实验二

实验环境：Windows 7/8 PC机

实验软件：VMware虚拟机，Ubuntu 15.10操作系统

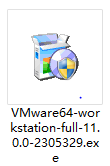
实验内容：利用多任务库进行多线程文件拷贝。

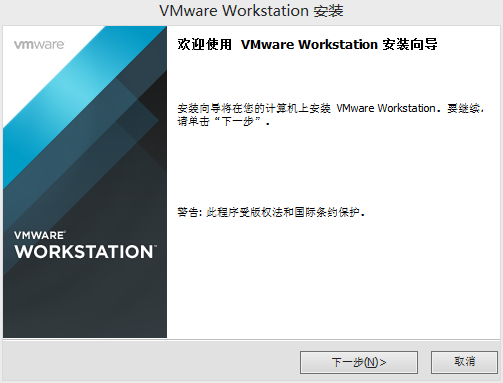
实验目的：熟练掌握Ubuntu操作系统的使用及多任务库编程

实验要求：文件内容随意，大小20kB，至少3个线程，拷贝文件内容准确无误

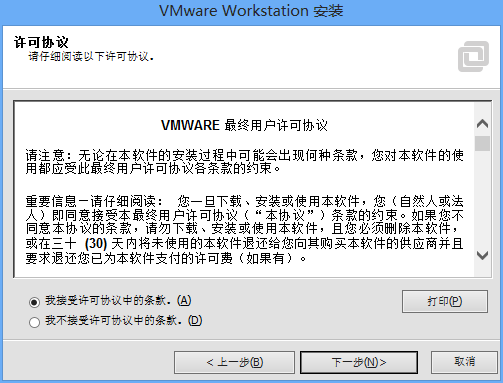
指导书前三章为实验环境搭建，及hello world程序用例演示，第四章为Linux系统终端常用操作命令，第五章为本实验用到的库函数

###### 一、VMware软件安装

根据操作系统选择安装包，双击见下图



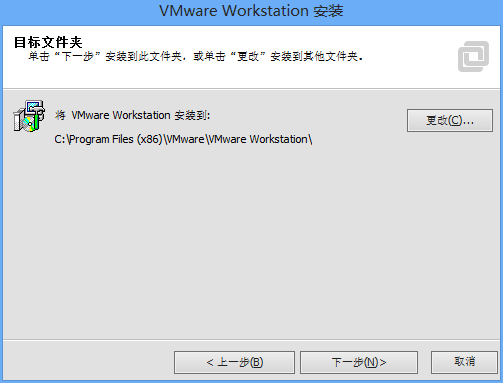
点击下一步，见下图



点击下一步，见下图

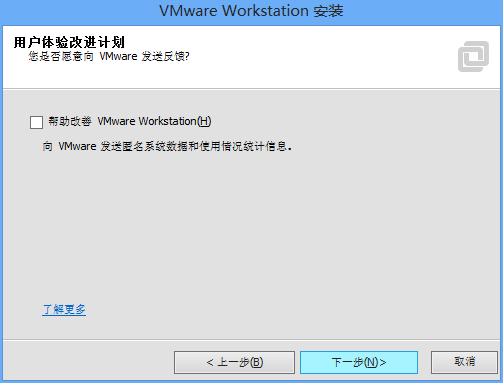


点击“典型”后，见下图

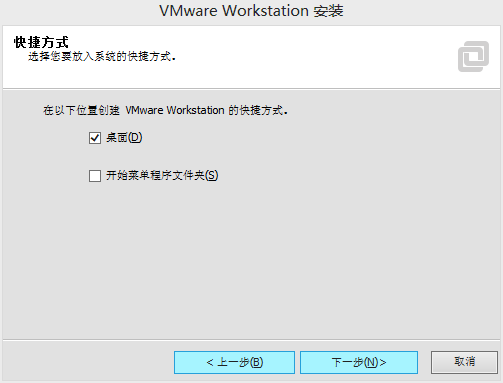


选择VMware软件安装路径，点击下一步，见下图

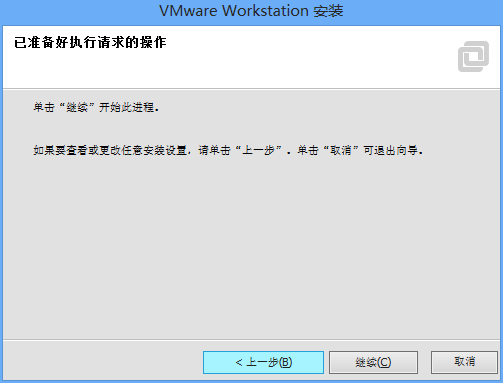
  
点击下一步，见下图



点击下一步，见下图



点击下一步，见下图



点击继续，等待安装，下图为VMware各版本秘钥，64位操作系统用的是v11版本，根据需要选择秘钥。通过后则VMware软件安装完成

VMware Workstation v12 for Windows

5A02H-AU243-TZJ49-GTC7K-3C61N

VMware Workstation v11 for Windows

1F04Z-6D111-7Z029-AV0Q4-3AEH8

VMware Workstation v10 for Windows

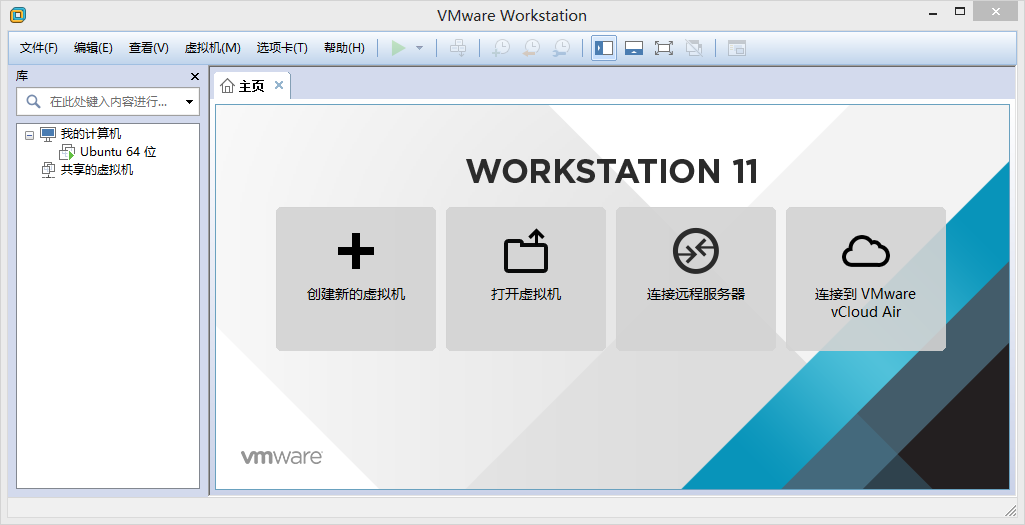
1Z0G9-67285-FZG78-ZL3Q2-234JG

4C4EK-89KDL-5ZFP9-1LA5P-2A0J0

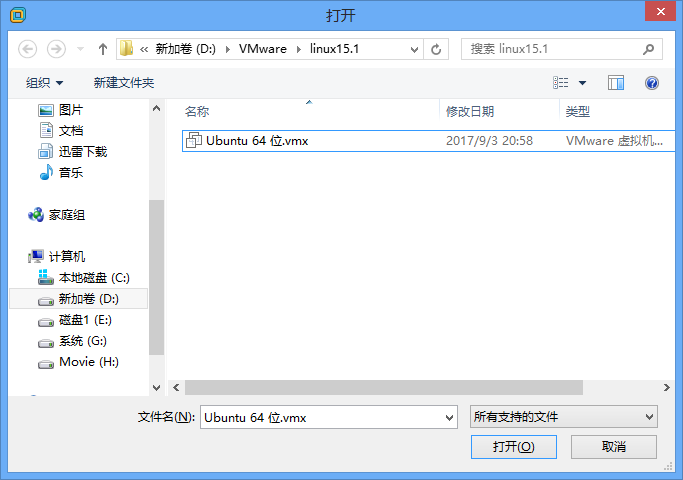
HY086-4T01N-CZ3U0-CV0QM-13DNU

###### ****二、Linux虚拟机安装****

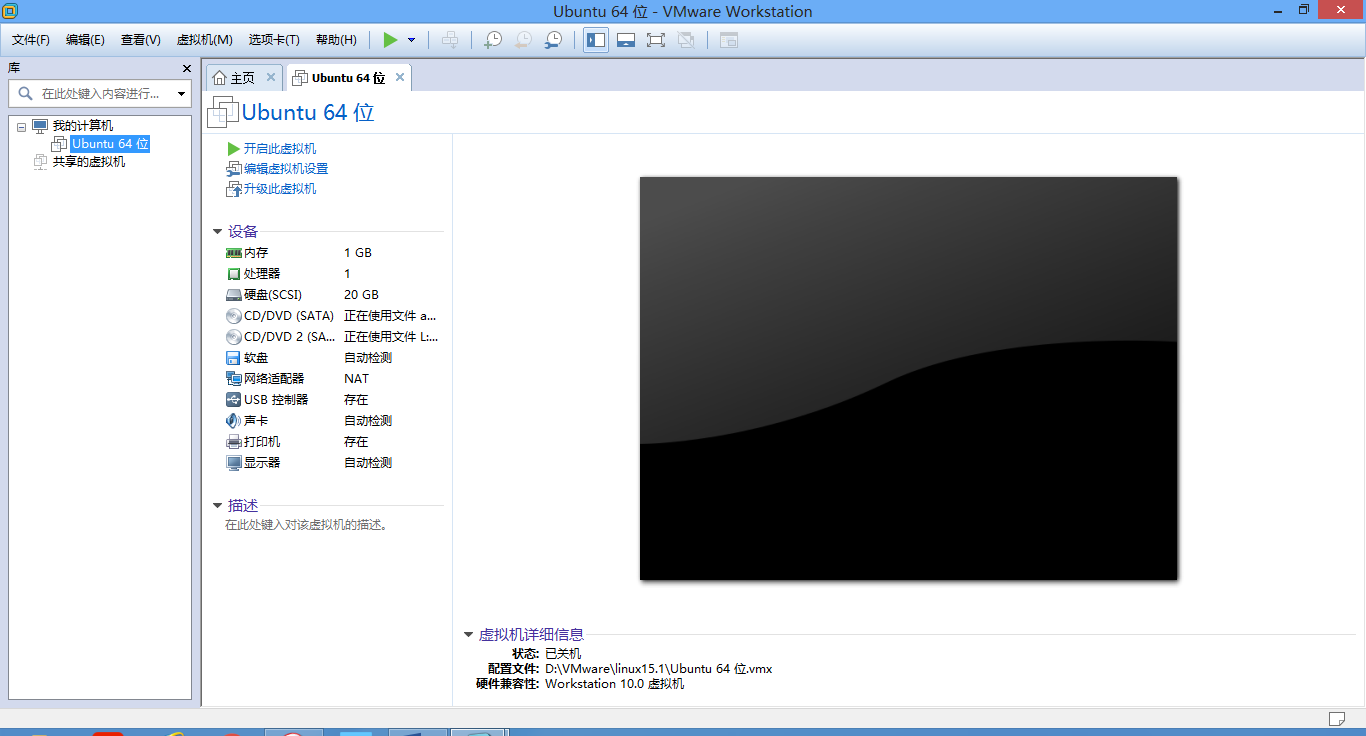
桌面打开，见下图



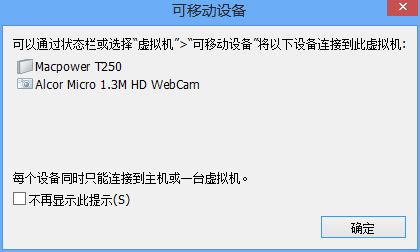
选择打开虚拟机，选择Linux路径



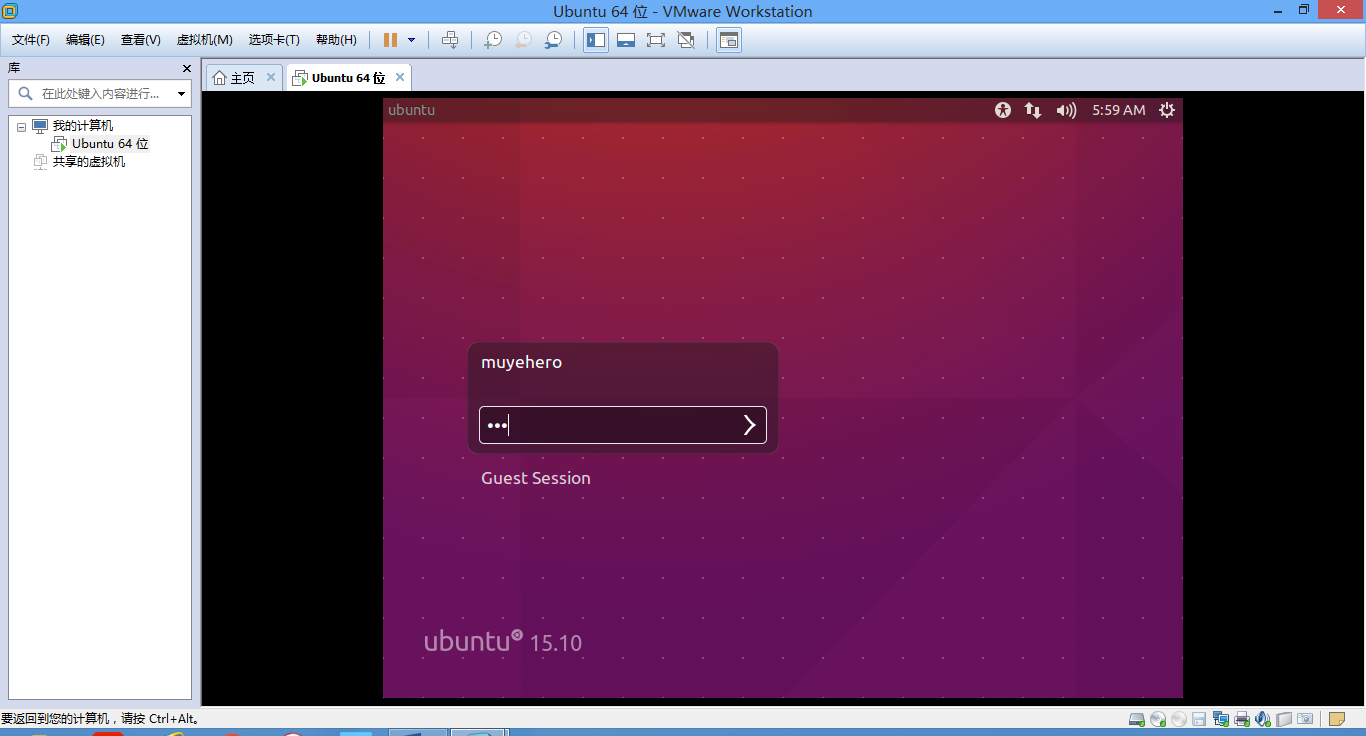
点击打开（如果没有反应，关闭VMware软件，以管理员身份重新打开，重复上述步骤）见下图



选择开启此虚拟机



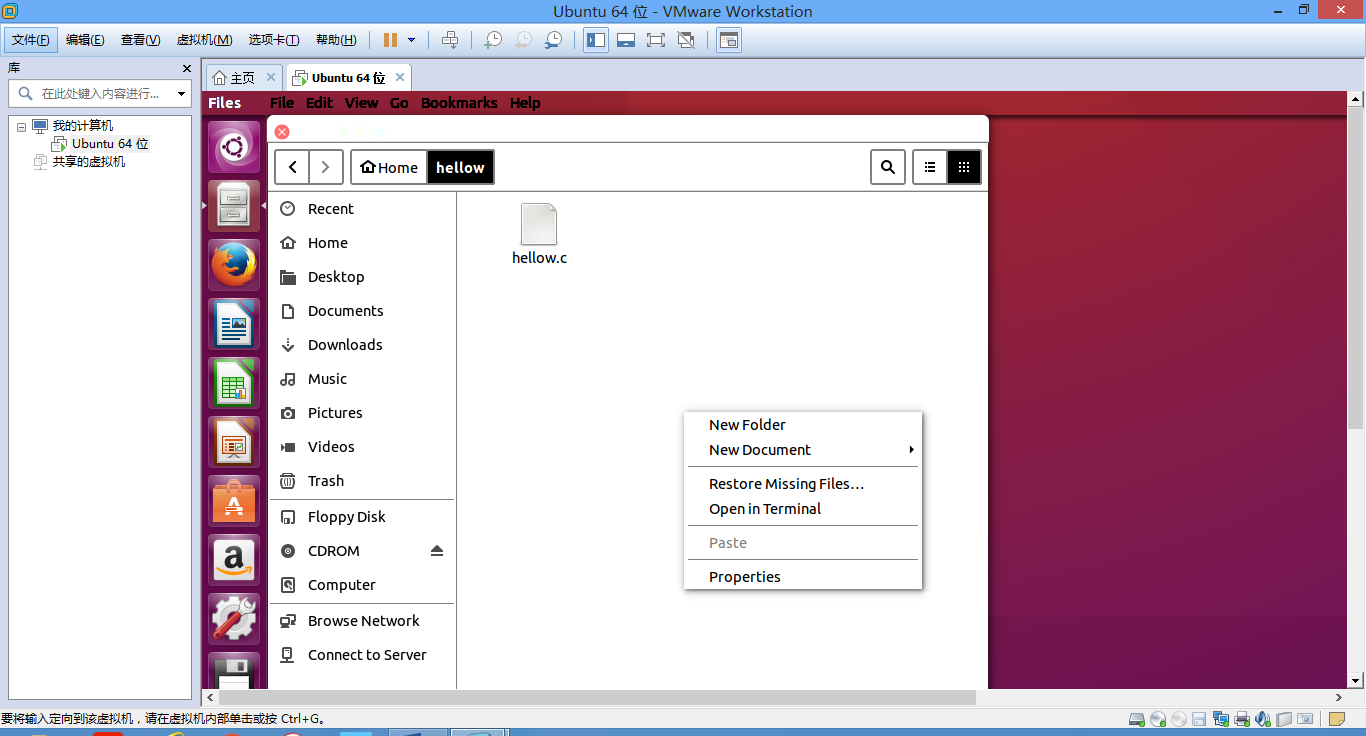
点击确定



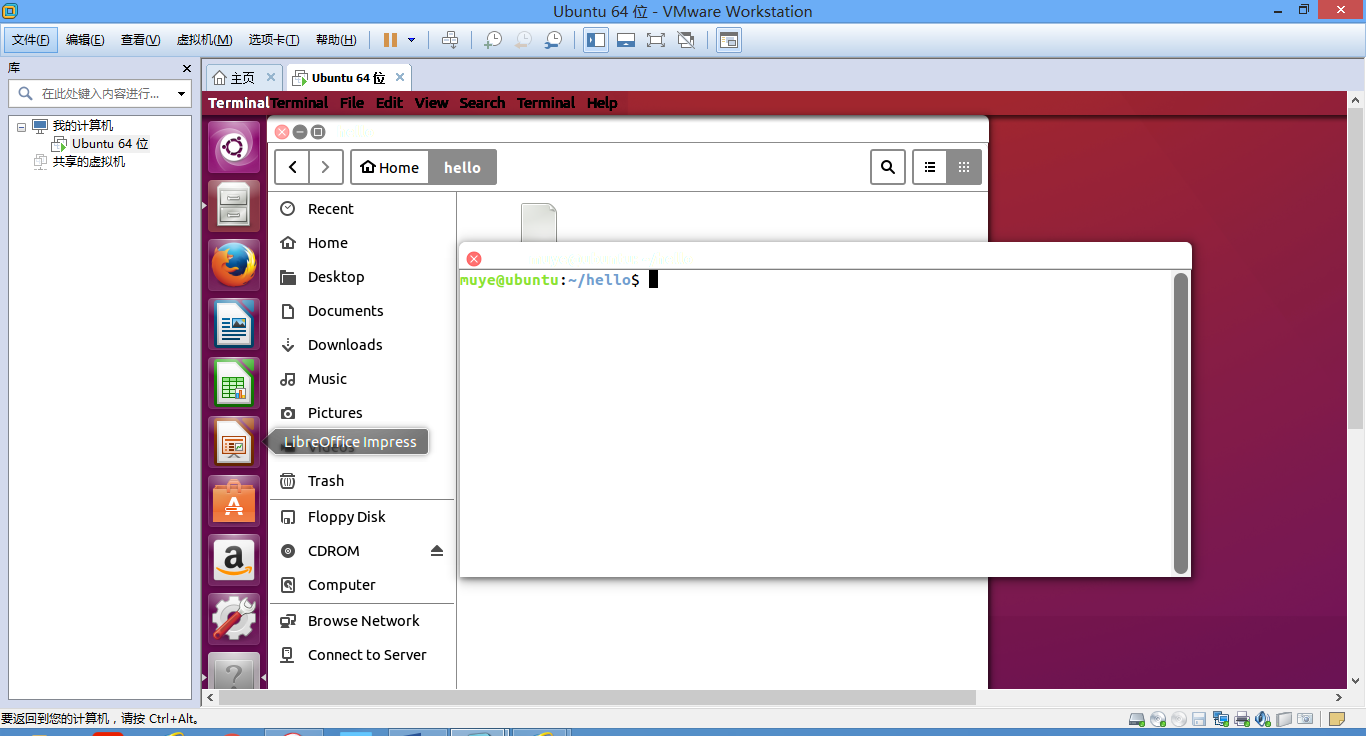
输入密码123，该操作系统为Ubuntu 15，至此Linux虚拟机安装完成。

###### 三、编译运行hello world

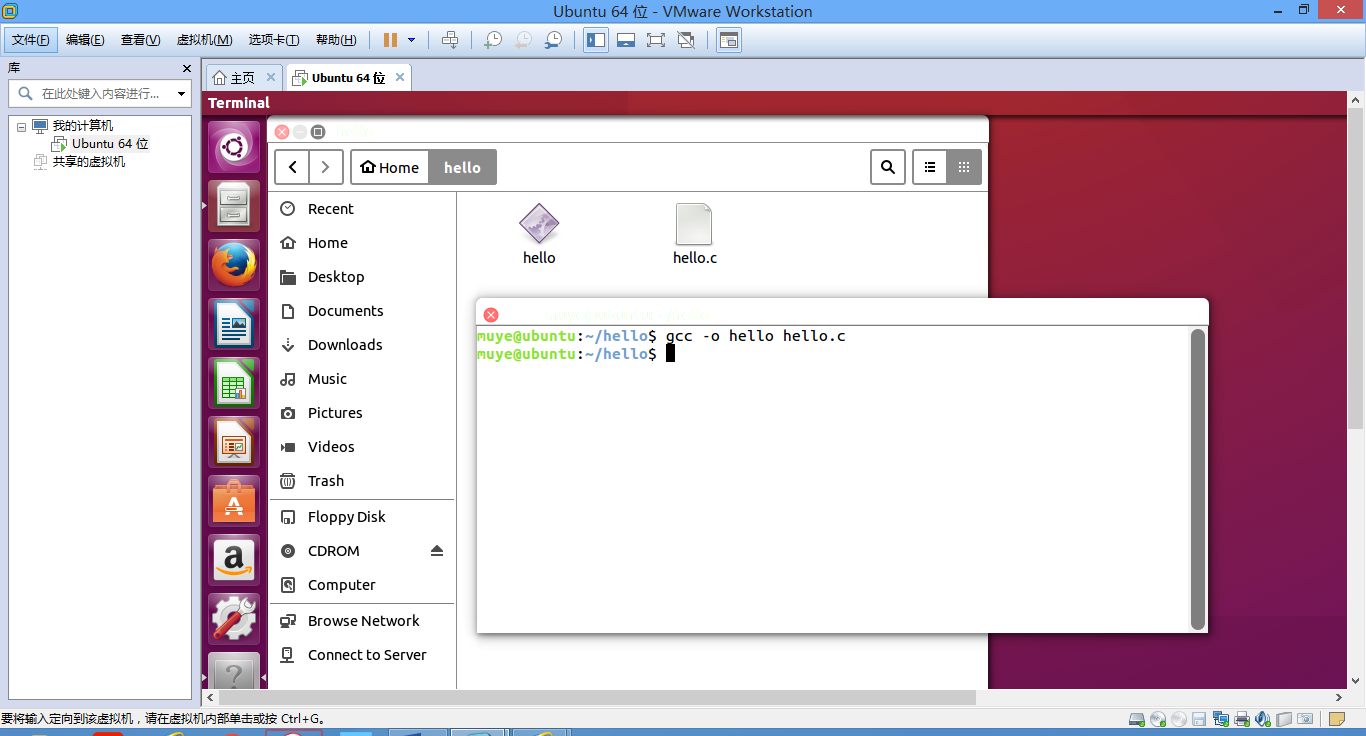
建立hello.c文件，打开超级终端，在其所在文件夹内右键选择Open in Terminal



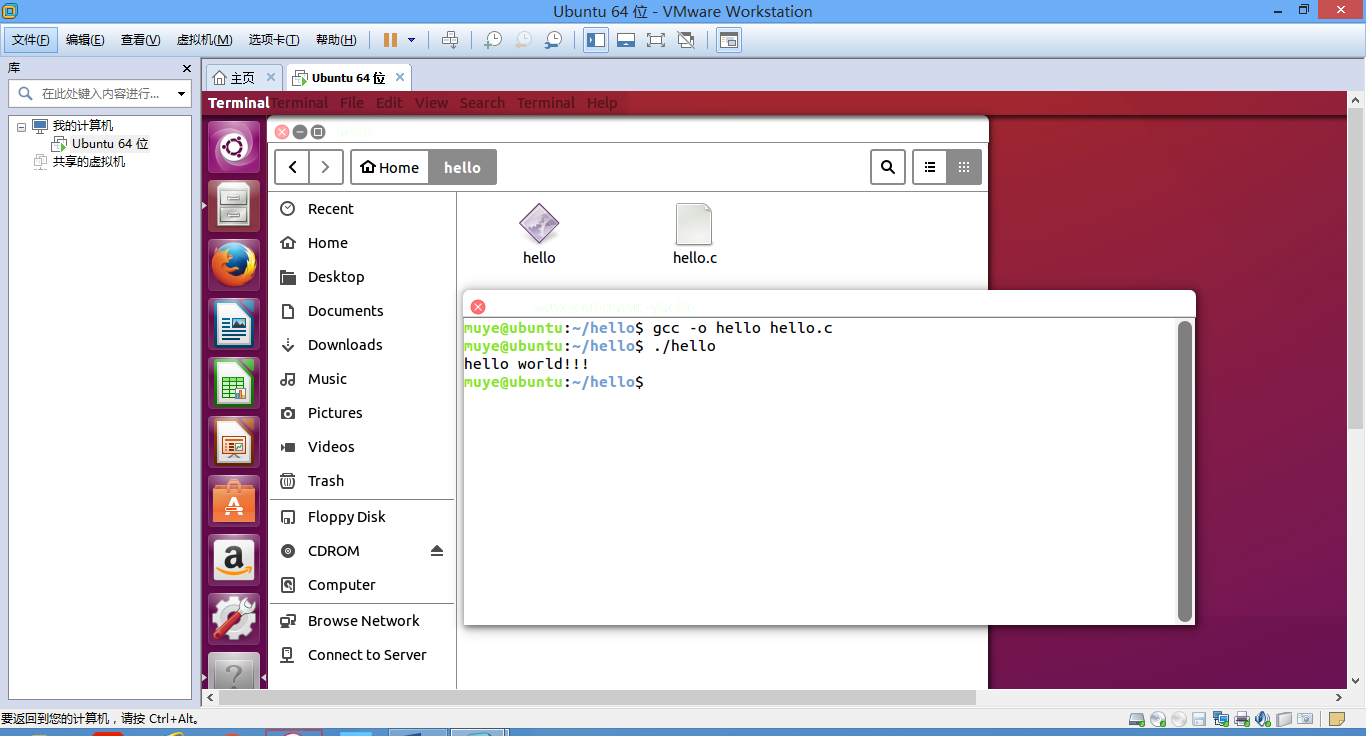
打开超级终端，在其所在文件夹内右键选择Open in Terminal



编译写好的hello.c文件，在终端内输入gcc -o hello hello.c



运行编译生成的hello程序，在终端内输入./hello



结果如上图所示，至此成功编译运行hello world程序。

###### 四、常用命令

根据命令使用类别的不同分为了9个大类，都在下面一一列举了出来，

* 文件目录类

1.建立目录：mkdir 目录名

2.删除空目录：rmdir 目录名

3.无条件删除子目录： rm -rf 目录名

4.改变当前目录：cd 目录名 (进入用户home目录：cd ~;进入上一级目录：cd -)

5.查看自己所在目录：pwd

6.查看当前目录大小：du

7.显示目录文件列表：ls -l (-a：增加显示隐含目录)

其中：蓝：目录;绿：可执行文件;红：压缩文件;浅蓝：链接文件;灰：其他文件;红底白字：错误的链接文件

8.浏览文件：more 文件名.txt;less 文件名.txt

9.复制文件： cp 源文件 目标文件 (-r：包含目录)

10.查找文件：(1)find (2)locate 命令名

11.链接：(1)建立hard链接：ln 来源文件 链接文件(-d：创建目录链接）；(2)建立符号链接：ln -s 来源文件 链接文件

* 驱动挂载类

1.检查硬盘使用情况：df -T -h

2.检查磁盘分区：fdisk -l

3.挂载软硬光区：mount -t /dev/fdx|hdax /mnt/目录名

其中：：modos--FAT16;vfat--FAT32;ntfs--NTFS;光驱--iso9660

支持中文名：mount -o iocharset=x /dev/hdax /mnt/目录名(其中：x=cp936或

挂载光驱：mount -t auto /dev/cdrom /mnt/cdrom

挂载ISO文件：mount -t iso9660 -o loop xxx.iso /path

4.解除挂载：umount /mnt/目录名

解除所有挂载：umount -a

5.建立文件系统：mkfs -t /dev/hdxx。其中：ftype：ext2、ext3、swap等

* 程序安装类

1.RPM包安装：(1)安装 rpm -ivh somesoft.rpm

(2)反安装 rpm -e somefost.rpm

(3)查询 rpm -q somefost 或 rpm -qpi somefost.rpm(其中：p未安装;i包含的信息)

(4)查询安装后位置：rpm -ql somefost.rpm

(5)升级安装：rpm -Uvh somesoft.rpm

(6)强制安装：rpm -ivh --nodeps somesoft.rpm 或 rpm -ivh --nodeps --force somesoft.rpm

* 源代码包安装：

查阅README

基本用法

(1)配置：解压目录下 ./configure

(2)编译：解压目录下 make

(3)安装：解压目录下 make install

3.src.rpm的安装

需要用到rpmbuild命令加上--rebuild参数。如 rpmbuild --rebuild \*\*\*.src.rpm。然后在/usr/src/下找

3.FC3下iso程序安装：system-config-packages --isodir=iso所在目录

RH下iso程序安装：redhat-config-packages --isodir=iso所在目录

* 压缩解压类

1.tar.gz类：(1)解压：tar -xvzf 文件.tar.gz；(2)tar.gz解至tar：gzip -d 文件.tar.gz(2)压缩：gzip 待压缩文件

2.tar未压缩类：(1)解包：tar -xvf 文件.tar；(2)打包：tar -cvf 文件.tar 文件列表

3.zip类：(1)解压：unzip 文件.zip -d dir；(2)压缩：zip zipfile 待压缩文件列表

4.bz2类：(1)解压：bunzip2 文件.bz2或bzip2 -d 文件.bz2；(2)压缩：bzip2 待压缩文件

5.z类：(1)解压：uncompress 文件.z；(2)压缩：compress 文件

* 进程控制类

1.列出当前进程ID：ps -auxw

2.终止进程：

(1)终止单一进程：kill 进程ID号

(2)终止该程序所有进程：Killall 程序名

(3)终止X-Window程序：xkill

3.查看资源占用情况：(1)top (2)free (3)dmesg

4.查看环境变量值：env

5.重启：(1)reboot (2)Ctrl Alt Del (3)init 6

6.关机：(1)shutdown -h now (2)halt (3)init 0

7.切换桌面：switchdesk gnome|KDE|...

* 程序运行类

1.查询命令：whereis 命令名

2.后台运行X-Window程序：程序名&

3.强行退出X-Window程序：Ctrl Alt Backspace

4.查看帮助：

(1)简明帮助：命令名 --help | less

(2)更多帮助：man 命令名

(3)info 命令名

(4)help 命令名

5.查看系统路径：echo $PATH

6.查看当前shell堆栈：echo $SHLVL

7.< / >：输入/输出重定向;|：管道左的输入是管道右输入

* 用户帐号类

1.增加用户帐号：(1)用 户 名：adduser 用户帐号名

(2)设置密码： passwd 用户帐号名

2.删除用户帐号：userdel 用户帐号名

3.增加用户组：groupadd 用户组名

4.删除用户组：groupdel 用户组名

5.暂时终止用户帐号：passwd -l 用户帐号名

6.恢复被终止帐号：passwd -u 用户帐号名

7.权限设定

(1)chmod -a|u|g|o |-|=r|w|x 文件/目录名

其中：a--所有用户(all);u--本用户(user);g--用户组(group);o--其他用户(other users)

--增加权限;---删除权限;=--设置权限

文件：r--只读权限(read);w--写权限(write);x--执行权限(execute)

目录：r--允许列目录下文件和子目录;w--允许生成和删除目录下文件;x--允许访问该目录

(2)chmod xxx 文件/目录名

其中：execute=1;write=2;read=4

x取值：0--没有任何权限(常用);1--只能执行(不常见);2--只能写(不常见);3--只能写和执行(不常见);4--只读(常见);5--只读和执行(常见);6--读和写(常见);7--读.写和执行

* vi编辑类

1.进入后为命令模式：(1)插入i；(2)打开0；(3)修改c；(4)取代r；(5)替换s

2.经(1)后进入全屏幕编辑模式。

3.命令模式-->编辑模式(a/i)；编辑模式-->命令模式(Esc)；命令模式-->末行模式(：)。

4.：w/w newfile保存

5.：q/q!退出iv；：wq保存退出

* 网络服务

1.显示网络接口参数：ifconfig

2.显示系统邮件：mail

3.启动/终止web服务：httpd -k start|stop|restart

4.查看网络状况：(1)联机状况：ping xxx.xxx.xxx.xxx；

(2)显示网络状况：netstat ，其中：options：-a==所有sockets；-l==包含网络设备；-n==数字IP；

-o==其他信息；-r==路由表；-t==只列TCP sockets；-u==只列UDP sockets；-w==只列raw sockets；

-x==只列Unix Domain sockets

###### 五、库函数

**文件读写操作**

FILE \*fopen( const char \* filename, const char \* mode );

int fclose( FILE \*fp );

int fputs( const char \*s, FILE \*fp );

char \*fgets( char \*buf, int n, FILE \*fp );

**线程相关操作说明**

* pthread\_t

　　pthread\_t在头文件/usr/include/bits/pthreadtypes.h中定义：

　　typedef unsigned long int pthread\_t;

　　它是一个线程的标识符。

* pthread\_create

　　函数pthread\_create用来创建一个线程，它的原型为：

　　extern int pthread\_create \_\_P ((pthread\_t \*\_\_thread, \_\_const pthread\_attr\_t \*\_\_attr,

　　void \*(\*\_\_start\_routine) (void \*)， void \*\_\_arg));

　　第一个参数为指向线程标识符的指针，第二个参数用来设置线程属性，第三个参数是线程运行函数的起始地址，最后一个参数是运行函数的参数。这里，我们的函数thread不需要参数，所以最后一个参数设为空指针。第二个参数我们也设为空指针，这样将生成默认属性的线程。对线程属性的设定和修改我们将在下一节阐述。当创建线程成功时，函数返回0,若不为0则说明创建线程失败，常见的错误返回代码为EAGAIN和EINVAL.前者表示系统限制创建新的线程，例如线程数目过多了;后者表示第二个参数代表的线程属性值非法。创建线程成功后，新创建的线程则运行参数三和参数四确定的函数，原来的线程则继续运行下一行代码。

* pthread\_join pthread\_exit

　　函数pthread\_join用来等待一个线程的结束。函数原型为：

　　extern int pthread\_join \_\_P ((pthread\_t \_\_th, void \*\*\_\_thread\_return));

第一个参数为被等待的线程标识符，第二个参数为一个用户定义的指针，它可以用来存储被等待线程的返回值。这个函数是一个线程阻塞的函数，调用它的函数将一直等待到被等待的线程结束为止，当函数返回时，被等待线程的资源被收回。一个线程的结束有两种途径，一种是象我们上面的例子一样，函数结束了，调用它的线程也就结束了;另一种方式是通过

函数pthread\_exit来实现。它的函数原型为：

　　extern void pthread\_exit \_\_P ((void \*\_\_retval)) \_\_attribute\_\_ ((\_\_noreturn\_\_));

　　唯一的参数是函数的返回代码，只要pthread\_join中的第二个参数thread\_return不是NULL,这个值将被传递给 thread\_return.最后要说明的是，一个线程不能被多个线程等待，否则第一个接收到信号的线程成功返回，其余调用pthread\_join的线程则返回错误代码ESRCH.

　　并掌握了最常用的三个函数pthread\_create,pthread\_join和pthread\_exit.下面，我们来了解线程的一些常用属性以及如何设置这些属性。

　　互斥锁相关

　　互斥锁用来保证一段时间内只有一个线程在执行一段代码。

1. pthread\_mutex\_init

　　函数pthread\_mutex\_init用来生成一个互斥锁。NULL参数表明使用默认属性。如果需要声明特定属性的互斥锁，须调用函数 pthread\_mutexattr\_init.函数pthread\_mutexattr\_setpshared和函数 pthread\_mutexattr\_settype用来设置互斥锁属性。前一个函数设置属性pshared,它有两个取值， PTHREAD\_PROCESS\_PRIVATE和PTHREAD\_PROCESS\_SHARED.前者用来不同进程中的线程同步，后者用于同步本进程的不同线程。在上面的例子中，我们使用的是默认属性PTHREAD\_PROCESS\_ PRIVATE.后者用来设置互斥锁类型，可选的类型有PTHREAD\_MUTEX\_NORMAL、PTHREAD\_MUTEX\_ERRORCHECK、 PTHREAD\_MUTEX\_RECURSIVE和PTHREAD \_MUTEX\_DEFAULT.它们分别定义了不同的上所、解锁机制，一般情况下，选用最后一个默认属性。

1. pthread\_mutex\_lock pthread\_mutex\_unlock pthread\_delay\_np

　　pthread\_mutex\_lock声明开始用互斥锁上锁，此后的代码直至调用pthread\_mutex\_unlock为止，均被上锁，即同一时间只能被一个线程调用执行。当一个线程执行到pthread\_mutex\_lock处时，如果该锁此时被另一个线程使用，那此线程被阻塞，即程序将等待到另一个线程释放此互斥锁。