

# Experiment3 C programming experiment

## Experimental purpose:

Further use the basic syntax of C programming language in Linux system, deepen the understanding of the knowledge.

### (1) Task 1

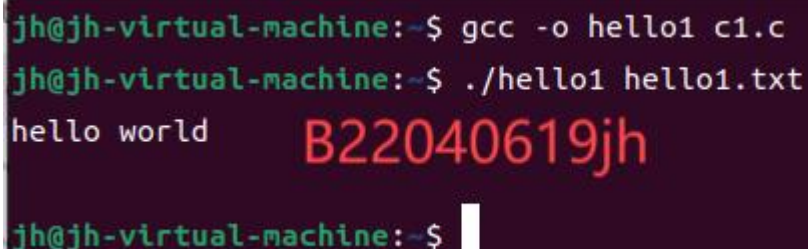
(1) Write a C program that uses standard I/O libraries to display the contents of text files. The program is compiled and linked by the make tool, which requires the generation of the.o file first, and then the generation of the executable file, and the function of deleting the intermediate file (.o)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



```
jh@jh-virtual-machine:~$ gcc -o hello1 c1.c
jh@jh-virtual-machine:~$ ./hello1 hello1.txt
hello world
jh@jh-virtual-machine:~$
```

B22040619jh

## (2) Task 2

---

(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
```

```
jh@jh-virtual-machine:~$ gcc -o hello1 c2.c
```

```
jh@jh-virtual-machine:~$ ./hello1
```

```
Segmentation fault (core dumped)
```

```
jh@jh-virtual-machine:~$ ./hello1 ~
```

```
hello1
```

```
c2.c
```

```
linklab.tar
```

```
c2.o
```

```
archlab-handout
```

```
printf.so
```

```
datalab-handout.tar
```

```
.ssh
```

```
111.c
```

```
Videos
```

```
datalab-handout
```

```
.pam_environment
```

```
B22040619 B22040619jh
```

```
B22040619.111
```

```
hello1.txt
```

```
.local
```

```
Public
```

```
Untitled 1.doc
```

### (3) Task 3

(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
```

```
jh@jh-virtual-machine:~$ gcc -o hello1 c3.c
jh@jh-virtual-machine:~$ ./hello ~\
>
bash: ./hello: No such file or directory
jh@jh-virtual-machine:~$ ./hello ~
bash: ./hello: No such file or directory
jh@jh-virtual-machine:~$ ./hello1 ~
/home/jh      B22040619jh
success
/home
jh@jh-virtual-machine:~$
```