Linux编程之序列化存储Python对象(下)（3） 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_144592.htm 特殊的状态方法 前面提到对一些对象类型（譬如，文件对象）不能进行 pickle。处理这种不能 pickle 的对象的实例属性时可以使用特殊的方法（_getstate_() 和 _setstate_()）来修改类实例的状态。这里有一个 Foo 类的示例，我们已经对它进行了修改以处理文件对象属性： class Foo(object): def __init__(self, value, filename): self.value = value self.logfile = file(filename, w) def __getstate__(self): """Return state values to be pickled.""" f = self.logfile return (self.value, f.name, f.tell()) def __setstate__(self, state): """Restore state from the unpickled state values.""" self.value, name, position = state f = file(name, w) f.seek(position) self.logfile = f 清单 15. 处理不能 pickle 的实例属性pickle Foo 的实例时，Python 将只 pickle 当它调用该实例的 _getstate_() 方法时返回给它的值。类似的，在 unpickle 时，Python 将提供经过 unpickle 的值作为参数传递给实例的 _setstate_() 方法。在 _setstate_() 方法内，可以根据经过 pickle 的名称和位置信息来重建文件对象，并将该文件对象分配给这个实例的 logfile 属性。 模式改进 随着时间的推移，您会发现自己必须要更改类的定义。如果已经对某个类实例进行了 pickle，而现在又需要更改这个类，则您可能要检索和更新那些实例，以便它们能在新的类定义下继续正常工作。而我们已经看到在对类或模块进行某些更改时，会出现一些错误。幸运的是，pickle 和 unpickle 过程提供了一些 hook，我们可以用它们来支持这种模式改进的需要。 在这一

节，我们将探讨一些方法来预测常见问题以及如何解决这些问题。由于不能 pickle 类实例代码，因此可以添加、更改和除去方法，而不会影响现有的经过 pickle 的实例。出于同样的原因，可以不必担心类的属性。您必须确保包含类定义的代码模块在 unpickle 环境中可用。同时还必须为这些可能导致 unpickle 问题的更改做好规划，这些更改包括：更改类名、添加或除去实例的属性以及改变类定义模块的名称或位置。