定义:对象间的一种一对多（变化）的依赖关系，以便当一个对象(Subject)的状态发生改变时，所有依赖于它的对象都得到通知并自动更新。

主要解决：一个对象状态改变给其他对象通知的问题，而且要考虑到易用和低耦合，保证高度的协作。

何时使用：一个对象（目标对象）的状态发生改变，所有的依赖对象（观察者对象）都将得到通知，进行广播通知。

如何解决：使用面向对象技术，可以将这种依赖关系弱化。

关键代码：在抽象类里有一个 ArrayList 存放观察者们。

优点： 1、观察者和被观察者是抽象耦合的。 2、建立一套触发机制。

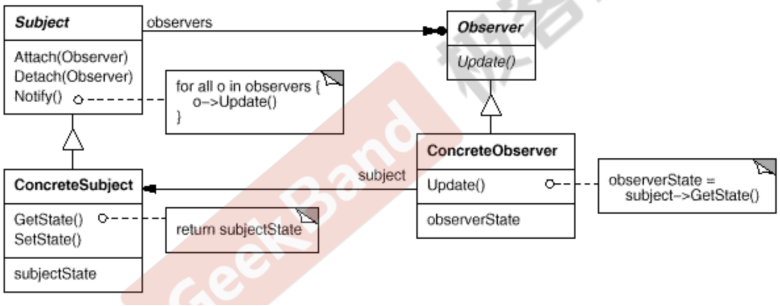
缺点： 1、如果一个被观察者对象有很多的直接和间接的观察者的话，将所有的观察者都通知到会花费很多时间。 2、如果在观察者和观察目标之间有循环依赖的话，观察目标会触发它们之间进行循环调用，可能导致系统崩溃。 3、观察者模式没有相应的机制让观察者知道所观察的目标对象是怎么发生变化的，而仅仅只是知道观察目标发生了变化。

使用场景：

- 一个抽象模型有两个方面，其中一个方面依赖于另一个方面。将这些方面封装在独立的对象中使它们可以各自独立地改变和复用。
- 一个对象的改变将导致其他一个或多个对象也发生改变，而不知道具体有多少对象将发生改变，可以降低对象之间的耦合度。
- 一个对象必须通知其他对象，而并不知道这些对象是谁。
- 需要在系统中创建一个触发链，A对象的行为将影响B对象，B对象的行为将影响C对象……，可以使用观察者模式创建一种链式触发机制。

模式优点:
观察者模式可以实现表示层和数据逻辑层的分离，并定义了稳定的消息更新传递机制，抽象了更新接口，使得可以有各种各样不同的表示层作为具体观察者角色。
观察者模式在观察目标和观察者之间建立一个抽象的耦合。
观察者模式支持广播通信。
观察者模式符合“开闭原则”的要求。

模式缺点:
如果一个观察目标对象有很多直接和间接的观察者的话，将所有的观察者都通知到会花费很多时间。
如果在观察者和观察目标之间有循环依赖的话，观察目标会触发它们之间进行循环调用，可能导致系统崩溃。
观察者模式没有相应的机制让观察者知道所观察的目标对象是怎么发生变化的，而仅仅只是知道观察目标发生了变化。



观察者模式角色如下：

抽象主题（Subject）角色：抽象主题角色提供维护一个观察者对象集合的操作方法，对集合的增加、删除等。

具体主题（ConcreteSubject）角色：将有关状态存入具体的观察者对象；在具体主题的内部状态改变时，给所有登记过的观察者发通知。具体主题角色负责实现抽象主题中的方法。

抽象观察者（Observer）角色：为具体观察者提供一个更新接口。

具体观察者（ConcreteObserver）角色：存储与主题相关的自洽状态，实现抽象观察者提供的更新接口。

Subject（目标）

——目标知道它的观察者。可以有任意多个观察者观察同一个目标；

——提供注册和删除观察者对象的接口。

Observer（观察者）

——为那些在目标发生改变时需获得通知的对象定义一个更新接口。

ConcreteSubject（具体目标）

——将有关状态存入各ConcreteObserver对象；

——当它的状态发生改变时，向它的各个观察者发出通知。

ConcreteObserver（具体观察者）

——维护一个指向ConcreteSubject对象的引用；

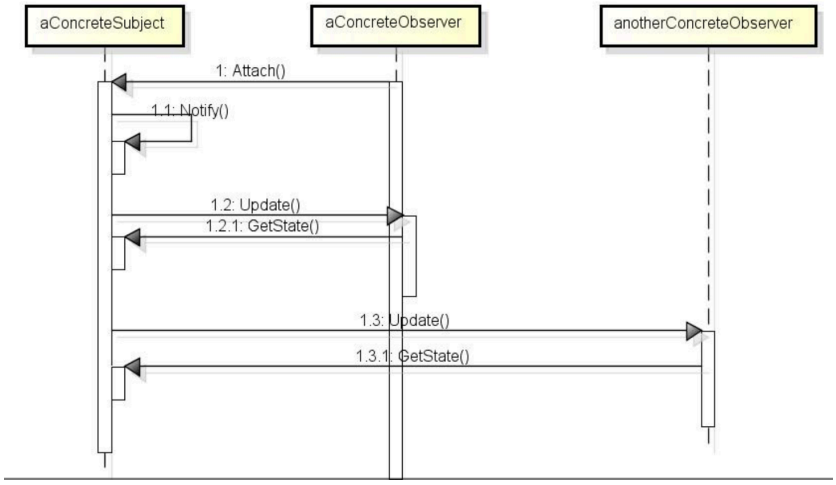
——存储有关状态，这些状态应与目标的状态保持一致；

——实现Observer的更新接口以使自身状态与目标的状态保持一致。

观察者模式按照以下方式进行协作：

1.当ConcreteSubject发生任何可能导致其观察者与其本身状态不一致的改变时，它将通知它的各个观察者；

2.在得到一个具体目标的改变通知后，ConcreteObserver对象可向目标对象查询信息。ConcreteObserver使用这些信息以使它的状态与目标对象的状态一致。




```cpp
#include <iostream>
#include <list>
using namespace std;
class Observer {
public:
    virtual void UpdateStatus(int nStatus) = 0;
    virtual ~Observer() { };
};
class Subject {
public:
    virtual void Attach(Observer *) = 0;
    virtual void Dettach(Observer *) = 0;
    virtual void Notify() = 0;
    virtual ~Subject() { }
};
class ConcreteObserverA : public Observer {
public:
    ConcreteObserverA(Subject *pObj) :m_pSubject(pObj) { }
    void UpdateStatus(int nStatus) {
        cout << "Observer A Update status, now status is " << nStatus << endl;
    }
private:
    Subject * m_pSubject;
};
class ConcreteObserverB : public Observer {
public:
    ConcreteObserverB(Subject *pObj) :m_pSubject(pObj) { }
    void UpdateStatus(int nStatus) {
        cout << "Observer B Update status, now status is " << nStatus << endl;
    }
private:
    Subject * m_pSubject;
};
class ConcreteSubject : public Subject {
public:
    void Attach(Observer *pOb);
    void Dettach(Observer *pOb);
```

```
    void Notify();
    void SetStatus(int nStatus) { m_nStatus = nStatus; }
private:
    int m_nStatus;
    list<Observer *>  m_listObserver;
};
void ConcreteSubject::Attach(Observer *pOb) {
    m_listObserver.push_back(pOb);
}
void ConcreteSubject::Dettach(Observer *pOb) {
    m_listObserver.remove(pOb);
}
void ConcreteSubject::Notify() {
   list<Observer *>::iterator it= m_listObserver.begin();
   while(it != m_listObserver.end()) {
       (*it)->UpdateStatus(m_nStatus);
       ++it;
   }
}
int main() {
    ConcreteSubject *pSubject = new ConcreteSubject();
    Observer *pObserverA = new ConcreteObserverA(pSubject);
    Observer *pObserverB = new ConcreteObserverB(pSubject);
    pSubject->SetStatus(2);
    pSubject->Attach(pObserverA);
    pSubject->Attach(pObserverB);
    pSubject->Notify();
    pSubject->SetStatus(3);
    pSubject->Dettach(pObserverB);
    pSubject->Notify();
    delete pSubject; delete pObserverA; delete pObserverB;
    return 0;
}
```

```
192:DesignPattnsStudy weishichun$ ls Observer
Observer_1.cpp          Observer_2.cpp          Observer观察者模式.pdf
192:DesignPattnsStudy weishichun$ g++ -o Observer1.out Observer_1.cpp
192:DesignPattnsStudy weishichun$ ./Observer1.out
Observer A Update status, now status is 2
Observer B Update status, now status is 2
Observer A Update status, now status is 3
192:DesignPattnsStudy weishichun$
```

```
#include <iostream>
#include <vector>
#include <string>
using namespace std;
//微信公众号和关注微信公众号的众多人，就是典型的观察者模式.
class IWeiXinUser //关注该微信公众号的微信用户抽象类，Observer
{
public:
    virtual void ConsultArticle(string strTitle) = 0; //查阅公众号文章
     virtual ~IWeiXinUser(){};
     void SetFansName(string strName)
     {
```

```
        m_strName = strName;
    }
    string m_strName;
};
class IWeiXinPublicAccount //微信公众号抽象类, Subject
{
public:
    virtual void GainFans(IWeiXinUser *objWeiXin) = 0;        //获得一个粉丝
    virtual void LoseFans(IWeiXinUser *objWeiXin) = 0;        //失去一个粉丝
    virtual void PublishArticle(string strArticleTitle) = 0; //公众号发布文章, 即通知所有观察者
    virtual ~IWeiXinPublicAccount(){};

public:
    int m_iTotalUser;    //粉丝总数
    int m_iPublishTotal; //发布文章总数
};
class Fans : public IWeiXinUser
{
public:
    Fans(IWeiXinPublicAccount *Obj) : m_pPubAcnt(Obj) {}
    void ConsultArticle(string strTitle)
    {
        cout << "粉丝: " << m_strName << "查阅了文章: " << strTitle << endl;
    }

public:
    IWeiXinPublicAccount *m_pPubAcnt; //粉丝关注的公众号
};
class YouyouDuShan : public IWeiXinPublicAccount
{
public:
    void GainFans(IWeiXinUser *objWeiXin);        //获得一个粉丝
    void LoseFans(IWeiXinUser *objWeiXin);        //失去一个粉丝
    void PublishArticle(string strArticleTitle); //公众号发布文章, 即通知所有观察者
public:
    vector<IWeiXinUser *> m_vecFans; //粉丝容器
};
void YouyouDuShan::GainFans(IWeiXinUser *pobjWeiXin)
{
    m_vecFans.push_back(pobjWeiXin);
    m_iTotalUser++;
    cout << "粉丝: " << pobjWeiXin->m_strName << "关注了悠悠独山, 目前粉丝数: " << m_iTotalUser << endl
}
void YouyouDuShan::LoseFans(IWeiXinUser *pobjWeiXin)
{
    vector<IWeiXinUser *>::iterator iter = m_vecFans.begin();
    for (; iter != m_vecFans.end(); iter++)
    {
        if (*iter == pobjWeiXin)
        {
            m_vecFans.erase(iter);
            m_iTotalUser--;
            cout << "粉丝: " << pobjWeiXin->m_strName << "取关了悠悠独山, 目前粉丝数: " << m_iTotalUser
            return;
```

```
        }
    }
    if (iter == m_vecFans.end())
    {
        cout << "粉丝: " << pobjWeiXin->m_strName << "并未关注悠悠独山, 无法取关" << endl;
    }
}
void YouyouDuShan::PublishArticle(string strArticleTitle)
{
    m_iPublishTotal++;
    cout << "悠悠独山发布了文章: " << strArticleTitle << ", 欢迎查阅, 目前共发布了: " << m_iPublishTotal
    vector<IWeiXinUser *>::iterator iter = m_vecFans.begin();
    for (; iter != m_vecFans.end(); iter++)
    {
        (*iter)->ConsultArticle(strArticleTitle);
    }
}

int main()
{
    YouyouDuShan *pYou = new YouyouDuShan();
    IWeiXinUser *pA1 = new Fans(pYou);
    pA1->SetFansName("张三");
    IWeiXinUser *pA2 = new Fans(pYou);
    pA2->SetFansName("李四");
    IWeiXinUser *pA3 = new Fans(pYou);
    pA3->SetFansName("王麻子");
    IWeiXinUser *pA4 = new Fans(pYou);
    pA4->SetFansName("小春");
    IWeiXinUser *pA5 = new Fans(pYou);
    pA5->SetFansName("丫头");
    IWeiXinUser *pA6 = new Fans(pYou);
    pA6->SetFansName("小赵");
     IWeiXinUser *pA7 = new Fans(pYou);
     pA7->SetFansName("小钱");
     IWeiXinUser *pA8 = new Fans(pYou);
     pA8->SetFansName("小孙");
     IWeiXinUser *pA9 = new Fans(pYou);
     pA9->SetFansName("小周");
     IWeiXinUser *pA10 = new Fans(pYou);
     pA10->SetFansName("小吴");
     pYou->GainFans(pA1);
     pYou->GainFans(pA2);
     pYou->PublishArticle("文章1");
     cout << "文章反响不错, 应该会涨粉, 好期待" << endl;
     pYou->GainFans(pA3);
     pYou->GainFans(pA5);
     pYou->PublishArticle("文章2");
     pYou->LoseFans(pA2);
     pYou->LoseFans(pA10);
     pYou->LoseFans(pA1);
     cout << "文章质量不好吗? 怎么掉粉了, 下次要注重文章质量" << endl;
     pYou->PublishArticle("文章3");
     pYou->GainFans(pA1);
```

```
    pYou->GainFans(pA2);
    pYou->GainFans(pA4);
    pYou->GainFans(pA6);
    pYou->GainFans(pA7);
    pYou->GainFans(pA8);
    pYou->GainFans(pA9);
    pYou->GainFans(pA10);
    cout << "看来上一篇文章质量很好，涨了这么多粉，再来一篇" << endl;
    pYou->PublishArticle("文章4");
    delete pA1; delete pA2; delete pA3; delete pA4; delete pA5; delete pA6;
    delete pA7; delete pA8; delete pA9; delete pA10; delete pYou; return 0;
}
```

```
192:DesignPattnsStudy weishichun$ g++ -o Observer2.out Observer_2.cpp
192:DesignPattnsStudy weishichun$ ./Observer2.out
粉丝：张三关注了悠悠独山，目前粉丝数：1
粉丝：李四关注了悠悠独山，目前粉丝数：2
悠悠独山发布了文章：文章1，欢迎查阅，目前共发布了：1 篇文章！
粉丝：张三查阅了文章：文章1
粉丝：李四查阅了文章：文章1
文章反响不错，应该会涨粉，好期待
粉丝：王麻子关注了悠悠独山，目前粉丝数：3
粉丝：丫头关注了悠悠独山，目前粉丝数：4
悠悠独山发布了文章：文章2，欢迎查阅，目前共发布了：2 篇文章！
粉丝：张三查阅了文章：文章2
粉丝：李四查阅了文章：文章2
粉丝：王麻子查阅了文章：文章2
粉丝：丫头查阅了文章：文章2
粉丝：李四取关了悠悠独山，目前粉丝数：3
粉丝：小吴并未关注悠悠独山，无法取关
粉丝：张三取关了悠悠独山，目前粉丝数：2
文章质量不好吗？怎么掉粉了，下次要注重文章质量
悠悠独山发布了文章：文章3，欢迎查阅，目前共发布了：3 篇文章！
粉丝：王麻子查阅了文章：文章3
粉丝：丫头查阅了文章：文章3
粉丝：张三关注了悠悠独山，目前粉丝数：3
粉丝：李四关注了悠悠独山，目前粉丝数：4
粉丝：小春关注了悠悠独山，目前粉丝数：5
粉丝：小赵关注了悠悠独山，目前粉丝数：6
粉丝：小钱关注了悠悠独山，目前粉丝数：7
粉丝：小孙关注了悠悠独山，目前粉丝数：8
粉丝：小周关注了悠悠独山，目前粉丝数：9
粉丝：小吴关注了悠悠独山，目前粉丝数：10
看来上一篇文章质量很好，涨了这么多粉，再来一篇
悠悠独山发布了文章：文章4，欢迎查阅，目前共发布了：4 篇文章！
粉丝：王麻子查阅了文章：文章4
粉丝：丫头查阅了文章：文章4
粉丝：张三查阅了文章：文章4
粉丝：李四查阅了文章：文章4
粉丝：小春查阅了文章：文章4
粉丝：小赵查阅了文章：文章4
粉丝：小钱查阅了文章：文章4
粉丝：小孙查阅了文章：文章4
粉丝：小周查阅了文章：文章4
粉丝：小吴查阅了文章：文章4
192:DesignPattnsStudy weishichun$
```