1. **实验目的及内容**

设计并实现类似Unix的“time”命令：“mytime”。“mytime”命令通过命令行参数接收要运行的程序，创建一个独立的进程来运行该程序，并记录该程序运行的时间。

实现Windows版本和Linux版本。

**二、实验环境**

操作系统：windows10 64位

编译器：Visual Studio 2012

Linux Ubuntu 14.04.5

编译器：gcc

1. **基本原理及方法概述**

**2.1 windows实现**

使用CreateProcess()来创建进程；

使用WaitForSingleObject()在父进程命令和子进程之间同步；

调用GetSystemTime()来获取时间；

**2.1.1CreateRrocess()函数功能及参数**

WIN32API函数CreateProcess用来创建一个新的进程和它的[主线程](https://baike.baidu.com/item/%E4%B8%BB%E7%BA%BF%E7%A8%8B" \t "https://baike.baidu.com/item/CreateProcess/_blank)，这个新进程运行指定的[可执行文件](https://baike.baidu.com/item/%E5%8F%AF%E6%89%A7%E8%A1%8C%E6%96%87%E4%BB%B6" \t "https://baike.baidu.com/item/CreateProcess/_blank)。返回值为布尔型。

BOOL CreateProcess(

LPCTSTR lpApplicationName, //指向可执行的文件名的指针

LPTSTR lpCommandLine, //指向可执行命令行字符串的指针

LPSECURITY\_ATTRIBUTES lpProcessAttributes, //指向进程安全属性的指针

LPSECURITY\_ATTRIBUTES lpThreadAttributes, //指向线程程安全属性的指针

BOOL bInheritHandles, //被创建的进程是否继承创建进程的句柄

DWORD dwCreationFlags, //创建标志

LPVOID lpEnvironment, //指向新创建的环境块的指针

LPCTSTR lpCurrentDirectory, //指向当前工作目录的指针

LPSTARTUPINFO lpStartupInfo, //指向进程启动信息结构体的指针

[LPPROCESS\_INFORMATION](https://baike.baidu.com/item/LPPROCESS_INFORMATION" \t "https://baike.baidu.com/item/CreateProcess/_blank)lpProcessInformation ,//指向进程信息结构体的指针

);

**2.1.2 WaitForSingleObject()函数功能及参数**

WaitForSingleObject函数用来检测hHandle事件的信号状态，在某一进程程中调用该函数时，进程暂时挂起，如果在挂起的dwMilliseconds毫秒内，进程所等待的对象变为有信号状态，则该函数立即返回；如果超时时间已经到达dwMilliseconds毫秒，但hHandle所指向的对象还没有变成有信号状态，函数照样返回。

参数dwMilliseconds有两个具有特殊意义的值：0和INFINITE。若为0，则该函数立即返回；若为INFINITE，则进程程一直被挂起，直到hHandle所指向的对象变为有信号状态时为止。

本实验dwMilliseconds的值为INFINITE，即父进程阻塞等待新创建的进程完成后父进程才结束。

DWORD WaitForSingleObject(  
 HANDLE hObject, //指明一个内核对象的句柄  
 DWORD dwMilliseconds //等待时间

);

**2.1.3 GetSystemTime()函数功能及参数**

在一个SYSTEMTIME数据结构中载入当前系统时间。函数的参数为SYSTEMTIME结构的地址。

SYSTEMTIME STRUCT

{

WORD wYear ; 年

WORD wMonth ;月

WORD wDayOfWeek ;星期，0=星期日，1=星期一...

WORD wDay ;日

WORD wHour ;时

WORD wMinute ;分

WORD wSecond ;秒

WORD wMilliseconds ;毫秒

}SYSTEMTIME ENDS

**2.1.4 其他的系统调用**

PROCESS\_INFORMATION pi; //该结构返回有关新进程及其主线程的信息

STARTUPINFO si; //用于子进程的 STARTUPINFO结构

**2.2 linux实现**

使用fork()/execv()来创建进程运行程序

使用wait()等待新创建的进程结束

调用gettimeofday()来获取时间

fork() 若成功调用一次则返回两个值，子进程返回0，[父进程](https://baike.baidu.com/item/%E7%88%B6%E8%BF%9B%E7%A8%8B" \t "https://baike.baidu.com/item/fork/_blank)返回子进程标记；否则，出错返回-1。

execv(const char \*path, char \*const argv[]);

path:被执行的应用程序。

argv: 传递给应用程序的参数列表，这个数组的第一个参数应该是应用程序名字本身，并且最后一个参数应该为NULL，不参将多个参数合并为一个参数放入数组。

wait()函数一般用在父进程中等待回收子进程的资源。只要有一个进程终止，wait就会返回。也就是说只要wait接收到一个SIGCHLD信号,wait()就会返回。对于两个或多个子进程的情况,需要调用wait两次或多次。

gettimeofday()获得当前精确时间，或者为执行计时，可以使用gettimeofday()函数。获取的时间需存放在timeval结构中。

获取秒和毫秒的示例：

start.tv\_sec\*1000000+start.tv\_usec;

1. **程序流程图**

Windows linux

开始

开始

pid\_t pid=fork()创建进程

CreateProcess创建一个新进程

GetSystemTime()获取子进程开始时间

gettimeofday()获取子进程开始时间

结束

计算子进程运行时间并输出

gettimeofday()获取子进程结束时间

wait（）等待子进程结束

execv()运行子进程的程序

结束

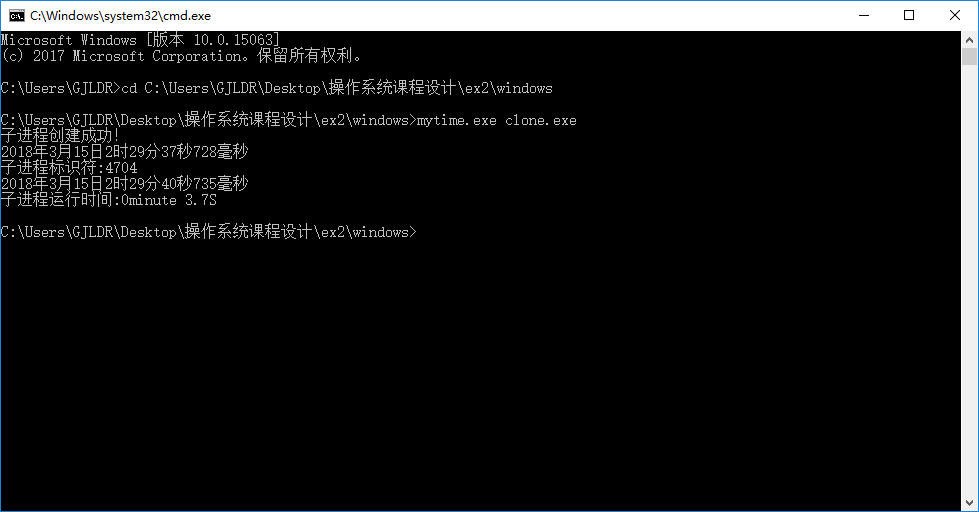
计算子进程运行时间并输出

GetSystemTime()获取子进程结束时间

WaitForSingleObject（）等待子进程结束

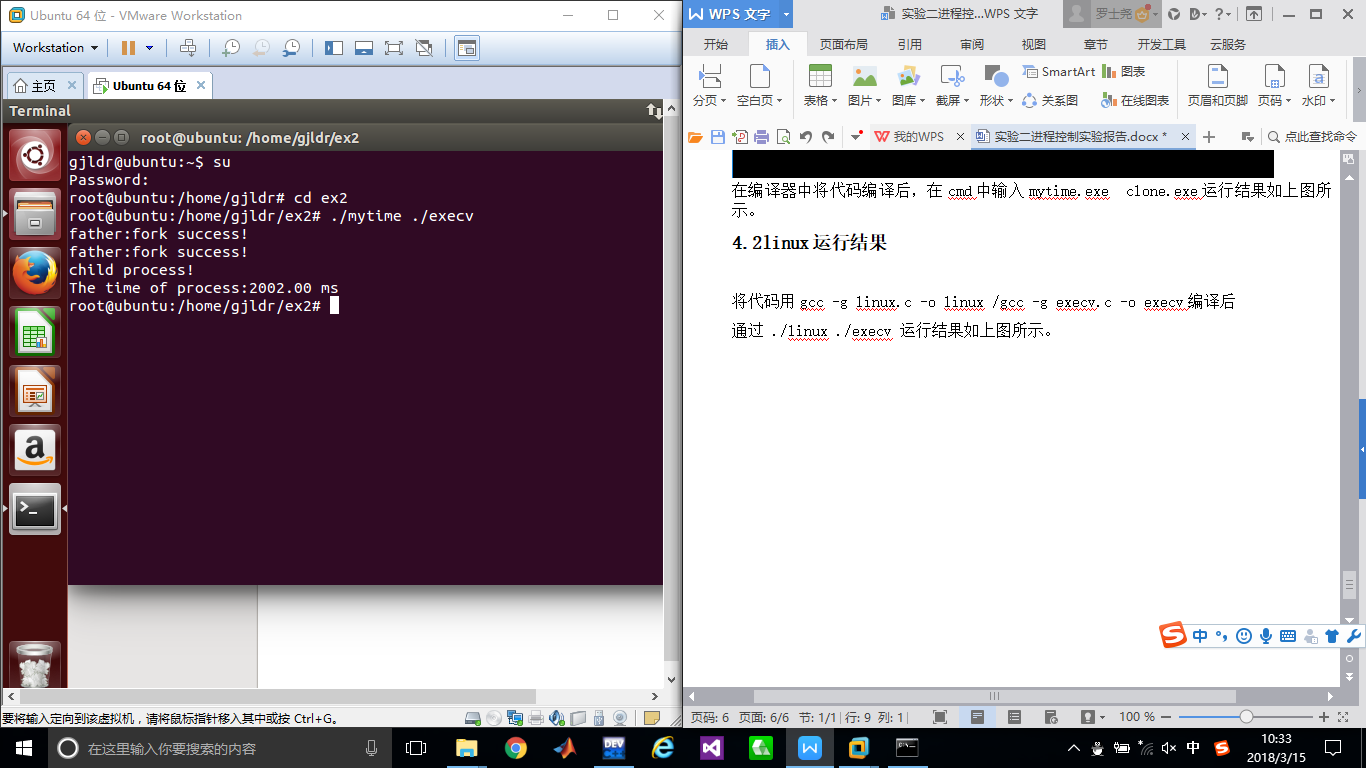
1. **程序运行结果及分析**

**4.1 windows运行结果**



在编译器中将代码编译后，在cmd中输入mytime.exe clone.exe运行结果如上图所示。

**4.2linux运行结果**



将代码用gcc -g mytime.c -o mytime /gcc -g execv.c -o execv编译后

通过 ./mytime ./execv 运行结果如上图所示。