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。

```
xuxing@ubuntu:~/B23041316$ gedit c1.c
xuxing@ubuntu:~/B23041316$ gedit Makefile
xuxing@ubuntu:~/B23041316$ make
cc -c c1.c  # 行首为Tab键，仅编译c1.c生成.o中间文件，不链接
cc -o hello1 c1.o  # 行首为Tab键，将c1.o链接为可执行文件hello1
xuxing@ubuntu:~/B23041316$ gedit test.txt
xuxing@ubuntu:~/B23041316$ echo "Hello Ubuntu\nThis is Task1 test file" > test.t
xt
xuxing@ubuntu:~/B23041316$ ls
c1.c                demo          Makefile          test.txt
c1.o                find_min.sh   number_check.sh   time_check.sh
count_exefile1.sh   hello1        prime_check.sh
xuxing@ubuntu:~/B23041316$ ./hello1 test.txt
Hello Ubuntu\nThis is Task1 test file
xuxing@ubuntu:~/B23041316$ make clean
rm -rf *.o  # 行首为Tab键，删除当前目录下所有以.o结尾的文件
xuxing@ubuntu:~/B23041316$
```

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

    // 校验命令行参数，未传入文件名则提示用法并退出
    if (argc < 2)
    {
        printf("please input sourcefile!(用法:./hello1 文件名)\n");
        return -1;
    }

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

    // 逐行读取文件内容并显示（fgets已包含换行符，printf不额外加\n避免空行）
    while (fgets(buf, 1024, fp))
    {
        printf("%s", buf);
    }

    // 关闭文件流，避免资源泄漏
    fclose(fp);
    return 0;
}
```

```
# 目标1：生成可执行文件hello1，依赖c1.o
hello1: c1.o
	cc -o hello1 c1.o  # 行首为Tab键，将c1.o链接为可执行文件hello1

# 目标2：生成中间文件c1.o，依赖c1.c
c1.o: c1.c
	cc -c c1.c  # 行首为Tab键，仅编译c1.c生成.o中间文件，不链接

# 目标3：清理所有.o中间文件
clean:
	rm -rf *.o  # 行首为Tab键，删除当前目录下所有以.o结尾的文件
```

（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。

```c
#include <stdio.h>
#include <dirent.h>    // 目录操作核心头文件（opendir、readdir等函数）
#include <sys/types.h> // 定义DIR、struct dirent等类型
#include <stdlib.h>    // 包含exit()函数，处理目录打开失败场景

int main(int argc, char* argv[])
{
    DIR* dirp = NULL;          // 目录指针，初始化避免异常
    struct dirent* direntp = NULL; // 目录项指针，存储目录下文件信息

    // 处理命令行参数：未传入目录则默认遍历当前目录（"."表示当前目录）
    char* dir_path = (argc >= 2) ? argv[1] : ".";

    // 打开指定目录，判断是否成功
    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); // 目录打开失败，退出程序
    }

    // 遍历目录：循环读取目录项，直到读取完毕（readdir返回NULL表示结束）
    while ((direntp = readdir(dirp)) != NULL)
    {
        // 输出文件名（d_name是struct dirent的成员，存储文件名，含隐藏文件如"."和".."）
        printf("%s\n", direntp->d_name);
    }

    // 关闭目录流，释放资源
    closedir(dirp);
    return 0;
}
```

```
xuxing@ubuntu: ~/B23041316
xuxing@ubuntu:~/B23041316$ gedit c2.c
xuxing@ubuntu:~/B23041316$ make hello2
cc -c c2.c  # 行首为Tab键，编译c2.c生成c2.o
cc -o hello2 c2.o  # 行首为Tab键，链接c2.o生成可执行文件hello2
xuxing@ubuntu:~/B23041316$ ls
c1.c  count_exefile1.sh  hello1    number_check.sh  time_check.sh
c2.c  demo               hello2    prime_check.sh
c2.o  find_min.sh        Makefile  test.txt
xuxing@ubuntu:~/B23041316$ ./hello2
c2.o
..
Makefile
find_min.sh
c1.c
test.txt
count_exefile1.sh
demo
c2.c
number_check.sh
.
prime_check.sh
time_check.sh
hello1
hello2
xuxing@ubuntu:~/B23041316$ make clean
rm -rf *.o
```

（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。

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>  // 包含chdir()（切换目录）和getcwd()（获取当前目录）函数

int main()
{
    char buf[1024] = {0};  // 存储切换前的工作目录
    char buf2[1024] = {0}; // 存储切换后的工作目录（可省略，直接再次调用getcwd）

    // 1. 获取切换前的当前工作目录，判断是否成功
    if (getcwd(buf, 1024) == NULL)
    {
        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);

    // 2. 切换工作目录到"/home"（可根据需求修改目标目录，如"/tmp"）
    if (chdir("/home") < 0)
    {
        printf("error:无法切换到/home目录（无权限或目录不存在）\n");
        return -1;
    }
    else
    {
        printf("目录切换成功!\n");
    }

    // 3. 获取切换后的工作目录，验证切换结果
    if (getcwd(buf2, 1024) == NULL)
    {
        printf("error:无法获取切换后的目录\n");
        return -1;
    }
    printf("切换后的工作目录:%s\n", buf2);

    return 0;
}
```

```
xuxing@ubuntu: ~/B23041316
xuxing@ubuntu:~/B23041316$ gedit c3.c
xuxing@ubuntu:~/B23041316$ make hello3
cc -c c3.c  # 行首为Tab键，编译c3.c生成c3.o
cc -o hello3 c3.o  # 行首为Tab键，链接c3.o生成hello3
xuxing@ubuntu:~/B23041316$ ls
c1.c  c3.o              find_min.sh  hello3           prime_check.sh
c2.c  count_exefile1.sh  hello1      Makefile         test.txt
c3.c  demo               hello2      number_check.sh  time_check.sh
xuxing@ubuntu:~/B23041316$ ./hello3
切换前的工作目录:/home/xuxing/B23041316
目录切换成功!
切换后的工作目录:/home
xuxing@ubuntu:~/B23041316$ make clean
rm -rf *.o
xuxing@ubuntu:~/B23041316$
```

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。