**操作系统课程设计实验报告**

1、Windows系统实现：

（1）实验思路：用GetSystemTime获取开始时间，结束时间，然后用差值来计算程序的运行时间。使用CreateProcess函数创建一个进程，当实现mytime的第一种用法，输入是两个参数的时候就使用WaitForSingleObject(pi.hProcess, INFINITE)等待子程序运行结束，计算其运行时间；当实现mytime第二种用法，输入是三个参数的时候，将控制时间作为第三个参数输入进去，当程序运行完成后，显示其运行时间。

（2）mytime程序设计：

#include <stdio.h>

#include <windows.h>

int main(int argc, char \*argv[])

{

    SYSTEMTIME start, end;

    TCHAR szFilename[MAX\_PATH];

    TCHAR szCmdLine[MAX\_PATH];

    GetModuleFileName(NULL, szFilename, MAX\_PATH);

    STARTUPINFO si;

    ZeroMemory(reinterpret\_cast<void \*>(&si), sizeof(si));

    si.cb = sizeof(si);

    PROCESS\_INFORMATION pi;

    BOOL bCreateOK = CreateProcess(NULL, szCmdLine, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi);

    if (!bCreateOK)

    {

        printf("create failed\n");

        exit(1);

    }

    GetSystemTime(&start);

    printf("create succeed!\n");

    WaitForSingleObject(pi.hProcess, INFINITE);

    GetSystemTime(&end);

    int ms = end.wMilliseconds - start.wMilliseconds;

    int s = end.wSecond - start.wSecond;

    int min = end.wMinute - start.wMinute;

    int h = end.wHour - start.wHour;

    printf("%d h %d min %d s %d ms 0 us\n", h, min, s, ms);

    return 0;

}

（3）program1程序设计：

#include<stdio.h>

#include <windows.h>

int main(int argc,char \*argv[])

{

    Sleep(500);

    printf("Hello world!\n");

    return 0;

}

（4）program2程序设计

#include <stdio.h>

#include <stdlib.h>

#include <windows.h>

int main(int argc, char \*argv[])

{

    printf("Run program2\n");

    printf("%s\n", argv[2]);

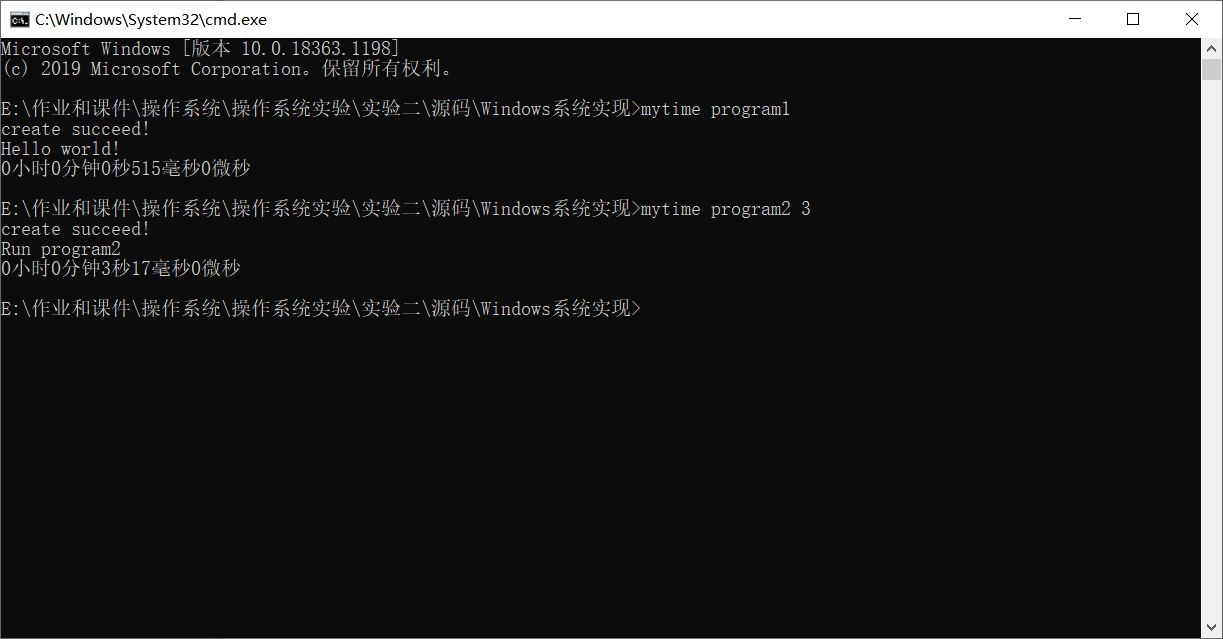
    int time = atoi(argv[1]);

    Sleep(1000 \* time);

    return 0;

}

1. 实验结果



2、Linux系统实现

（1）实验思路：用gettimeofday（）函数得到程序运行的开始时间、结束时间，然后用差值来计算程序的运行时间。使用fork（）函数创建一个子进程，当实现mytime的第一种用法，输入两个参数的时候就使用Wait（）函数等待子程序运行结束，计算其运行时间；当实现mytime第二种用法，输入是三个参数的时候，将控制时间作为第三个参数输入进去，当程序运行完成后，显示其运行时间。

（2）mytime程序设计

#include<sys/types.h>

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/wait.h>

#include<sys/time.h>

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

{

pid\_t pid;

struct timeval start,end;

pid=fork();

if(pid<0)

{

printf("fork() error\n");

exit(-1);

}

gettimeofday(&start,NULL);

if(pid==0)

{

//在子进程中运行另一个程序

if(execvp(argv[1],argv)<0)

{

printf("execvp() error\n");

exit(-1);

}

}

else

{

wait(0);

gettimeofday(&end,NULL);

//计算相差的总秒数和微秒数

int seconds=end.tv\_sec-start.tv\_sec;

int useconds=end.tv\_usec-start.tv\_usec;

if(useconds<0)

{

useconds+=1000000;

seconds-=1;

}

//计算时分秒毫秒微秒

int h=seconds/3600;

int min=(seconds-h\*3600)/60;

int s=seconds-h\*3600-min\*60;

int ms=useconds/1000;

int us=useconds-ms\*1000;

printf("运行时间：%d小时%d分钟%d秒%d毫秒%d微秒\n",h,min,s,ms,us);

}

return 0;

}

（3）program1程序设计

#include<stdio.h>

int main()

{

printf("Hello World!\n");

return 0;

}

（4）program2程序设计

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

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

{

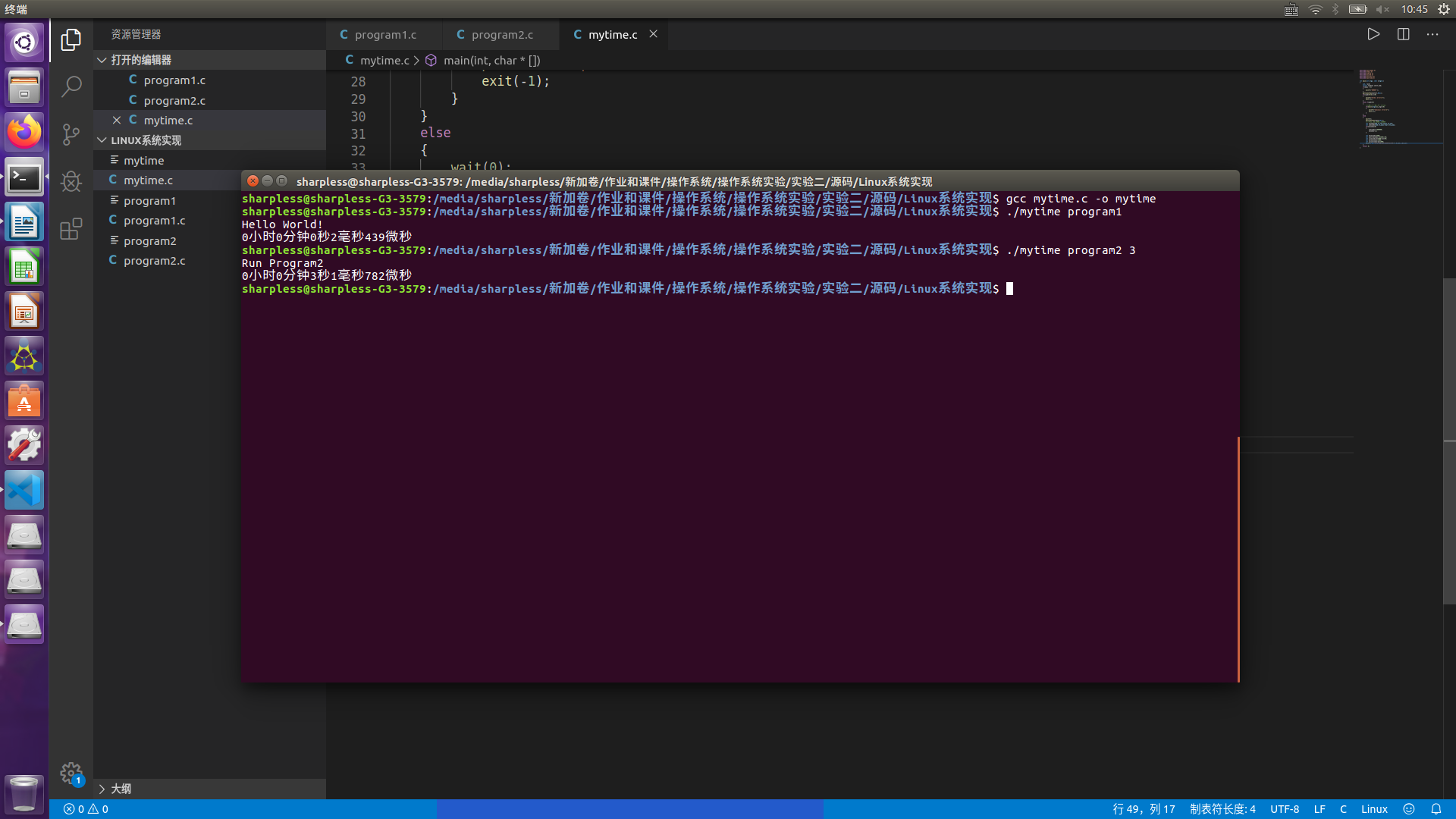
printf("Run Program2\n");

sleep(atoi(argv[2]));

return 0;

}

1. 实验结果



对于mytime的第一种用法，输入mytime program1，得到program1这个程序的运行时间：1毫秒18微秒；对于mytime的第二种用法，输入mytime program2 5，得到时间为5秒1毫秒278微秒，与输入的时间5秒基本一致，实验结果正确。

1. **实验收获与体会**

通过这个实验，我对进程控制这部分的知识有了一个更深刻的理解。在实验初期的时候觉得这个实验还挺难的，因为这些以前从未听说过的函数的调用，每一个参数都需要自己去一个一个的理解是什么意思，印象最深的两个地方是：

1、argc，argv[]数组

我通过这次试验才真正搞懂main中的这两个参数的含义，argc存放的数值是参数的个数，argv[0]存放的是程序的路径名，argv[1]存放的是DOS命令行中执行程序名后第一个字符串，argv[2]指向第二个字符串，以此类推…最后以NULL参数结尾。

2、exec等一系列函数

我在实验过程中发现这一系列在子进程中运行另一个程序的函数都是有所区别的，我仔细研究了他们的不同，在尝试了execv、execl、execle、execvp之后才最终使用了execvp以达到预想的效果。

这个实验的需要在Windows系统和Linux系统下都要实现。在设计程序的时候，感觉我们在一个系统中成功了，在另一个系统中实现的时候就会简单很多，思路都是差不多的。以后还要加深自己对进程控制的了解，对其函数的调用真正的了解清楚之后，相信写程序是水到渠成的事情。