一、实验目的

掌握 Linux 环境下 C 语言标准 I/O 操作、目录操作及进程工作目录控制；熟练使用 make 工具完成 C 程序编译链接，掌握 Makefile 编写规范，实现.o 中间文件生成与清理功能。

二、实验要求

1. 编写 c1.c，用标准 I/O 库函数读取并显示文本文件内容；用 make 工具编译（先生成.o 中间文件，再生成可执行文件），编写 Makefile 支持中间文件清理。

2. 编写 c2.c，遍历并显示当前目录下所有文件名；用 make 工具编译，Makefile 需包含.o 文件生成、可执行文件生成及中间文件清理功能。

3. 编写 c3.c，实现当前进程工作目录修改功能；用 make 工具编译，Makefile 支持中间文件生成、可执行文件生成及清理。

三、实验内容与说明

（一）任务 1：基于标准 I/O 库的文本文件内容显示程序

1. 实验目标：编写 c1.c 读取显示指定文本文件内容；编写 Makefile，通过 make 完成“.o 中间文件→可执行文件”编译流程，支持中间文件清理。

2. 程序设计与优化：核心逻辑为通过命令行参数获取文件路径，用标准 I/O 库函数读取输出内容。实验中对代码优化：初始化文件指针避免野指针；增加参数校验，未传入文件名则提示用法并退出；完善文件打开失败提示；修正 printf 输出格式，避免空行问题。

3. Makefile 配置与问题解决：目标可执行文件 hello1 依赖 c1.o，c1.o 依赖 c1.c（通过 gcc -c 生成）。初始因命令行首未用 Tab 缩进报错，修正后问题解决；clean 目标通过 rm -rf *.o 清理.o 文件。

4. 编译运行与结果分析：执行 make 生成 c1.o 和 hello1；创建 test.txt 并写入测试内容，运行./hello1 test.txt 成功输出文件内容，功能验证正常。

相关实操截图说明：含 c1.c 代码、Makefile 配置、make 编译及程序运行结果截图，完整记录实验流程。

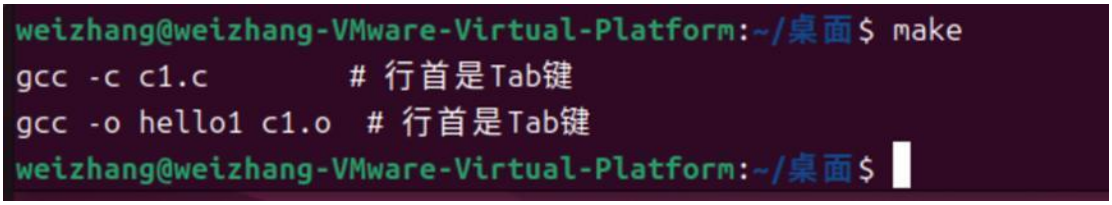
打开(O)

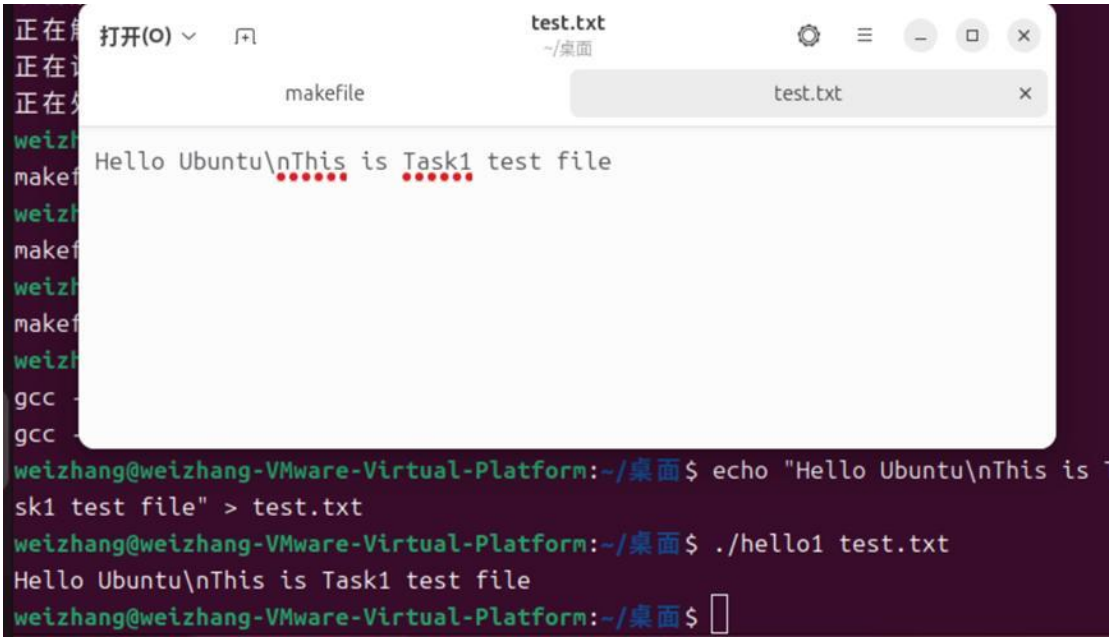
c1.c
~/桌面/linux code

```c
#include <stdio.h>
int main(int argc, char* argv[])
{
    char buf[1024] = { 0 };
    FILE* fp = NULL; // 初始化指针，避免野指针风险

    // 检查是否传入文件名参数
    if (argc < 2)
    {
        printf("please input source file! (用法: ./hello1 文件名)\n");
        return -1; // 缺少参数时退出，避免后续fp为NULL的错误
    }

    // 以读模式打开文件
    fp = fopen(argv[1], "r");
    if (fp == NULL)
    {
        printf("open source %s failed (文件不存在或无读取权限)\n", argv[1]);
        return -1;
    }

    // 逐行读取文件内容并显示
    while (fgets(buf, 1024, fp))
    {
        printf("%s", buf); // 去掉多余\n，避免输出空行（fgets已包含换行符）
    }
```

```
hello3:c3.o
	gcc -o hello3 c3.o  # 行首必须是Tab键
# 生成中间文件c3.o
c3.o:c3.c
	gcc -c c3.c
# 清理中间文件
clean:
	rm -rf *.o
```

```
weizhang@weizhang-VMware-Virtual-Platform:~/桌面$ make
makefile:3: *** 缺失分隔符。  停止。
weizhang@weizhang-VMware-Virtual-Platform:~/桌面$
```





（二）任务 2：当前目录下所有文件名的遍历显示程序

1. 实验目标：编写 c2.c 遍历显示指定目录下所有文件名；用 make 完成编译，Makefile 支持.o 文件生成、可执行文件生成及清理。

2. 程序设计与优化：核心逻辑为通过目录操作函数打开目录、遍历目录项并显示文件名。实验中完善代码：补充 stdlib.h 头文件适配 exit()函数；未传入目录参数时默认遍历当前目录；完善目录打开失败提示；初始化目录及目录项指针避免异常。

3. Makefile 配置：目标可执行文件 hello2 依赖 c2.o，c2.o 通过 gcc -c c2.c 生成；clean 目标用 rm -rf *.o 清理中间文件，命令行首均用 Tab 缩进避免报错。

4. 编译运行与结果分析：执行 make 生成 c2.o 和 hello2；运行./hello2 成功遍历显示当前目录下所有文件（含中间文件、可执行文件等）；执行 make clean 成功清理所有.o 文件，功能正常。

相关实操截图说明：含 c2.c 代码、Makefile 配置、编译运行及 make clean 结果截图，完整呈现实验环节。

```
weizhang@weizhang-VMware-Virtual-Platform:~/桌面$ gedit c2.c
weizhang@weizhang-VMware-Virtual-Platform:~/桌面$ gedit makefile
weizhang@weizhang-VMware-Virtual-Platform:~/桌面$ make
gcc -c c2.c
gcc -o hello2 c2.o  # 行首必须是Tab键
weizhang@weizhang-VMware-Virtual-Platform:~/桌面$ ./hello2
..
c2.o
linux code
Operating System code
hello2
hello1
your_studentID
.
c2.c
makefile
voln1.c
weizhang@weizhang-VMware-Virtual-Platform:~/桌面$ make clean
rm -rf *.o
weizhang@weizhang-VMware-Virtual-Platform:~/桌面$
```

打开(O)

c1.c

c2.c
~/桌面

```c
#include <stdio.h>     // 补充#，原代码遗漏
#include <dirent.h>  // 目录操作头文件
#include <sys/types.h>
#include <stdlib.h>    // 补充：exit()函数依赖此头文件

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

    // 若未传入目录参数，默认打开当前目录（.表示当前目录）
    char* dir_path = (argc >= 2) ? argv[1] : ".";

    // 打开指定目录
    if ((dirp = opendir(dir_path)) == NULL) {
        printf("error: 无法打开目录 %s（目录不存在或无权限）\n", dir_path);
        exit(1); // 退出程序，避免后续错误
    }

    // 遍历目录，打印所有文件名
    while ((direntp = readdir(dirp)) != NULL)
    {
        printf("%s\n", direntp->d_name); // 输出文件名（含隐藏文件，如.和..）
```

（三）任务 3：当前进程工作目录修改程序

1. 实验目标：编写 c3.c 实现当前进程工作目录修改功能；用 make 完成编译，Makefile 支持中间文件生成、可执行文件生成及清理。

2. 程序设计与优化：核心逻辑为获取当前工作目录，调用函数修改目录后再次获取目录验证结果。实验中优化代码：补充 getcwd()函数错误处理；完善目录切换提示信息；确保 unistd.h 头文件完整以支持 chdir()和 getcwd()函数。

3. Makefile 配置：目标可执行文件 hello3 依赖 c3.o，c3.o 通过 gcc -c c3.c 生成；clean 目标用 rm -rf *.o 清理中间文件，遵循命令行首 Tab 缩进规范。

4. 编译运行与结果分析：执行 make 生成 c3.o 和 hello3；运行./hello3 成功将工作目录从桌面切换到/home，输出切换前后目录信息；执行 make clean 顺利清理.o 文件，功能验证正常。

相关实操截图说明：含 c3.c 代码、Makefile 配置、编译运行及清理结果截图，完整记录实验流程。

```
weizhang@weizhang-VMware-Virtual-Platform:~/桌面$ gedit c3.c
weizhang@weizhang-VMware-Virtual-Platform:~/桌面$ gedit makefile
weizhang@weizhang-VMware-Virtual-Platform:~/桌面$ make
gcc -c c3.c
gcc -o hello3 c3.o  # 行首必须是Tab键
weizhang@weizhang-VMware-Virtual-Platform:~/桌面$ ./hello3
切换前的工作目录：/home/weizhang/桌面
目录切换成功！
切换后的工作目录：/home
weizhang@weizhang-VMware-Virtual-Platform:~/桌面$ make clean
rm -rf *.o
weizhang@weizhang-VMware-Virtual-Platform:~/桌面$
```

打开(O)

c3.c
~/桌面

c1.c | c2.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h> // chdir()、getcwd()依赖此头文件

int main(){
    char buf[1024] = {0};
    char buf2[1024] = {0};

    // 获取切换前的当前工作目录
    if (getcwd(buf, 1024) == NULL) { // 补充错误判断，原代码未处理
        printf("error: 无法获取当前目录\n");
        return -1;
    }
    printf("切换前的工作目录: %s\n", buf);

    // 切换工作目录到/home
    if(chdir("/home") < 0){
        printf("error: 无法切换到/home目录（无权限）\n");
        return -1;
    }
    else
    {
        printf("目录切换成功! \n");
    }
```

```
hello3:c3.o
	gcc -o hello3 c3.o  # 行首必须是Tab键
# 生成中间文件c3.o
c3.o:c3.c
	gcc -c c3.c
# 清理中间文件
clean:
	rm -rf *.o
```

四、实验总结

本次实验掌握 Linux 环境下 C 语言标准 I/O、目录操作及进程工作目录控制；熟练使用 make 工具及 Makefile 编写规范，解决了 Makefile“缺失分隔符”报错，实现完整编译与清理流程；积累了代码优化与问题排查经验。

实验提升了程序健壮性与可读性，深化了 make 工具自动化编译优势的理解，强化了 Linux 下 C 语言开发实操能力，为后续系统编程实验奠定基础。