操作系统

实验报告

实验名称： 进程的创建实验

院系： 计算机学院·网络空间安全学院

班级：

学号：

姓名：

指导老师：匡林爱

日期：2020年09月29日

1. **实验目的和要求：**

本实验要求设计一个实现进程创建的程序，使所创建的进程在处理机上并发执行，要求程序能对出现的异常进行报告。通过本上机实验，学会在Win32程序中使用操作系统提供的API创建进程，并通过所创建的进程的运行情况获得对于进程并发的感性认识。

1. **实验内容：**
2. **根据实验要求运行相关C++代码创建进程**
3. **通过实验学会在Win32程序中使用系统提供的API创建进程**
4. **了解并掌握程序所用相关Win32API函数**
5. **学会运行实验提供的程序代码，并且分析结果**

**相关Win32 Api函数介绍：**

1. **GetLastError( )函数//返回最近一次发生的错误代码。**
2. **FormatMessage( )函数//生成具体的出错信息。**
3. **CreateProcess( )函数//创建一个新进程和它的主线程，这个新进程运行指定的可执行文件。**
4. **GetForegroundWindow( )函数//获取当前活动窗口的句柄。**
5. **MoveWindow( )函数//改变指定窗口的位置和大小。**
6. **实验步骤：**

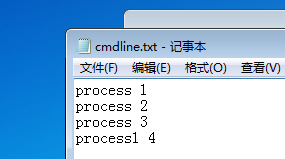
程序将“cmdline.txt”文件作为输入，该文本文件的每一行都是有一个应用程序名以及该应用程序的参数构成(应用程序名和参数由空格隔开)。

程序从“cmdline.txt”文件中依次读出每一行存入cmdline字符串中，再以cmdline为函数实参数，调用NewProc( )函数，通过CreateProcess( )这个Win32 API函数来建立一个新进程，在该进程中运行对应的应用程序。

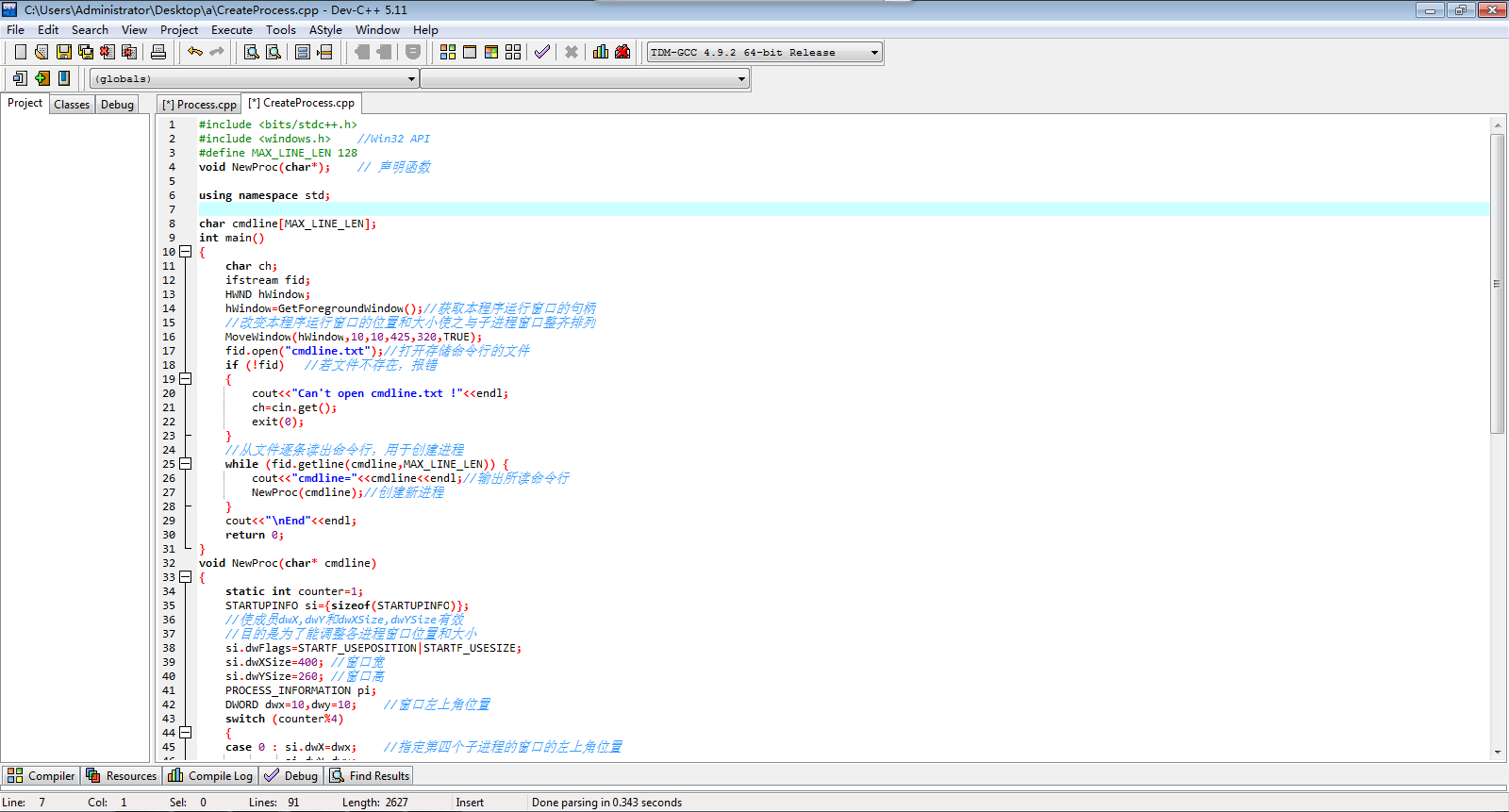
“cmdline.txt”文件是在记事本中预先编制好的一个文本文本。命令行“Process 1”中的“Process”是指当前目录下的可执行文件“Process.exe”，命令行中的“1”是该可执行文件的输入参数。该命令行也可写成“Process.exe 1”。

Process.exe是为了测试进程的并发执行而预先编制的C++源程序编译而成的，其源代码如下面介绍。上述4个命令行中，第4个命令是为了测试出错信息的，其错误原因是应用程序Process1.exe不存在。

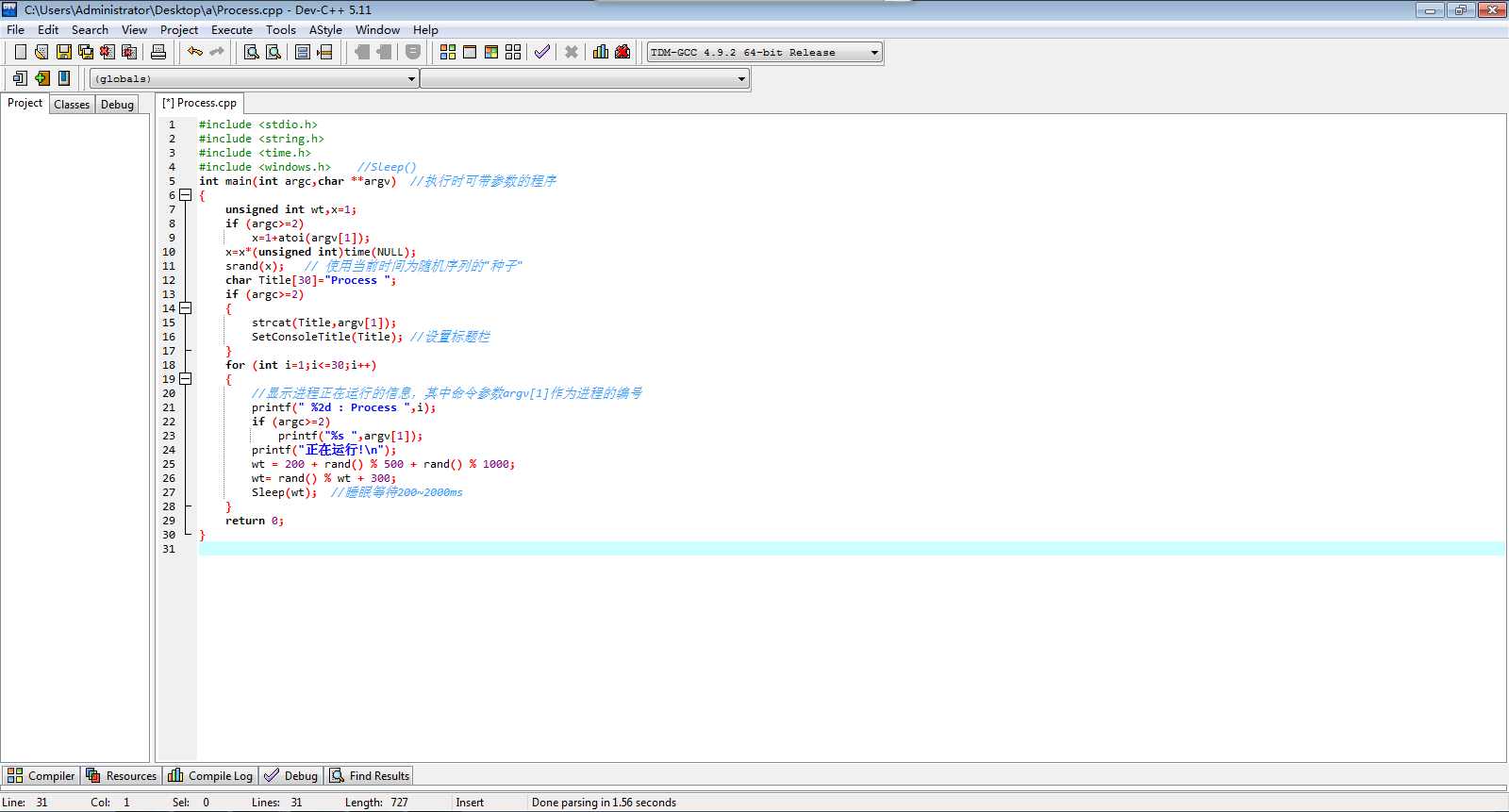
1. 编写“cmdline.txt”，内容如下：



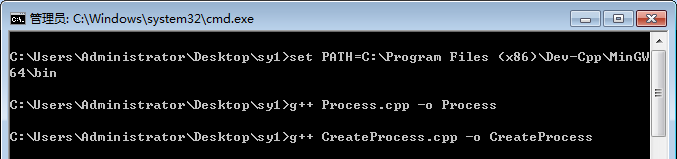
1. 编写“CreateProcess.cpp”，内容如下：



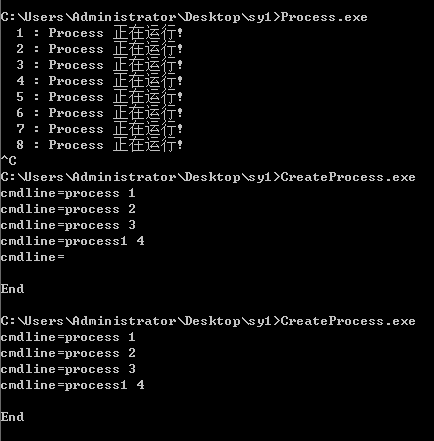
1. 编写“Process.cpp”，内容如下：

****

1. 设置临时环境变量，编译文件：



1. 运行测试



1. **实验源程序**

**Process.cpp：**

**#include <stdio.h>**

**#include <string.h>**

**#include <time.h>**

**#include <windows.h> //Sleep()**

**int main(int argc,char \*\*argv) //执行时可带参数的程序**

**{**

**unsigned int wt,x=1;**

**if (argc>=2)**

**x=1+atoi(argv[1]);**

**x=x\*(unsigned int)time(NULL);**

**srand(x); // 使用当前时间为随机序列的"种子"**

**char Title[30]="Process ";**

**if (argc>=2)**

**{**

**strcat(Title,argv[1]);**

**SetConsoleTitle(Title); //设置标题栏**

**}**

**for (int i=1;i<=30;i++)**

**{**

**//显示进程正在运行的信息，其中命令参数argv[1]作为进程的编号**

**printf(" %2d : Process ",i);**

**if (argc>=2)**

**printf("%s ",argv[1]);**

**printf("正在运行!\n");**

**wt = 200 + rand() % 500 + rand() % 1000;**

**wt= rand() % wt + 300;**

**Sleep(wt); //睡眠等待200~2000ms**

**}**

**return 0;**

**}**

**CreateProcess.cpp：**

**#include <bits/stdc++.h>**

**#include <windows.h> //Win32 API**

**#define MAX\_LINE\_LEN 128**

**void NewProc(char\*); // 声明函数**

**using namespace std;**

**char cmdline[MAX\_LINE\_LEN];**

**int main()**

**{**

**char ch;**

**ifstream fid;**

**HWND hWindow;**

**hWindow=GetForegroundWindow();//获取本程序运行窗口的句柄**

**//改变本程序运行窗口的位置和大小使之与子进程窗口整齐排列**

**MoveWindow(hWindow,10,10,425,320,TRUE);**

**fid.open("cmdline.txt");//打开存储命令行的文件**

**if (!fid) //若文件不存在，报错**

**{**

**cout<<"Can't open cmdline.txt !"<<endl;**

**ch=cin.get();**

**exit(0);**

**}**

**//从文件逐条读出命令行，用于创建进程**

**while (fid.getline(cmdline,MAX\_LINE\_LEN)) {**

**cout<<"cmdline="<<cmdline<<endl;//输出所读命令行**

**NewProc(cmdline);//创建新进程**

**}**

**cout<<"\nEnd"<<endl;**

**return 0;**

**}**

**void NewProc(char\* cmdline)**

**{**

**static int counter=1;**

**STARTUPINFO si={sizeof(STARTUPINFO)};**

**//使成员dwX,dwY和dwXSize,dwYSize有效**

**//目的是为了能调整各进程窗口位置和大小**

**si.dwFlags=STARTF\_USEPOSITION|STARTF\_USESIZE;**

**si.dwXSize=400; //窗口宽**

**si.dwYSize=260; //窗口高**

**PROCESS\_INFORMATION pi;**

**DWORD dwx=10,dwy=10; //窗口左上角位置**

**switch (counter%4)**

**{**

**case 0 : si.dwX=dwx; //指定第四个子进程的窗口的左上角位置**

**si.dwY=dwy;**

**break;**

**case 1 : si.dwX=dwx+425;//指定第一个子进程的窗口的左上角位置**

**si.dwY=dwy;**

**break;**

**case 2 : si.dwX=dwx; //指定第二个子进程的窗口的左上角位置**

**si.dwY=dwy+310;**

**break;**

**case 3 : si.dwX=dwx+425;//指定第三个子进程的窗口的左上角位置**

**si.dwY=dwy+310;**

**break;**

**}**

**counter++;**

**BOOL result=CreateProcess(**

**NULL, //没指定可执行模块名**

**cmdline, //路径和可执行模块名**

**NULL, //子进程不能继承此函数返回的句柄**

**NULL, //子进程的线程不能继承此函数返回的句柄**

**FALSE, //新进程不能从调用此函数进程继承此句柄**

**NORMAL\_PRIORITY\_CLASS|CREATE\_NEW\_CONSOLE,**

**//创建一个没有特殊调度需求的普通进程、开辟新窗口**

**NULL, //新进程使用调用此函数进程(父进程)的环境块**

**NULL, //新进程使用父进程的当前目录作为自己的当前目录**

**&si, //STARTUPINFO结构决定新进程主窗口如何出现(显示)**

**&pi //PROCESS\_INFORMATION结构用于接收新进程的标识信息**

**);**

**//显示详细的创建进程出错信息**

**if (GetLastError()) {**

**LPVOID lpMsgBuf;**

**FormatMessage(**

**FORMAT\_MESSAGE\_ALLOCATE\_BUFFER |**

**FORMAT\_MESSAGE\_FROM\_SYSTEM |**

**FORMAT\_MESSAGE\_IGNORE\_INSERTS,**

**NULL,**

**GetLastError(),**

**MAKELANGID(LANG\_NEUTRAL,SUBLANG\_DEFAULT),**

**(LPSTR) &lpMsgBuf,**

**0,**

**NULL**

**);**

**//弹出出错窗口指示错误(该信息由FormatMessage存入了lpMsgBuf中)**

**MessageBox(NULL,(LPSTR)lpMsgBuf,"Error",MB\_OK | MB\_ICONINFORMATION);**

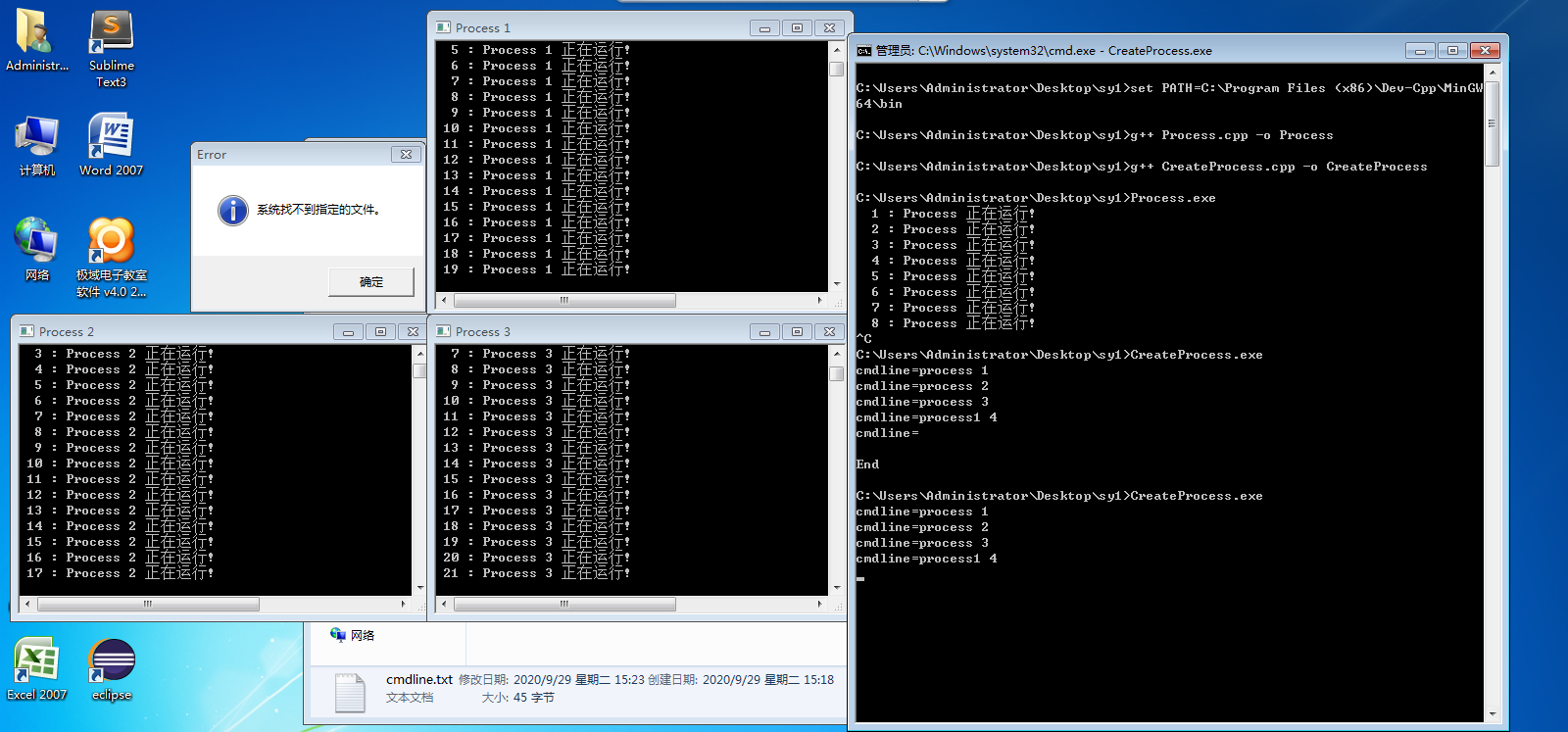
**LocalFree(lpMsgBuf);**

**//释放在FormatMessage中动态分配的空间**

**}**

**}**

1. **实验结果**

****

1. **实验总结：**

通过这次上机实验，我学会了在Win32程序中使用操作系统提供的API创建进程，并且对所创建的进程的运行情况有了一个感性的认识。同时，我对命令行编译程序的运用，也有了一些自己的心得和领悟。当然，我也了解并接触了许多以前从来没有接触到的Win32 API函数，也学会了他们的用法和调用方式，对于程序运行中的报错处理也有了自己的体悟，而且我在代码方面的编写能力也有了一定程度的提高，总而言之，这次实验带给我的收获很大，下一次的实验，我也要全力以赴，好好学知识！