and jaraco/zippGH-124
(cherry picked from commit 2231286)
(cherry picked from commit 17b77bb)

Co-authored-by: Jason R. Coombs <jaraco@jaraco.com>

3.10 (#123426)

jaraco committed 19 hours ago Verified 1 parent d3f39ce commit 0aa1ee2

Showing **3 changed files** with **81 additions** and **67 deletions**.

Whitesp... Ignore whitesp... S... Uni...

Filter changed files

- Lib
  - test
    - test_zipfile.py
    - zipfile.py
  - Misc/NEWS.d/next/Library
    - 2024-08-26-13-45-2...

76 Lib/test/test_zipfile.py

```
5    5    import itertools
6    6    import os
7    7    import pathlib
     8  + import platform
8    9    import posixpath
9    10   import string
10   11   import struct
3282 3283
3283 3284        def test_malformed_paths(self):
3284 3285            """
3285      -          Path should handle malformed paths.
     3286  +          Path should handle malformed paths
               gracefully.
     3287  +
     3288  +          Paths with leading slashes are not
               visible.
     3289  +
     3290  +          Paths with dots are treated like
               regular files.
3286 3291            """
3287 3292            data = io.BytesIO()
3288 3293            zf = zipfile.ZipFile(data, "w")
3291 3296            zf.writestr("../parent.txt",
               b"content")
3292 3297            zf.filename = ''
3293 3298            root = zipfile.Path(zf)
```

```
-         assert list(map(str, root.iterdir())) == [
-             'one-slash.txt',
-             'two-slash.txt',
-             'parent.txt',
-         ]
+         assert list(map(str, root.iterdir())) == ['../']
+         assert root.joinpath('..').joinpath('parent.txt').read_bytes() == b'content'
+
+     @unittest.skipIf(platform.system() == "Windows", "GH-123693")
+     def test_unsupported_names(self):
+         """
+         Path segments with special characters are readable.
+
+         On some platforms or file systems, characters like
+         ``:`` and ``?`` are not allowed, but they are valid
+         in the zip file.
+         """
+         data = io.BytesIO()
+         zf = zipfile.ZipFile(data, "w")
+         zf.writestr("path?", b"content")
+         zf.writestr("V: NMS.flac", b"fLaC...")
+         zf.filename = ''
+         root = zipfile.Path(zf)
+         contents = root.iterdir()
+         assert next(contents).name == 'path?'
+         item = next(contents)
+         assert item.name == 'V: NMS.flac', item.name
+         assert root.joinpath('V: NMS.flac').read_bytes() == b"fLaC..."
+
+     @unittest.skipIf(platform.system() == "Windows", "GH-123693")
+     def test_backslash_not_separator(self):
+         """
+         In a zip file, backslashes are not separators.
+         """
+         data = io.BytesIO()
+         zf = zipfile.ZipFile(data, "w")
+         zf.writestr(DirtyZipInfo.for_name("foo\\bar", zf), b"content")
+         zf.filename = ''
+         root = zipfile.Path(zf)
+         (first,) = root.iterdir()
+         assert not first.is_dir()
```

```
3335 +         assert first.name == 'foo\\bar',
     first.name
3336 +
3337 +
3338 + class DirtyZipInfo(zipfile.ZipInfo):
3339 +     """
3340 +     Bypass name sanitization.
3341 +     """
3342 +
3343 +     def __init__(self, filename, *args,
     **kwargs):
3344 +         super().__init__(filename, *args,
     **kwargs)
3345 +         self.filename = filename
3346 +
3347 +     @classmethod
3348 +     def for_name(cls, name, archive):
3349 +         """
3350 +         Construct the same way that
     ZipFile.writestr does.
3351 +
3352 +         TODO: extract this functionality and
     re-use
3353 +         """
3354 +         self = cls(filename=name,
     date_time=time.localtime(time.time())[:6])
3355 +         self.compress_type =
     archive.compression
3356 +         self.compress_level =
     archive.compresslevel
3357 +         if self.filename.endswith('/'):  #
     pragma: no cover
3358 +             self.external_attr = 0o40775 << 16
     # drwxrwxr-x
3359 +             self.external_attr |= 0x10  # MS-
     DOS directory flag
3360 +         else:
3361 +             self.external_attr = 0o600 << 16  #
     ?rw-------
3362 +         return self
3299 3363
3300 3364
3301 3365   class StripExtraTests(unittest.TestCase):
```

69 ▪▪▪▪▫ Lib/zipfile.py

```
2152 2152   def _ancestry(path):
2153 2153       """
2154 2154       Given a path with elements separated by
2155      -     posixpath.sep, generate all elements of
           that path
     2155 +     posixpath.sep, generate all elements of
           that path.
2156 2156
2157 2157       >>> list(_ancestry('b/d'))
2158 2158       ['b/d', 'b']
2164 2164       ['b']
```

```
2165 2165       >>> list(_ancestry(''))
2166 2166       []
     2167 +
     2168 +     Multiple separators are treated like a
          +   single.
     2169 +
     2170 +     >>> list(_ancestry('//b//d///f//'))
     2171 +     ['//b//d///f', '//b//d', '//b']
2167 2172       """
2168 2173       path = path.rstrip(posixpath.sep)
2169      -     while path and path != posixpath.sep:
     2174 +     while path.rstrip(posixpath.sep):
2170 2175           yield path
2171 2176           path, tail = posixpath.split(path)
2172 2177
2183 2188       return
          +   itertools.filterfalse(set(subtrahend).__contain
          +   s__, minuend)
2184 2189
2185 2190
2186      - class SanitizedNames:
2187      -     """
2188      -     ZipFile mix-in to ensure names are
          +   sanitized.
2189      -     """
2190      -
2191      -     def namelist(self):
2192      -         return list(map(self._sanitize,
          +   super().namelist()))
2193      -
2194      -     @staticmethod
2195      -     def _sanitize(name):
2196      -         r"""
2197      -         Ensure a relative path with posix
          +   separators and no dot names.
2198      -         Modeled after
2199      -
          +   https://github.com/python/cpython/blob/bcc1be39
          +   cb1d04ad9fc0bd1b9193d3972835a57c/Lib/zipfile/__
          +   init__.py#L1799-L1813
2200      -         but provides consistent cross-platform
          +   behavior.
2201      -         >>> san = SanitizedNames._sanitize
2202      -         >>> san('/foo/bar')
2203      -         'foo/bar'
2204      -         >>> san('//foo.txt')
2205      -         'foo.txt'
2206      -         >>> san('foo/.././bar.txt')
2207      -         'foo/bar.txt'
2208      -         >>> san('foo../.bar.txt')
2209      -         'foo../.bar.txt'
2210      -         >>> san('\\foo\\bar.txt')
2211      -         'foo/bar.txt'
2212      -         >>> san('D:\\foo.txt')
2213      -         'D/foo.txt'
2214      -         >>> san('\\\\server\\share\\file.txt')
```

```
2215        -         'server/share/file.txt'
2216        -         >>> san('\\\\?\\GLOBALROOT\\Volume3')
2217        -         '?/GLOBALROOT/Volume3'
2218        -         >>> san('\\\\.\\PhysicalDrive1\\root')
2219        -         'PhysicalDrive1/root'
2220        -         Retain any trailing slash.
2221        -         >>> san('abc/')
2222        -         'abc/'
2223        -         Raises a ValueError if the result is
            empty.
2224        -         >>> san('../..')
2225        -         Traceback (most recent call last):
2226        -         ...
2227        -         ValueError: Empty filename
2228        -         """
2229        -
2230        -         def allowed(part):
2231        -             return part and part not in {'..',
            '.'}
2232        -
2233        -         # Remove the drive letter.
2234        -         # Don't use ntpath.splitdrive, because
            that also strips UNC paths
2235        -         bare = re.sub('^([A-Z]):', r'\1', name,
            flags=re.IGNORECASE)
2236        -         clean = bare.replace('\\', '/')
2237        -         parts = clean.split('/')
2238        -         joined = '/'.join(filter(allowed,
            parts))
2239        -         if not joined:
2240        -             raise ValueError("Empty filename")
2241        -         return joined + '/' *
            name.endswith('/')
2242        -
2243        -
2244        - class CompleteDirs(SanitizedNames, ZipFile):
        2191 + class CompleteDirs(ZipFile):
2245    2192      """
2246    2193      A ZipFile subclass that ensures that
            implied directories
2247    2194      are always included in the namelist.
```

3

Misc/NEWS.d/next/Library/2024-08-26-13-45-20.gh-issue-1232…

```
...  ...  @@ -0,0 +1,3 @@
        1  + Applied a more surgical fix for malformed payloads
             in :class:`zipfile.Path`
        2  + causing infinite loops (gh-122905) without
             breaking contents using
        3  + legitimate characters.
```

0 comments on commit `0aa1ee2`

Please sign in to comment.