

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：C 语言编程实验
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	梁榆漫	班级学号	B220412/B22100106
学院(系)	计算机学院	专 业	信息安全

# 实验报告

实验名称	Linux 实验 3			指导教师	王磊
实验类型	验证	实验学时	2	实验时间	2024.12.13
<div>一、实验目的和要求</div> <div>进一步在 Linux 系统中使用 C 语言的基本语法，加深对相关知识的理解。</div>					
<div>二、实验环境(实验设备)</div> <div>硬件：微型计算机</div> <div>软件：实验环境：Linux 系统：CentOS 7</div>					
<div>三、实验原理及内容</div> <div>1. 任务 1：显示文本文件内容</div> <div>任务要求：</div> <div>编写一个 C 程序，使用标准 I/O 库来显示文本文件的内容。该程序通过 make 工具进行编译和链接，要求首先生成.o 文件，然后生成可执行文件，并在 makefile 文件中加入删除中间文件（.o）的功能。</div> <div><pre>#include &lt;stdio.h&gt; int main(int argc, char* argv[]) {     char buf[1024] = { 0 };     FILE* fp = fopen(argv[1], "r");     if (argc &lt; 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))</pre></div>					

```
{
    printf("%s\n", buf);
}
return 0;
}
```

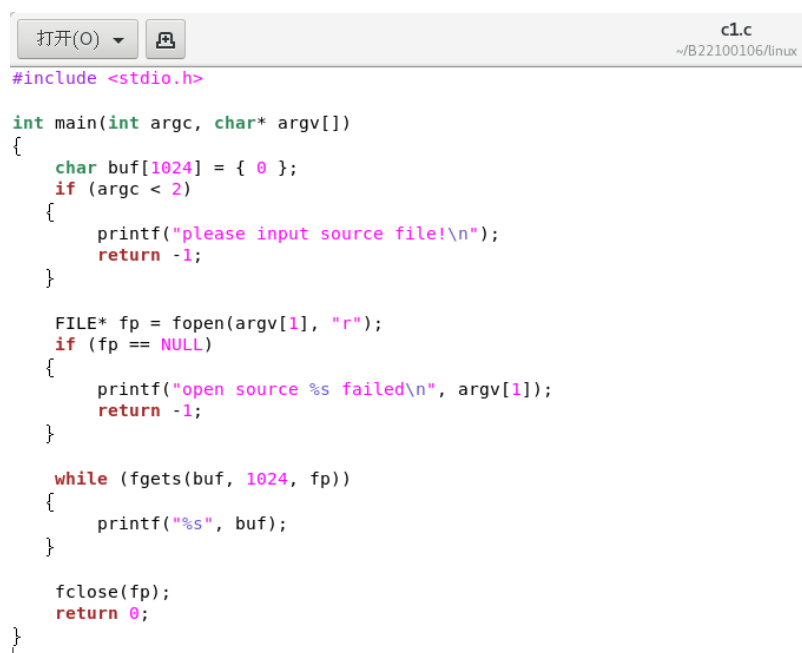
**解析：** 这个任务要求我们编写一个程序，读取并输出指定文本文件的内容。首先，使用 `fopen` 打开文件，如果文件打开失败则输出错误信息。然后通过 `fgets` 逐行读取文件内容并输出到屏幕上。我们还使用了命令行参数 `argv[1]` 来指定文件路径。

### Makefile:

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

### 解析：

通过 Makefile 文件，我们能够方便地编译 C 程序。`hello1:c1.o` 部分是指定链接规则，生成 `hello1` 可执行文件；`c1.o:c1.c` 是指定如何从源代码 `c1.c` 生成目标文件 `c1.o`；`clean` 则用于清理所有中间文件。



```
c1.c
~/B22100106/linux

#include <stdio.h>

int main(int argc, char* argv[])
{
    char buf[1024] = { 0 };
    if (argc < 2)
    {
        printf("please input source file!\n");
        return -1;
    }

    FILE* fp = fopen(argv[1], "r");
    if (fp == NULL)
    {
        printf("open source %s failed\n", argv[1]);
        return -1;
    }

    while (fgets(buf, 1024, fp))
    {
        printf("%s", buf);
    }

    fclose(fp);
    return 0;
}
```

图 1



```
Makefile
~/B22100106/linux

hello1: c1.o
    gcc -o hello1 c1.o

c1.o: c1.c
    gcc -c c1.c

clean:
    rm -rf *.o hello1
```

图 2

运行 `make` 命令，该命令将会：

首先生成 c1.o 文件（中间文件）。

然后链接生成最终的可执行文件 hello1

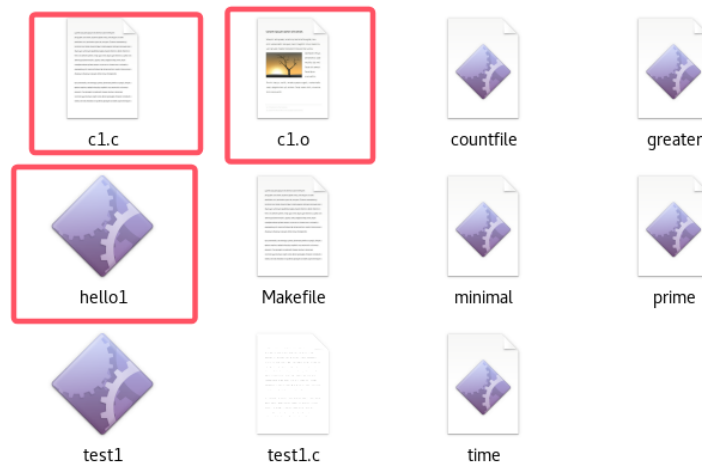


图 3

**运行程序：** 使用生成的可执行文件读取文本文件内容：

`./hello1 Makefile`

```
[user@localhost linux]$ ./hello1 Makefile
hello1: c1.o
        gcc -o hello1 c1.o

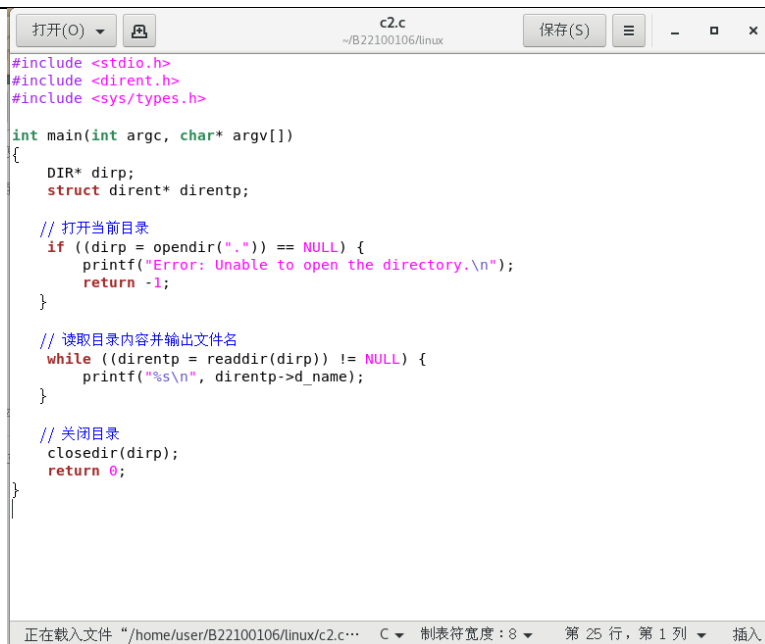
c1.o: c1.c
        gcc -c c1.c

clean:
        rm -rf *.o hello1
```

图 4

## 2. 任务 2：显示当前目录下的所有文件名

**任务要求：** 编写一个 C 程序，显示当前目录下的所有文件名。该程序通过 make 工具进行编译和链接，要求首先生成.o 文件，然后生成可执行文件，并在 makefile 文件中加入删除中间文件（.o）的功能。



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

int main(int argc, char* argv[])
{
    DIR* dirp;
    struct dirent* direntp;

    // 打开当前目录
    if ((dirp = opendir(".")) == NULL) {
        printf("Error: Unable to open the directory.\n");
        return -1;
    }

    // 读取目录内容并输出文件名
    while ((direntp = readdir(dirp)) != NULL) {
        printf("%s\n", direntp->d_name);
    }

    // 关闭目录
    closedir(dirp);
    return 0;
}
```

正在载入文件 "/home/user/B22100106/linux/c2.c... C 制表符宽度: 8 第 25 行, 第 1 列 插入

图 5

同样生成 Makefile 文件，运行 make 指令：

该命令将：

生成 c2.o 文件（中间文件）。

链接生成最终的可执行文件 hello2。



图 6

1. 运行程序：使用生成的可执行文件列出当前目录中的文件名：  
./hello2

```
[user@localhost linux]$ ./hello2
.
..
test1.c
test1
greater
countfile
minimal
prime
time
c1.c
Makefile
c1.o
hello1
c2.c
c2.o
hello2
```

图 7

### 3. 任务 3: 更改当前进程的工作目录

#### 任务要求:

编写一个 C 程序, 修改当前进程的工作目录。程序通过 `make` 工具进行编译和链接, 要求首先生成 `.o` 文件, 然后生成可执行文件, 并在 `makefile` 文件中加入删除中间文件 (`.o`) 的功能。

```
#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;
}
```

#### 解析:

程序使用 `getcwd` 获取当前工作目录并打印。然后, 使用 `chdir` 函数修改工作目录为 `/home`。如果修改成功, 输出 `success`, 否则输出 `error`。修改后的工作目录再次通过 `getcwd` 函数获取并打印。

同样生成 `Makefile` 文件, 运行 `make` 指令:

该命令将:

生成 c3.o 文件（中间文件）。  
链接生成最终的可执行文件 hello3。

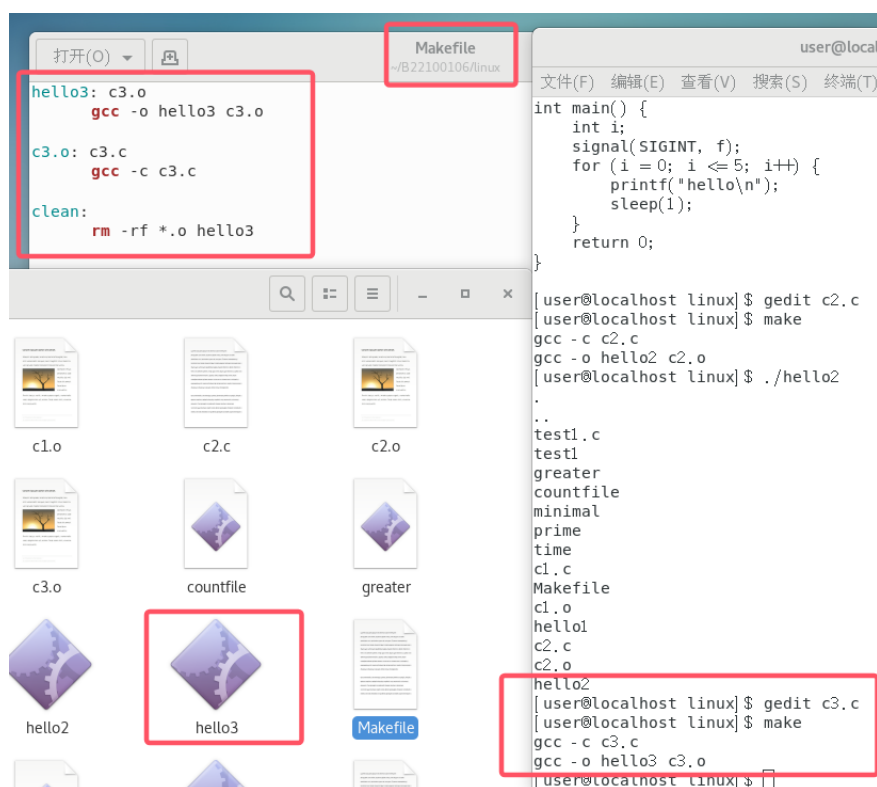


图 8

运行进程

```
[user@localhost linux]$ ./hello3
Current directory: /home/user/B22100106/linux
Changed directory to /home successfully.
New directory: /home
```

图 9

可以观察到工作路径已经改变为指定的新路径

## 四、 实验总结

### 1. 遇到问题和解决方法：

**问题：**任务 3 中，执行程序后发现工作路径的更改未在后续命令中生效。

**原因：**程序中调用 `chdir` 仅对当前进程的工作目录有效，对其他进程没有影响。

**解决：**通过实验验证这一特性，并理解操作系统中进程间的独立性。

### 2. 心得体会

使用 Makefile 工具可以显著提升 C 程序的编译管理效率，特别是在有多个中间文件或依赖关系时尤为重要。

理解进程独立性对操作系统编程十分关键，尤其在修改工作目录等操作中，需要清楚当前进程的作用范围和限制。

通过本次实验，深刻体会到 C 语言编程与 Makefile 工具在实际开发中的重要性。任务不仅巩固了对标准 I/O 库和进程管理的理解，还提升了代码管理与编译的实践能

力，为更复杂的系统开发打下了扎实的基础。

## 实 验 报 告