南京邮电大学

实 验 报 告

（2024 / 2025 学年 第 一 学期）

课程名称	Linux 编程
实验名称	实验三　Linux 下 C 编程
实验时间	2024 年 12 月 9 日
指导单位	计算机学院
指导教师	王磊

学生姓名	顾茂阳	学号	B21111530
学院(系)	计算机学院	专　　业	计算机科学与技术

实 验 报 告

实验名称	实验三 Linux 下 C 编程			指导教师	王磊
实验类型	验证	实验学时	2	实验时间	2024.12.9

一、实验目的及实验要求

实验目的

进一步在 Linux 系统中运用 C 语言的基本语法，加深对该知识的理解。

二、实验环境(实验设备)

硬件：微型计算机

软件：Windows + VMWare + Ubuntu

三、实验原理及内容

（1）编写一个 C 程序，使用标准 I/O 库来显示文本文件的内容。程序由 make 工具编译和链接，这需要生成.o 文件，然后生成可执行文件，以及删除 makefile 文件中的中间文件（.o）的功能。

1、创建 C 程序 c1.c 和 makefile

c1.c 源代码：

```
#include <stdio.h>
int main(int argc, char* argv[])
{
    char buf[1024] = { 0 };
    FILE* fp = fopen(argv[1],"r");
    if (argc < 2)
    {
        printf("please input source file!\n");
```

```
    }
    if (fp == NULL)
    {
        printf("open source %s failed\n", argv[1]);
        return -1;
    }
    while (fgets(buf,1024, fp))
    {
        printf("%s\n", buf);
    }
    return 0;
}
```

makefile:

```
hello1:c1.o
    gcc -o hello1 c1.o
c1.o:c1.c
    gcc -c c1.c
clean:
    rm -rf *.o
```

2、编译和构建可执行文件

打开终端，进入保存文件的目录，运行以下命令：

```
make
```

3、运行程序

执行以下命令来显示文件内容:

```
./hello1 test.txt
```

4、运行程序结束后清理生成的文件：

```
make clean
```

删除所有 .o 文件和 hello1 可执行文件。

运行结果：

```
gumaoyang@gumaoyang-virtual-machine:~/Documents$ mkdir linux_test3_1
gumaoyang@gumaoyang-virtual-machine:~/Documents$ cd linux_test3_1
gumaoyang@gumaoyang-virtual-machine:~/Documents/linux_test3_1$ touch c1.c
gumaoyang@gumaoyang-virtual-machine:~/Documents/linux_test3_1$ touch makefile
gumaoyang@gumaoyang-virtual-machine:~/Documents/linux_test3_1$ touch test.txt
gumaoyang@gumaoyang-virtual-machine:~/Documents/linux_test3_1$ make
makefile:2: *** missing separator.  Stop.
gumaoyang@gumaoyang-virtual-machine:~/Documents/linux_test3_1$ make
gcc -c c1.c
gcc -o hello1 c1.o
gumaoyang@gumaoyang-virtual-machine:~/Documents/linux_test3_1$ ./hello1 test.txt
这是一个测试文件。

文件的第二行。

文件的第三行。



gumaoyang@gumaoyang-virtual-machine:~/Documents/linux_test3_1$ make clean
rm -rf *.o
gumaoyang@gumaoyang-virtual-machine:~/Documents/linux_test3_1$
```

图 1 编写一个 C 程序，使用标准 I/O 库来显示文本文件的内容

（2）编写一个 C 程序，显示当前目录下的所有文件名。程序由 make 工具编译和链接，这需要生成.o 文件，然后生成可执行文件，以及删除 makefile 文件中的中间文件（.o）的功能。

1、创建 C 程序 c2.c 和 makefile：

c2.c 源代码：

```
#include <stdio.h>
#include <dirent.h>
#include <sys/types.h>

int    main(int argc, char* argv[])
{
       DIR* dirp;
       struct dirent* direntp;
       if ((dirp = opendir(argv[1])) == NULL) {
              printf("error\n");
        //      exit(1);
       }
       while ((direntp = readdir(dirp)) != NULL)
              printf("%s\n", direntp->d_name);
       closedir(dirp);
       // exit(0);
}
```

makefile:

```
hello2:c2.o
	gcc -o hello1 c2.o
c2.o:c2.c
	gcc -c c2.c
clean:
	rm -rf *.o
```

2、编译和构建可执行文件

打开终端，进入保存文件的目录，运行以下命令：

```
make
```

3、运行程序

执行以下命令来显示当前目录下的所有文件名:

```
./hello1 .
```

4、运行程序结束后清理生成的文件：

```
make clean
```

删除所有 .o 文件和 hello1 可执行文件。

运行结果：

```
gumaoyang@gumaoyang-virtual-machine:~/Documents/linux_test3_1$ cd ..
gumaoyang@gumaoyang-virtual-machine:~/Documents$ mkdir linux_test3_2
gumaoyang@gumaoyang-virtual-machine:~/Documents$ cd linux_test3_2
gumaoyang@gumaoyang-virtual-machine:~/Documents/linux_test3_2$ touch c2.c
gumaoyang@gumaoyang-virtual-machine:~/Documents/linux_test3_2$ touch makefile
```

```
gumaoyang@gumaoyang-virtual-machine:~/Documents/linux_test3_2$ make
gcc -c c2.c
gcc -o hello1 c2.o
gumaoyang@gumaoyang-virtual-machine:~/Documents/linux_test3_2$ ./hello1 Documents
error
Segmentation fault (core dumped)
gumaoyang@gumaoyang-virtual-machine:~/Documents/linux_test3_2$ ./hello2 Documents
bash: ./hello2: No such file or directory
gumaoyang@gumaoyang-virtual-machine:~/Documents/linux_test3_2$ ./hello1 .
makefile
.
c2.c
hello1
c2.o
..
gumaoyang@gumaoyang-virtual-machine:~/Documents/linux_test3_2$ make clean
rm -rf *.o
```

图 2 编写一个 C 程序，显示当前目录下的所有文件名

（3）写一个 C 程序，改变当前进程的工作目录。程序由 make 工具编译和链接，这需要生成.o 文件，然后生成可执行文件，以及删除 makefile 文件中的中间文件（.o）的功能。

1、创建 C 程序 c3.c 和 makefile：

c3.c 源代码：

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main(){
	char buf[1024] = {0};
	char buf2[1024]={0};
	getcwd(buf, 1024);
	printf("%s\n", buf);
	if(chdir("/home")<0){
		printf("error\n");
	}
	else
	{
		printf("success\n");
	}
	getcwd(buf2,1024);
	printf("%s\n",buf2);
	return 0;
}
```

makefile:

```
hello3:c3.o
	gcc -o hello1 c3.o
c3.o:c3.c
	gcc -c c3.c
clean:
	rm -rf *.o
```

2、编译和构建可执行文件

打开终端，进入保存文件的目录，运行以下命令：

make

3、运行程序

执行以下命令来改变当前进程的工作目录:

./hello1

4、运行程序结束后清理生成的文件：

make clean

删除所有 .o 文件和 hello1 可执行文件。

运行结果：

```
gumaoyang@gumaoyang-virtual-machine:~/Documents/linux_test3_2$ cd ..
gumaoyang@gumaoyang-virtual-machine:~/Documents$ mkdir linux_test3_3
gumaoyang@gumaoyang-virtual-machine:~/Documents$ cd linux_test3_3
gumaoyang@gumaoyang-virtual-machine:~/Documents/linux_test3_3$ touch c3.c
gumaoyang@gumaoyang-virtual-machine:~/Documents/linux_test3_3$ touch makefile
gumaoyang@gumaoyang-virtual-machine:~/Documents/linux_test3_3$ make
gcc -c c3.c
gcc -o hello1 c3.o
gumaoyang@gumaoyang-virtual-machine:~/Documents/linux_test3_3$ ./hello1
/home/gumaoyang/Documents/linux_test3_3
success
/home
gumaoyang@gumaoyang-virtual-machine:~/Documents/linux_test3_3$ make clean
rm -rf *.o
gumaoyang@gumaoyang-virtual-machine:~/Documents/linux_test3_3$
```

图 3 写一个 C 程序，改变当前进程的工作目录

四、实验小结（包括问题和解决方法、心得体会、意见与建议等）

在完成这三个实验后，我深刻感受到编程和调试过程中的细节和重要性。从编写简单的 C 程序到使用 make 工具进行项目的管理，每一步都加强了我对程序结构、编译过程和错误排查的理解。

文件内容显示程序： 第一个实验要求编写一个程序，读取并显示文件内容。这一实验让我更加熟悉了文件操作和标准 I/O 库的使用。通过逐行读取文件内容并进行输出，我掌握了如何处理文件打开失败、读取失败等常见问题。在实现过程中，我认识到正确的错误检查非常重要，尤其是在文件处理过程中，操作系统可能会因为路径不存在或权限不足等问题导致打开失败。

列出目录文件名程序： 第二个实验要求实现一个列出指定目录下所有文件名的程序。这个实验让我学习了如何使用 dirent.h 库来访问和遍历目录内容。最初，我遇到了一个 Segmentation fault 错误，这让我意识到程序中的指针操作和命令行参数的传递方式至关重要。通过对错误的分析和对代码的改进，我学会了如何通过 perror 提供详细的错误信息，以及如何更好地处理参数检查和内存访问问题。

改变当前进程的工作目录：第三个实验要求实现一个改变当前进程的工作目录的程序，这个实验主要涉及如何使用 chdir() 系统调用来改变当前进程的工作目录，以及如何使用 make 工具来管理构建过程。

通过这三个实验，我不仅掌握了 C 语言中文件操作、目录操作以及标准 I/O 的基本用法，还深入理解了 make 工具的强大功能。这些技能在日后的程序开发和调试中无疑是不可或缺的。在实际编程中，细心处理边界情况、检查错误和善用工具是提高效率和代码质量的关键。每个实验的解决方案都让我感受到，编程不仅是写出正确的代码，更是在面对问题时保持耐心和细致的心态，通过调试和不断修改完善最终的解决方案。

.