|  |  |
| --- | --- |
| 1. unique\_ptr 2. 传递unique\_ptr参数和返回unique\_ptr 3. 向unique\_ptr传递删除器 | Unique\_ptr<double> p1;  Unique\_ptr<int> p2(new int(42));  Unique\_ptr p2(p1);//错误 不支持拷贝或赋值  Unique\_ptr<string> p2(p1.release());  P2.reset(p3.release());  P2.release();  Auto p = p2.release();  Unique\_ptr<int> clone(int p){  Return unique\_ptr<int>(new int(p));//或者局部对象  }  Unqiue\_ptr<objT,delT> p(new objT,fcn);  Void f(destination &d){  Connection c = connect(&d);  Unique\_ptr<connection,decltype(end\_connection)\*  > P(&c,end\_connection);  } |
| 1. unique\_ptr 2. 只能有一个unique\_ptr指向一个给定对象。 3. 没有类似make\_ptr的标准库函数返回一个unique\_ptr。定义时需要绑定到一个new返回的指针上 4. 必须使用直接初始化方式unique\_ptr<string> p1(new string("Stegosaurus")); 5. 由于只能有一个unique\_ptr拥有它指向的对象，所以不支持普通的拷贝或赋值操作 6. 虽然不能拷贝或赋值unique\_ptr，但可以通过使用releas或reset将指针的所有权从一个非const unique\_ptr转移到另一个unique。 7. Release()会将对象置为空，转移给新的unique\_ptr，若没有接受的unique\_ptr会导致无法销毁这个内存了。Reset是先销毁左侧的指向内存再指向新的内存。 8. 传递unique\_ptr参数和返回unique\_ptr 9. 但是将要删除unique\_ptr时编译器允许拷贝或者赋值操作   Note：向后兼容：auto\_ptr，以前版本，不要用了   1. 向unique\_ptr传递删除器 2. 默认是delete释放它指向的对象。 3. 必须在尖括号中unique\_ptr指向类型之后提供删除器类型。在创建或reset一个这种unique\_ptr类型对象时，必须提供一个指定类型的可调用对象(删除器) 4. Decltype指明函数指针类型，但需要加上\*，表示我们在使用该类型的一个指针。 | |

