

Commit

✖ [3.10] gh-123270: Replaced SanitizedNames with a more surgical fix. (G...
...H-123354) (#123426)

Browse files

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 8 + import platform

9 9 import posixpath

10 10 import string

11 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.

3291 """

3292 data = io.BytesIO()

3293 zf = zipfile.ZipFile(data, "w")

3294 zf.writestr("../parent.txt", b"content")

3295 zf.filename = ''

3296 root = zipfile.Path(zf)

```

3294         -         assert list(map(str, root.iterdir()))
3295         -         'one-slash.txt',
3296         -         'two-slash.txt',
3297         -         'parent.txt',
3298         -         ]
3299     +         assert list(map(str, root.iterdir()))
3300     +         == ['../']
3301     +         assert
3302     +         root.joinpath('..').joinpath('parent.txt').read
3303     +         _bytes() == b'content'
3304     +
3305     +         @unittest.skipIf(platform.system() ==
3306     +         "Windows", "GH-123693")
3307     +         def test_unsupported_names(self):
3308     +             """
3309     +             Path segments with special characters
3310     +             are readable.
3311     +
3312     +             On some platforms or file systems,
3313     +             characters like
3314     +             ``:`` and ``?`` are not allowed, but
3315     +             they are valid
3316     +             in the zip file.
3317     +             """
3318     +             data = io.BytesIO()
3319     +             zf = zipfile.ZipFile(data, "w")
3320     +             zf.writestr("path?", b"content")
3321     +             zf.writestr("V: NMS.flac", b"fLaC...")
3322     +             zf.filename = ''
3323     +             root = zipfile.Path(zf)
3324     +             contents = root.iterdir()
3325     +             assert next(contents).name == 'path?'
3326     +             item = next(contents)
3327     +             assert item.name == 'V: NMS.flac',
3328     +             item.name
3329     +             assert root.joinpath('V:
3330     +             NMS.flac').read_bytes() == b"fLaC..."
3331     +
3332     +             @unittest.skipIf(platform.system() ==
3333     +             "Windows", "GH-123693")
3334     +             def test_backslash_not_separator(self):
3335     +                 """
3336     +                 In a zip file, backslashes are not
3337     +                 separators.
3338     +                 """
3339     +                 data = io.BytesIO()
3340     +                 zf = zipfile.ZipFile(data, "w")
3341     +                 zf.writestr(DirtyZipInfo.for_name("foo\\bar",
3342     +                 zf), b"content")
3343     +                 zf.filename = ''
3344     +                 root = zipfile.Path(zf)
3345     +                 (first,) = root.iterdir()
3346     +                 assert not first.is_dir()

```

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



69




 Lib/zipfile.py
 

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

```

2165 >>> list(_ancestry(''))
2166 []
2167 +
2168 + Multiple separators are treated like a
      single.
2169 +
2170 + >>> list(_ancestry('//b//d///f//'))
2171 + ['//b//d///f', '//b//d', '//b']
2172 + ""
2173 path = path.rstrip(posixpath.sep)
2169 - while path and path != posixpath.sep:
2174 + while path.rstrip(posixpath.sep):
2175     yield path
2176     path, tail = posixpath.split(path)
2177
2178 return
2179
2180 itertools.filterfalse(set(subtrahend).__contains__, minuend)
2181
2182
2183
2184
2185
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 -
2202 -         >>> san = SanitizedNames._sanitize
2203 -         >>> san('/foo/bar')
2204 -         'foo/bar'
2205 -         >>> san('//foo.txt')
2206 -         'foo.txt'
2207 -         >>> san('foo/../../bar.txt')
2208 -         'foo/bar.txt'
2209 -         >>> san('foo../.bar.txt')
2210 -         'foo../.bar.txt'
2211 -         >>> san('\\foo\\bar.txt')
2212 -         'foo/bar.txt'
2213 -         >>> san('D:\\foo.txt')
2214 -         'D/foo.txt'
2215 -         >>> 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	-	
2234	-	# Remove the drive letter.
2235	-	# Don't use ntpath.splitdrive, because
2236	-	that also strips UNC paths
2237	-	bare = re.sub('^[A-Z]):', r'\1', name,
2238	-	flags=re.IGNORECASE)
2239	-	clean = bare.replace('\\\\', '/')
2240	-	parts = clean.split('/')
2241	-	joined = '/'.join(filter(allowed,
2242	-	parts))
2243	-	if not joined:
2244	-	raise ValueError("Empty filename")
2245	-	return joined + '/' *
2246	-	name.endswith('/')
2247	-	
2248	-	
2249	-	
2250	-	
2251	-	
2252	-	
2253	-	
2254	-	
2255	-	
2256	-	
2257	-	
2258	-	
2259	-	
2260	-	
2261	-	
2262	-	
2263	-	
2264	-	
2265	-	
2266	-	
2267	-	
2268	-	
2269	-	
2270	-	
2271	-	
2272	-	
2273	-	
2274	-	
2275	-	
2276	-	
2277	-	
2278	-	
2279	-	
2280	-	
2281	-	
2282	-	
2283	-	
2284	-	
2285	-	
2286	-	
2287	-	
2288	-	
2289	-	
2290	-	
2291	-	
2292	-	
2293	-	
2294	-	
2295	-	
2296	-	
2297	-	
2298	-	
2299	-	
2300	-	
2301	-	
2302	-	
2303	-	
2304	-	
2305	-	
2306	-	
2307	-	
2308	-	
2309	-	
2310	-	
2311	-	
2312	-	
2313	-	
2314	-	
2315	-	
2316	-	
2317	-	
2318	-	
2319	-	
2320	-	
2321	-	
2322	-	
2323	-	
2324	-	
2325	-	
2326	-	
2327	-	
2328	-	
2329	-	
2330	-	
2331	-	
2332	-	
2333	-	
2334	-	
2335	-	
2336	-	
2337	-	
2338	-	
2339	-	
2340	-	
2341	-	
2342	-	
2343	-	
2344	-	
2345	-	
2346	-	
2347	-	
2348	-	
2349	-	
2350	-	
2351	-	
2352	-	
2353	-	
2354	-	
2355	-	
2356	-	
2357	-	
2358	-	
2359	-	
2360	-	
2361	-	
2362	-	
2363	-	
2364	-	
2365	-	
2366	-	
2367	-	
2368	-	
2369	-	
2370	-	
2371	-	
2372	-	
2373	-	
2374	-	
2375	-	
2376	-	
2377	-	
2378	-	
2379	-	
2380	-	
2381	-	
2382	-	
2383	-	
2384	-	
2385	-	
2386	-	
2387	-	
2388	-	
2389	-	
2390	-	
2391	-	
2392	-	
2393	-	
2394	-	
2395	-	
2396	-	
2397	-	
2398	-	
2399	-	
2400	-	
2401	-	
2402	-	
2403	-	
2404	-	
2405	-	
2406	-	
2407	-	
2408	-	
2409	-	
2410	-	
2411	-	
2412	-	
2413	-	
2414	-	
2415	-	
2416	-	
2417	-	
2418	-	
2419	-	
2420	-	
2421	-	
2422	-	
2423	-	
2424	-	
2425	-	
2426	-	
2427	-	
2428	-	
2429	-	
2430	-	
2431	-	
2432	-	
2433	-	
2434	-	
2435	-	
2436	-	
2437	-	
2438	-	
2439	-	
2440	-	
2441	-	
2442	-	
2443	-	
2444	-	
2445	-	
2446	-	
2447	-	
2448	-	
2449	-	
2450	-	
2451	-	
2452	-	
2453	-	
2454	-	
2455	-	
2456	-	
2457	-	
2458	-	
2459	-	
2460	-	
2461	-	
2462	-	
2463	-	
2464	-	
2465	-	
2466	-	
2467	-	
2468	-	
2469	-	
2470	-	
2471	-	
2472	-	
2473	-	
2474	-	
2475	-	
2476	-	
2477	-	
2478	-	
2479	-	
2480	-	
2481	-	
2482	-	
2483	-	
2484	-	
2485	-	
2486	-	
2487	-	
2488	-	
2489	-	
2490	-	
2491	-	
2492	-	
2493	-	
2494	-	
2495	-	
2496	-	
2497	-	
2498	-	
2499	-	
2500	-	
2501	-	
2502	-	
2503	-	
2504	-	
2505	-	
2506	-	
2507	-	
2508	-	
2509	-	
2510	-	
2511	-	
2512	-	
2513	-	
2514	-	
2515	-	
2516	-	
2517	-	
2518	-	
2519	-	
2520	-	
2521	-	
2522	-	
2523	-	
2524	-	
2525	-	
2526	-	
2527	-	
2528	-	
2529	-	
2530	-	
2531	-	
2532	-	
2533	-	
2534	-	
2535	-	
2536	-	
2537	-	
2538	-	
2539	-	
2540	-	
2541	-	
2542	-	
2543	-	
2544	-	
2545	-	
2546	-	
2547	-	
2548	-	
2549	-	
2550	-	
2551	-	
2552	-	
2553	-	
2554	-	
2555	-	
2556	-	
2557	-	
2558	-	
2559	-	
2560	-	
2561	-	
2562	-	
2563	-	
2564	-	
2565	-	
2566	-	
2567	-	
2568	-	
2569	-	
2570	-	
2571	-	
2572	-	
2573	-	
2574	-	
2575	-	
2576	-	
2577	-	
2578	-	
2579	-	
2580	-	
2581	-	
2582	-	
2583	-	
2584	-	
2585	-	
2586	-	
2587	-	
2588	-	
2589	-	
2590	-	
2591	-	
2592	-	
2593	-	
2594	-	
2595	-	
2596	-	
2597	-	
2598	-	
2599	-	
2600	-	
2601	-	
2602	-	
2603	-	
2604	-	
2605	-	
2606	-	
2607	-	
2608	-	
2609	-	
2610	-	
2611	-	
2612	-	
2613	-	
2614	-	
2615	-	
2616	-	
2617	-	
2618	-	
2619	-	
2620	-	
2621	-	
2622	-	
2623	-	
2624	-	
2625	-	
2626	-	
2627	-	
2628	-	
2629	-	
2630	-	
2631	-	
2632	-	
2633	-	
2634	-	
2635	-	
2636	-	
2637	-	
2638	-	
2639	-	
2640	-	
2641	-	
2642	-	
2643	-	
2644	-	
2645	-	
2646	-	
2647	-	
2648	-	
2649	-	
2650	-	
2651	-	
2652	-	
2653	-	
2654	-	
2655	-	
2656	-	
2657	-	
2658	-	
2659	-	
2660	-	
2661	-	
2662	-	
2663	-	
2664	-	
2665	-	
2666	-	
2667	-	
2668	-	
2669	-	
2670	-	
2671	-	
2672	-	
2673	-	
2674	-	
2675	-	
2676	-	
2677	-	
2678	-	
2679	-	
2680	-	
2681	-	
2682	-	
2683	-	
2684	-	
2685	-	
2686	-	
2687	-	
2688	-	
2689	-	
2690	-	
2691	-	
2692	-	
2693	-	
2694	-	
2695	-	
2696	-	
2697	-	
2698	-	
2699	-	
2700	-	
2701	-	
2702	-	
2703	-	
2704	-	
2705	-	
2706	-	
2707	-	
2708	-	
2709	-	
2710	-	
2711	-	
2712	-	
2713	-	
2714	-	
2715	-	
2716	-	
2717	-	
2718	-	
2719	-	
2720	-	
2721	-	
2722	-	
2723	-	
2724	-	
2725	-	
2726	-	
2727	-	
2728	-	
2729	-	
2730	-	
2731	-	
2732	-	
2733	-	
2734	-	
2735	-	
2736	-	
2737	-	
2738	-	
2739	-	
2740	-	
2741	-	
2742	-	
2743	-	
2744	-	
2745	-	
2746	-	
2747	-	
2748	-	
2749	-	
2750	-	
2751	-	
2752	-	
2753	-	
2754	-	
2755	-	
2756	-	
2757	-	
2758	-	
2759	-	
2760	-	
2761	-	
2762	-	
2763	-	
2764	-	
2765	-	