Linux编程之序列化存储Python对象(下)（1） PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/144/2021_2022_Linux_E7_BC_96_E7_A8_c103_144595.htm 相等，但并不总是相同 正如在上一个示例所暗示的，只有在这些对象引用内存中同一个对象时，它们才是相同的。在 pickle 情形中，每个对象被恢复到一个与原来对象相等的对象，但不是同一个对象。换句话说，每个 pickle 都是原来对象的一个副本： >>> j = [1, 2, 3] >>> k = j >>> k is j 1 >>> x = pickle.dumps(k) >>> y = pickle.loads(x) >>> y [1, 2, 3] >>> y == k 1 >>> y is k 0 >>> y is j 0 >>> k is j 1 清单 8. 作为原来对象副本的被恢复的对象同时，我们看到 Python 能够维护对象之间的引用，这些对象是作为一个单元进行 pickle 的。然而，我们还看到分别调用 dump() 会使 Python 无法维护对在该单元外部进行 pickle 的对象的引用。相反，Python 复制了被引用对象，并将副本和被 pickle 的对象存储在一起。对于 pickle 和恢复单个对象层次结构的应用程序，这是没有问题的。但要意识到还有其它情形。 值得指出的是，有一个选项确实允许分别 pickle 对象，并维护相互之间的引用，只要这些对象都是 pickle 到同一文件即可。pickle 和 cPickle 模块提供了一个 Pickler（与此相对应是 Unpickler），它能够跟踪已经被 pickle 的对象。通过使用这个 Pickler，将会通过引用而不是通过值来 pickle 共享和循环引用： >>> f = file(temp.pkl, w) >>> pickler = pickle.Pickler(f) >>> pickler.dump(a) >>> pickler.dump(b) >>> f.close() >>> f = file(temp.pkl, r) >>> unpickler = pickle.Unpickler(f) >>> c = unpickler.load() >>> d = unpickler.load() >>> c[2] [3, 4, [1, 2,

[...]]] >>> d[2] [1, 2, [3, 4, [...]]] >>> c[2] is d 1 >>> d[2] is c 1 清单 9. 维护分别 pickle 的对象间的引用不可 pickle 的对象 一些对象类型是不可 pickle 的。例如，Python 不能 pickle 文件对象（或者任何带有对文件对象引用的对象），因为 Python 在 unpickle 时不能保证它可以重建该文件的状态（另一个示例比较难懂，在这类文章中不值得提出来）。试图 pickle 文件对象会导致以下错误： >>> f = file(temp.pkl, w) >>> p = pickle.dumps(f) Traceback (most recent call last): File "", line 1, in ? File "/usr/lib/python2.2/copy_reg.py", line 57, in _reduce raise TypeError, "cant pickle %s objects" % base.__name__ TypeError: cant pickle file objects 清单 10. 试图 pickle 文件对象的结果 