实验手册

第一天　Linux 基础

实验一 安装 linux 系统

【实验内容】

通过实验，学员掌握安装 linux 操作系统的方法和使用虚拟机。

【实验目的】

安装 linux 操作系统，系统为 ubuntu12.04 发行版。

【实验平台】

PC 机，如果你的电脑内存低于 512M，希望你不要装虚拟机。

【实验步骤】

1.打开虚拟机，如图 1：

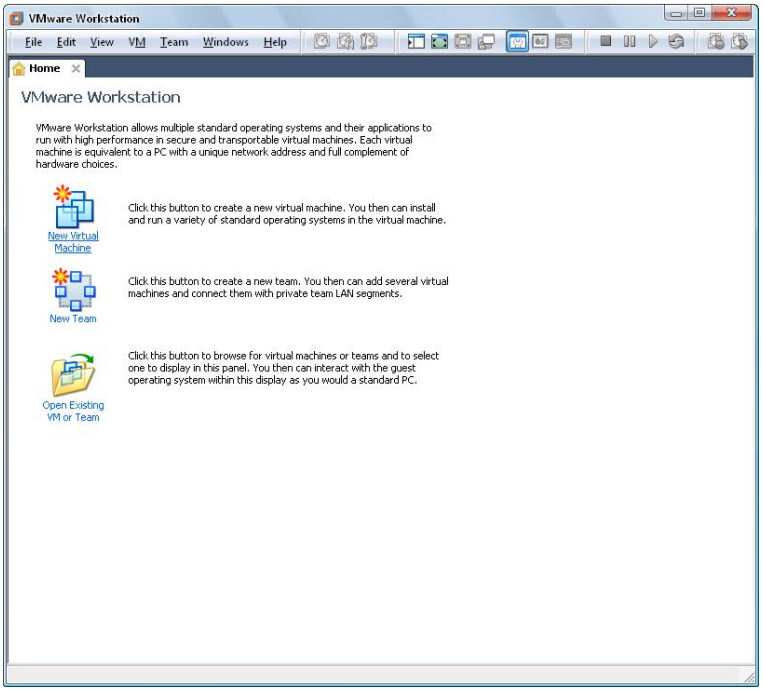

图 1

1. 启动已经安装好的 ubuntu 镜像

实验二 Linux 常用命令和 vi 的使用

【实验内容】

本课程要求学员对 Linux 基本操作命令有一定了解和掌握。下面列出的一些常用命令作为参考。最好针对每一个都能亲自练习、掌握。

【实验平台】

PC 机

【实验步骤】

1．ls 命令

---

ls　　以默认方式显示当前目录文件列表
ls　–a　　显示所有文件包括隐藏文件
ls　–l　　显示文件属性，包括大小，日期，符号连接，是否可读写及是否可执行

2.cd 命令
cd　dir　　切换到当前目录下的 dir 目录
cd　..　　切换到到上一级目录
cd　~　　切换到用户目录，比如是 root 用户，则切换到/root 下

3．rm 命令
rm　file　　删除某一个文件
rm　-rf　dir　　删除当前目录下叫 dir 的整个目录

4.cp 命令
cp　source　target　　将文件 source 复制为 target
cp　–a　soure_dir　target_dir　　将整个目录复制，两目录完全一样

5.mv 命令
mv　source　target　　将文件或者目录 source 更名为 target

6.find 命令
find　-name　<path>　file　在/path 目录下查找看是否有文件 file

7.vi 的使用

vi　file　　编辑文件 file
vi 原基本使用及命令：
vi 分为编辑状态和命令状态。输入命令要先按 ESC，退出编辑状态， 然后输入命令。
常用命令有：
: q(退出)
:q!(退出不保存)

:w(保存文件)
:w!(不询问方式写入文件)
:r file(读文件 file)
i 进入编辑插入状态
ESC 退出编辑状态

实验三 GCC 编译器的使用

【实验内容】

使用 GCC 编译工具，编译 C 程序，并生成可执行程序。

【实验目的】

掌握 GCC 编译 C 程序的方法，深入理解整个编译过程中的各个阶段。

【实验平台】

带有 Linux 操作系统的 PC 机。

【实验步骤】

1. 使用 vi 编辑器，编写 helloworld.c 程序代码如下：

```
#include <stdio.h>

int    main (int argc,char **argv)
{
       printf("hello,world!\n");
       return 0;
}
```

2. 使用 gcc 编译程序

$gcc –o hello helloworld.c

3. 执行应用程序

$./hello

实验四 C 语言高级应用之存储实验

【实验内容】

编写一个应用程序，生成一个文件，写入 Hello File

【实验目的】

深入了解标准 IO 文件读写的基本原理。

1.学习如何判断文件是否结束

2.熟练掌握标准 I/O 函数

【实验平台】

PC 机、ubuntu 操作系统，gcc 等工具

【实验提示】

4. 使用 vi 编辑器，编写 test.c 程序代码如下：

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

int    main (int argc,char **argv)
{
       int fd = open ("hello_file.txt", O_CREAT | O_RDWR);   //创建一个文件，文件名”hello_file.txt”
       if (fd < 0)
              return -1;

       write (fd, "Hello File", 11);     //向文件内写入
       close (fd);

       return 0;
}
```

5. 使用 gcc 编译程序

$gcc –o test test.c

6. 执行应用程序

$./test

查看目录下是否生成了 hello_file.txt 文件，并查看文件内容.

实验五 用程序在 Linux 下实现文件拷贝

（扩展，基础较好的同学选做）

【实验内容】

本实验通过一个简单的 copy 程序，完成文件的复制程序，了解基本的文件 I/O 文件读写的基本步骤。

【实验目的】

通过本实验掌握文件 I/O 的基本用法

【实验平台】

PC 机、ubuntu 操作系统，gcc 等工具

【实验步骤】

编写代码，实现相应的功能

打开源文件

打开目标文件

循环读取源文件并写入目标文件

关闭源文件

关闭目标文件

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

#define maxsize 256                          //定义一次从文件读字符的最大数
int main(int argc,char *argv[])
{

    int fd1,fd2;                          //定义源文件和目的文件的文件描述符
    char buff[maxsize];                     //缓冲
    int i;
    if(argc!=3)                             //如果命令格式不正确
    {
        printf("command error!\n");
        return -1;// exit(-1);
    }
```

```
    fd1=open(argv[1],O_RDONLY);            //以只读的方式打开源文件
    if(fd1==-1)
    {
        return -1;//exit(-1);
    }

    if((fd2=open(argv[2],O_WRONLY|O_CREAT|O_APPEND))==-1)// 以追加的方式创建目的文件
    {           printf("cannot creat file %s",argv[1]);
                return -1;// exit(-1);
    }

    while(1)
    {
      i=read(fd1,buff,maxsize);
      write(fd2,buff,i);
      if(i!=maxsize) break;     //如果读到的字节数不是希望的 bufsize，结束文件读写
    }
    close(fd1);
    close(fd2);
}
```

实验六 Linux 下用 C 语言实现网络通信

（扩展，基础超级好的同学选做）

【实验目的】

了解 socket 网络编程的基本方法

【实验内容】

本实验通过一个简单的 tcp 服务器端，接收客户端的连接请求，并发送欢迎信息。

【实验内容】

程序代码：

```
//tcp_server.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define MAXLINE 50
typedef struct sockaddr    SA;

int main(int argc, char **argv)
{
    int listenfd, connfd;
    socklen_t    clilen;
    struct sockaddr_in    myaddr, cliaddr;
    char    buf[MAXLINE] = “Welcome to TCP Server”;

    if ((listenfd = socket(PF_INET, SOCK_STREAM, 0)) < 0)
    {
        perror(”fail to socket”);
        exit(-1);
    }
    bzero(&myaddr, sizeof(myaddr));
    myaddr.sin_family           = PF_INET;
    myaddr.sin_addr.s_addr    = inet_addr(“192.168.1.100”);    //htonl(INADDR_ANY);
    myaddr.sin_port              = htons(8888);      /* port number */

    if (bind(listenfd, (SA *) &myaddr, sizeof(myaddr)) < 0)
    {
        perror(“fail to bind”);
        exit(-1);
```

```
    }
    listen(listenfd, 5);

    for ( ; ; )
    {
        clilen = sizeof(cliaddr);
        connfd = accept(listenfd, (SA *) &cliaddr, &clilen);
        printf("connection from %s, port %d\n",
                        inet_ntoa(cliaddr.sin_addr), ntohs(cliaddr.sin_port));
        send(connfd, buf, sizeof(buf), 0);
        close(connfd);
     }
     return 0;
}


//tcp_client.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define MAXLINE 50
typedef struct sockaddr SA;

int main(int argc, char **argv)
{
    int sockfd, nbyte;
    struct sockaddr_in    servaddr, myaddr;
    char    buf[MAXLINE];

    if ((sockfd = socket(PF_INET, SOCK_STREAM, 0)) < 0)
    {
        perror("fail to socket");
        exit(-1);
    }
    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family          = PF_INET;
    servaddr.sin_addr.s_addr = inet_addr("192.168.1.100");    //server ip;
    servaddr.sin_port              = htons(8888);      /* port number */
```

```
    if (connect(sockfd, (SA *) &servaddr, sizeof(servaddr)) < 0)
    {
        perror("fail to connect");
        exit (-1);
    }

    if ( (nbyte = recv(sockfd, buf, MAXLINE, 0)) < 0)
    {
        perror("fail to recv");
        exit(-1);
    }
    printf("recv from server : %s\n", buf);

    return 0;
}
```