Rewriting pyc files for fun and reproducibility

Zbigniew Jędrzejewski-Szmek

zbyszek@in.waw.pl

FOSDEM, Bruxelles/Brussel 2.2.2025

About me

- RedHatter working on systemd and various open source things
- Fedora contributor working on package build reproducibility
- Long time ago some small contributions to CPython

What is build reproducibility?

> *A build is reproducible if given the same source code, build environment and build instructions, any party can recreate bit-by-bit identical copies of all specified artifacts.*
>
> – reproducible-builds.org

Two angles of motiviation:

- Security (independent verification of suply chain security)
- Quality (issues in hardware, build systems, packaging, software)

How do we achieve build reproducibility?

- packages are built in a container with no network access
- dependencies are installed as packages
- build process must be determininistic
- operation independent of the environment (e.g. time clamped to `$SOURCE_DATE_EPOCH`)

To solve issues that cannot be resolved by changing individual packages or tools, we apply a post-build cleanup...

How do we achieve build reproducibility?

post-build cleanups

Debian has `strip-nondeterminism`
Fedora now has `add-determinism`

`add-determinism` runs after the install phase of the package build

```
+ /usr/bin/add-determinism --brp -j2 /builddir/build/BUILD/python-tables-3.10.1-build/BUILDROOT
/…/BUILDROOT/…/tables/misc/__pycache__/__init__.cpython-313.pyc:
           rewriting with normalized contents
/…/BUILDROOT/…/tables/misc/__pycache__/enum.cpython-313.pyc:
           rewriting with normalized contents
...
Scanned 36 directories and 362 files,
               processed 94 inodes,
               94 modified (30 replaced + 64 rewritten),
               0 unsupported format, 0 errors
```

- ~~ownership and mtimes in `*.zip`, `*.jar`, and `*.a` archives~~
- ~~timestamps in javadoc `*.html`~~
- python `*.pyc` files

The intro is finally over, phew!

pyc files

i.e. the thing this talk is supposed to be about...

`.py` source file $\rightarrow$ `.pyc` cached bytecode

- CPython will (attempt to) write .pyc files every time it loads a .py file
- writing may fail
- Fedora packages include .pyc files for speed and reliability

pyc contents

basic objects

[VERSION1 VERSION2 MAGIC1 MAGIC2 4–12 byte header]
[object1] [object2] ... [object...]

Object can be:
an 32-bit integer: ['i' BYTE4 BYTE3 BYTE2 BYTE1]
an 64-bit float: ['g' BYTE8 BYTE7 ... BYTE2 BYTE1]
an 2×64-bit complex: ['y' REAL8 ... REAL1 IMAG8 ... IMAG1]
a Python integer: ['l' SIZE4 SIZE3 SIZE2 SIZE1
DIGIT1_4 DIGIT1_3 DIGIT1_2 DIGIT1_1
... DIGITn_1]
normal string: ['s'/'t'/'u'/'a'/'A' SIZE4 ... SIZE1 CHAR1 ... CHARn]
short ASCII string: ['z'/'Z' SIZE CHAR1 ... CHARn]
special Python stuff: ['N'/'F'/'T'/'.'/'S']

pyc contents

complex objects

list: ['[' SIZE4 ... SIZE1 [object1] ... [objectn]]
tuple: ['(' SIZE4 ... SIZE1 [object1] ... [objectn]]
['')' SIZE [object1] ... [objectn]]
sets: ['<'/'>' SIZE4 ... SIZE1 [object1] ... [objectn]]
dicts: ['{' [key] [value] ...[key] [value] '0']

New in Python 3.14 — slice objects: [':' [start] [stop] [step]]

pyc contents

very complex objects

code object: ['c' [ARGCOUNT] [POSONLYARGCOUNT] [KWONLYARGCOUNT] ... [FLAGS] [code] [consts] [names] ... [filename] [name] [qualname] ...]

the whole pyc file:
[VERSION1 VERSION2 MAGIC1 MAGIC2 4–12 byte header] [object1] [object2] ... [object...][code][code [string1] [string2] ... [list ...]]

pyc contents

reference objects

reference: ['r' BYTE4 ... BYTE1]

[HEADER] [object1] [object2 🚩] [object3] [object4 🚩] ...
...
[REF 0] ... [object] ... [REF 1]

```
Code "<module>" 204/(ref to 204)"<module>" 0
  (ref to 22)"/usr/lib/python3.12/site-packages/elftools/construct/adapters.py":1
  argcount=0 posonlyargcount=0 kwonlyargcount=0 stacksize=5 flags=0
  -code: [560 bytes]
  -consts: (
    1 2,
    ("Adapter" 3, "AdaptationError" 4, "Pass" 5),
    ("int_to_bin" 6, "bin_to_int" 7, "swap_bytes" 8),
    ("FlagsContainer" 9, "HexString" 10),
    ("BytesIO" 11, "decodebytes" 12),
    Code (ref to 14)"BitIntegerError"/(ref to 14)"BitIntegerError"
      "/usr/lib/python3.12/site-packages/elftools/construct/adapters.py" 22:10
      argcount=0 posonlyargcount=0 kwonlyargcount=0 stacksize=1 flags=0
      -code: [16 bytes]
      -consts: ("BitIntegerError" 14, None)
      -names: ("__name__" 16, "__module__" 17, "__qualname__" 18, "__slots__" 19) 15
      -locals+names: () 20
      -locals+kinds: [] 21
      -linetable: [7 bytes]
      -exceptiontable: (ref to 21)[] 21,
    (ref to 14)"BitIntegerError",
    Code (ref to 25)"MappingError"/(ref to 25)"MappingError"
      (ref to 22)"/usr/lib/python3.12/site-packages/elftools/construct/adapters.py":12
      argcount=0 posonlyargcount=0 kwonlyargcount=0 stacksize=1 flags=0
      -code: [16 bytes]
      -consts: ("MappingError" 25, None)
      -names: (ref to 15)("__name__" 16, "__module__" 17, "__qualname__" 18, "__slots__" 19)
      -locals+names: (ref to 20)()
      -locals+kinds: (ref to 21)[]
      -linetable: (ref to 23)[7 bytes]
      -exceptiontable: (ref to 21)[] 21,
    (ref to 25)"MappingError",
    Code (ref to 27)"ConstError"/(ref to 27)"ConstError"
      (ref to 22)"/usr/lib/python3.12/site-packages/elftools/construct/adapters.py":14
      argcount=0 posonlyargcount=0 kwonlyargcount=0 stacksize=1 flags=0
      -code: [16 bytes]
      -consts: ("ConstError" 27, None)
      -names: (ref to 15)("__name__" 16, "__module__" 17, "__qualname__" 18, "__slots__" 19)
```

Irreproducibilities observed

- Only objects with  can be referenced
- Objects may be flagged without being referenced
  $\rightarrow$ “unused flags”
- Not all objects have to replaced by references
- Many different equivalent serializations

Solution:

- rewrite the object stream with minimal number of flags and maximal number of references

```
-Code "<module>" ▶204/(ref to 204)"<module>" ▶0
-  (ref to 22)"/usr/lib/python3.12/site-packages/elftools/construct/adapters.py":1
+Code "<module>" ▶118/(ref to 118)"<module>"
+  (ref to 20)"/usr/lib/python3.12/site-packages/elftools/construct/adapters.py":1
   argcount=0 posonlyargcount=0 kwonlyargcount=0 stacksize=5 flags=0
   -code: [560 bytes]
   -consts: (
-    1 ▶2,
-    ("Adapter" ▶3, "AdaptationError" ▶4, "Pass" ▶5),
-    ("int_to_bin" ▶6, "bin_to_int" ▶7, "swap_bytes" ▶8),
-    ("FlagsContainer" ▶9, "HexString" ▶10),
-    ("BytesIO" ▶11, "decodebytes" ▶12),
-    Code (ref to 14)"BitIntegerError"/(ref to 14)"BitIntegerError"
-      "/usr/lib/python3.12/site-packages/elftools/construct/adapters.py" ▶22:10
+    1 ▶0,
+    ("Adapter" ▶1, "AdaptationError" ▶2, "Pass" ▶3),
+    ("int_to_bin" ▶4, "bin_to_int" ▶5, "swap_bytes" ▶6),
+    ("FlagsContainer" ▶7, "HexString" ▶8),
+    ("BytesIO" ▶9, "decodebytes" ▶10),
+    Code (ref to 12)"BitIntegerError"/(ref to 12)"BitIntegerError"
+      "/usr/lib/python3.12/site-packages/elftools/construct/adapters.py" ▶20:10
       argcount=0 posonlyargcount=0 kwonlyargcount=0 stacksize=1 flags=0
       -code: [16 bytes]
-      -consts: ("BitIntegerError" ▶14, None)
-      -names: ("__name__" ▶16, "__module__" ▶17, "__qualname__" ▶18, "__slots__" ▶19) ▶15
-      -locals+names: () ▶20
-      -locals+kinds: [] ▶21
-      -linetable: []
-      -exceptiontable: (ref to 21)[] ▶21,
-    (ref to 14)"BitIntegerError",
+      -consts: ("BitIntegerError" ▶12, None)
+      -names: ("__name__" ▶14, "__module__" ▶15, "__qualname__" ▶16, "__slots__" ▶17) ▶13
+      -locals+names: () ▶18
+      -locals+kinds: [] ▶19
+      -linetable: (ref to 19)[]
+      -exceptiontable: (ref to 19)[] ▶19,
+    (ref to 12)"BitIntegerError",
```

Questions & further steps

- CPython could be improved to ... maximize references and minimize flags
- Is it OK to reference mutable objects?
- Can we change 's' $\rightarrow$ 'z'? (3 bytes less, more references)
- Can we change 'A'/'Z' $\rightarrow$ 'a/'z'? (more references)
- Can we change 'I' $\rightarrow$ 'i'? (4 bytes less, simpler processing, more references)
- Can we change '[' $\leftarrow\rightarrow$ '('/')'? (more references, less bytes)
- `add-determinism -p` is useful, but no bytecode decoder
- `diffoscope` should use `marshalparser -p`/ `marshal-parser -p`/`add-determinism -p`

Links and references

For more info:

- reproducible-builds.org
- Fedora ReproduciblePackageBuilds Change
- Flock 2024 Reproducible builds in Fedora talk

Tools:

- github.com/keszybz/add-determinism
- packages.debian.org/sid/dh-strip-nondeterminism
- github.com/fedora-python/marshalparser
- crates.io/crates/marshal-parser