

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;
}
```

Make sure your filename is c1.c.
We can use the following makefile.

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



A terminal window showing the execution of the provided makefile. The user is in a directory named '桌面' (Desktop). The commands and their outputs are as follows:

```
hello@hello-virtual-machine:~/桌面$ gcc -c -o c1.o c1.c
hello@hello-virtual-machine:~/桌面$ ls
2.1.sh  2.3.sh  2.5.sh  实验2  实验4      archlab-handout.tar  c1.o
2.2.sh  2.4.sh  实验1  实验3  archlab-handout  c1.c
hello@hello-virtual-machine:~/桌面$ rm -rf *.o
hello@hello-virtual-machine:~/桌面$ ls
2.1.sh  2.3.sh  2.5.sh  实验2  实验4      archlab-handout.tar
2.2.sh  2.4.sh  实验1  实验3  archlab-handout  c1.c
```

```

hello@hello-virtual-machine:~/桌面$ gcc -o c1 c1.c
hello@hello-virtual-machine:~/桌面$ ./c1 1.txt
open source 1.txt failed
hello@hello-virtual-machine:~/桌面$ ./c1 2.1.sh
#!/bin/bash

hour=`date +%H`

case $hour in

0[1-9] | 1[01] )

echo "Good morining !!"

;;

1[234567] )

echo "Good afternoon !!"

;;

```

2、 Write a C program that displays all the file names in the current directory. 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
```

```
hello@hello-virtual-machine:~/桌面$ gcc -c -o c2.o c2.c
hello@hello-virtual-machine:~/桌面$ ls
2.1.sh  2.4.sh  实验2  archlab-handout  c1.c  c2.o
2.2.sh  2.5.sh  实验3  archlab-handout.tar  c2    c3
2.3.sh  实验1  实验4  c1                c2.c  c3.c
hello@hello-virtual-machine:~/桌面$ rm -rf *.o
hello@hello-virtual-machine:~/桌面$ ls
2.1.sh  2.4.sh  实验2  archlab-handout  c1.c  c3
2.2.sh  2.5.sh  实验3  archlab-handout.tar  c2    c3.c
2.3.sh  实验1  实验4  c1                c2.c
```

```
hello@hello-virtual-machine:~/桌面$ gcc -o c2 c2.c
hello@hello-virtual-machine:~/桌面$ ./c2 /home
.
..
b22040721
hello
```

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;
}

```

Make sure your filename is c3.c.
We can use the following makefile.

```

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

```

```

hello@hello-virtual-machine:~/桌面$ touch c3.c
hello@hello-virtual-machine:~/桌面$ gcc -c -o c3.o c3.c
hello@hello-virtual-machine:~/桌面$ ls
2.1.sh  2.4.sh  实验2  archlab-handout      c1.c  c3.c
2.2.sh  2.5.sh  实验3  archlab-handout.tar  c2    c3.o
2.3.sh  实验1  实验4  c1                  c2.c
hello@hello-virtual-machine:~/桌面$ rm -rf *.o
hello@hello-virtual-machine:~/桌面$ ls
2.1.sh  2.3.sh  2.5.sh  实验2  实验4      archlab-handout.tar  c1.c  c2.c
2.2.sh  2.4.sh  实验1  实验3  archlab-handout  c1                  c2    c3.c

```

```

hello@hello-virtual-machine:~/桌面$ gcc -o c3 c3.c
hello@hello-virtual-machine:~/桌面$ ./c3
/home/hello/桌面
success
/home

```