

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	在 Linux 系统中使用 C 编程语言的基本语法		
实验时间	2024	年 12 月 13 日	
指导单位	计算机学院网络空间安全系		
指导教师	王磊		

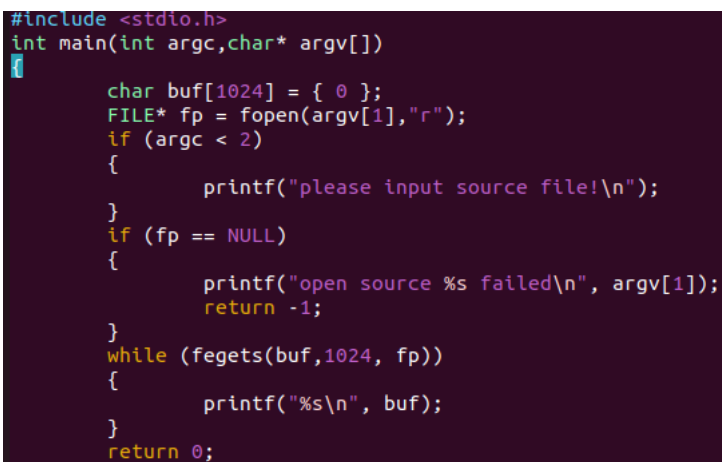
学生姓名	丁伟	班级学号	B21090511
学院(系)	计软网安学院	专 业	信息安全

实 验 报 告

实验名称	掌握 Linux 命令的使用			指导教师	王磊
实验类型	上机	实验学时		实验时间	2024. 12. 13
一、 实验目的和要求 1. 目标 进一步在 Linux 系统中使用 C 语言的基本语法，加深对知识的理解。 2. 实验要求 (1) 编写一个使用标准 I/O 库来显示文本文件内容的 C 程序。程序由 make 工具编译和链接，需要先生成 .o 文件，再生成可执行文件，并有删除 makefile 文件中中间文件 (.o) 的功能输入两个数字，检查哪个更大，然后输出结果 Linux 用户管理 (2) 编写一个 C 程序，显示当前目录中的所有文件名。程序由 make 工具编译和链接，需要先生成 .o 文件，再生成可执行文件，并有删除 makefile 文件中中间文件 (.o) 的功能。 (3) 编写一个 C 程序来更改当前进程的工作目录。程序由 make 工具编译和链接，需要先生成 .o 文件，再生成可执行文件，并有删除 makefile 文件中中间文件 (.o) 的功能					
二、 实验环境(实验设备) 1. 安装 ubuntu					
三、 实验内容 1. Write a C program that uses standard I/O libraries to display the contents of text files. The program is compiled and linked by the make tool, which requires the generation of the.o file first, and then the generation of the executable file, and					

the function of deleting the intermediate file (.o) in the makefile file.

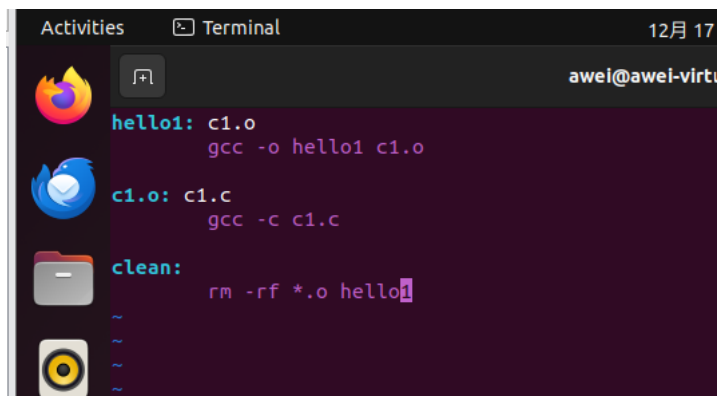
```
#include <stdio.h>
int main(int argc, char* argv[])
{
    char buf[1024] = { 0 };
    FILE* fp = fopen(argv[1], "r");
    if (argc < 2)
    {
        printf("please input source file!\n");
    }
    if (fp == NULL)
    {
        printf("open source %s failed\n", argv[1]);
        return -1;
    }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



```
#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;
}
```

Make sure your filename is `cl.c`
We can use the following makefile.

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

A screenshot of a Linux terminal window titled 'Terminal' with the date '12月 17' in the top right corner. The terminal shows the execution of the makefile commands. The first command is 'hello1: cl.o', which triggers the compilation of 'cl.o' into 'hello1' using 'gcc -o hello1 cl.o'. The second command is 'cl.o: cl.c', which triggers the compilation of 'cl.c' into 'cl.o' using 'gcc -c cl.c'. The third command is 'clean:', which triggers the removal of object files using 'rm -rf *.o hello1'. The terminal output is as follows:

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

运行结果:

2. Write a C program that uses standard I/O libraries to display the contents of text files. The program is compiled and linked by the make tool, which requires the generation of the.o file first, and then the generation of the executable file, and

the function of deleting the intermediate file (.o) in the makefile file.

```
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);
}
```

Make sure your filename is c2.c

We can use the following makefile.

```
hello2:c2.o

    gcc -o hello1 c2.o

c2.o:c2.c

    gcc -c c2.c

clean:

    rm -rf *.o
```

运行结果:


3. Write a C program that changes the working directory of the current process. The program is compiled and linked by the make tool, which requires the generation of the.o file first, and then the generation of the executable file, and the function of deleting the intermediate file (.o) in the makefile file

```
#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;
}

```



```

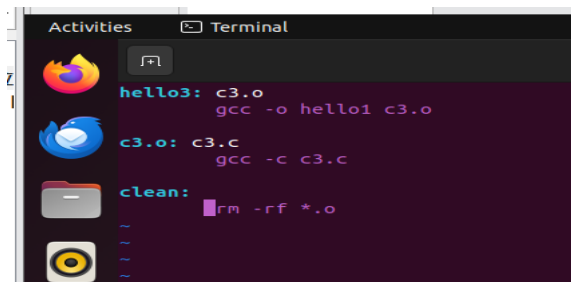
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main(){
    char buf[1024] = {0};
    char buf2[1024] = {0};
    getcwd(buf, 1024);
    printf("%s\n", buf);
    if(chdir("/home")<0){
        printf("error\n");
    }
    else
    {
        printf("success\n");
    }
    getcwd(buf2, 1024);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

```

```

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

```



```

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

```

运行结果:

```
awei@awei-virtual-machine:~$ vim c3.c
awei@awei-virtual-machine:~$ vim makefile
awei@awei-virtual-machine:~$ make
gcc -c c3.c
gcc -o hello1 c3.o
awei@awei-virtual-machine:~$ ./hello3
bash: ./hello3: No such file or directory
awei@awei-virtual-machine:~$ make clean
rm -rf *.o
awei@awei-virtual-machine:~$
```


四、实验小结（包括问题和解决方法、心得体会、意见与建议等）

通过本实验,我掌握了如何使用 `chdir` 函数来更改当前进程的工作目录。此外,我还复习了如何使用 `getcwd` 函数来获取当前工作目录的路径。这次实验不仅让我熟悉了目录操作的相关函数,还让我对 `make` 工具和 `Makefile` 的使用有了更深入的理解。同时,我也学会了如何在 C 语言中进行基本的错误处理,如检查函数返回值以判断操作是否成功。

五、指导教师评语

成 绩		批阅人		日 期	
-----	--	-----	--	-----	--