任务1：在工作目录(20373007)下编写makefile完成以下功能：

1. 执行make后，可以将同目录下的hello_os.c编译为hello_os.o(仅编译但不链接).
2. 用make clean命令后，可以将编译出来的hello_os.o清除

```
all:
    gcc hello_os.c -o os_hello

clean:
    rm os_hello
```

任务2 创建lab0-exam.sh完成以下功能

1. #!/bin/bash 后的第一行为make命令，调用前文所写的Makefile创建hello_os.o
2. 在lab0-exam.sh文件同目录下创建一个hello_os空文件
3. 在lab0-exam.sh文件同目录下创建一个hello_os_dir的目录
4. 将第一步的make命令创建的hello_os.o复制到hello_os_dir中
5. 再复制一份hello_os.o文件到hello_os_dir文件夹内
6. 清除lab0-exam.sh文件同目录下的hello_os.o文件
7. 最后往hello_os.txt中输出hello_os.c中所有包含“os_hello”（区分大小写）的行的"os_hello"以左的内容。如果一行出现多个，则输出第一次出现以左的部分。

```
#!/bin/bash
touch hello_os
mkdir hello_os_dir
cp os_hello ./hello_os_dir
mv ./hello_os_dir/os_hello ./hello_os_dir/hello_os
cp os_hello ./hello_os_dir
rm os_hello

grep -in os_hello hello_os.c > hello_os.txt
```

---

1. 在 src 目录下，存在一个未补全的 Makefile 文件，借助刚刚掌握的 Makefile 知 识，将其补全，以实现通过 make 命令触发 src 目录下的 palindrome.c 文件的编译链接 的功能，生成的可执行文件命名为 palindrome。

```
palindrome: palindrome.c
    gcc -o palindrome palindrome.c
```

---

2. 在 src/sh_test 目录下，有一个 file 文件和 hello_os.sh 文件。hello_os.sh是一个未完成的脚本文档，请同学们借助 shell 编程的知识，将其补完，以实现通过命令bash hello_os.sh AAA BBB，在hello_os.sh 所处

的目录新建一个名为 BBB 的文件，其内容为 AAA 文件的第 8、32、128、512、1024 行的内容提取(AAA 文件行数一定超过 1024行)。[注意：对于命令 bash hello_os.sh AAA BBB，AAA 及 BBB 可为任何合法文件的名称，例如 bash hello_os.sh filehello_os.c，若已有 hello_os.c 文件，则将其原有内容覆盖]

```
#!/bin/bash
sed -n '8p' $1 > $2
sed -n '32p' $1 >> $2
sed -n '128p' $1 >> $2
sed -n '512p' $1 >> $2
sed -n '1024p' $1 >> $2
```

3. 补全后的 palindrome.c、Makefile、hello_os.sh 依次复制到路径 dst/palindrome.c, dst/Makefile, dst/sh_test/hello_os.sh [注意：文件名和路径必须与题目要求相同]

```
cp palindrome.c ./dst/palindrome.c
cp Makefile dst/Makefile
cp hello_os.sh dst/sh_test/hello_os.sh
```

4. 在 Lab0 工作区 ray/sh_test1 目录中，含有 100 个子目录file1~file100，还存在一个名为 changefile.sh 的文件，将其补完，以实现通过命令 bash changefile.sh，可以删除该目录内 file71~file100 共计 30 个子目录，将 file41~file70 共计 30 个子目录重命名为 newfile41~newfile70。

```
#!/bin/bash
a=1
while [ $a -le 100 ]
do
    if [ $a -gt 70 ]       #if loop variable is greater than 70
    then rm -rf file$a

    elif [ $a -gt 40 ]         # else if loop variable is great than 40
    then mv file$a newfile$a

    fi
            #don't forget change the loop variable
        a=$[$a+1]
done
```

5. 在 Lab0 工作区的 ray/sh_test2 目录下，存在一个未补全的 search.sh 文件，将其补完，以实现通过命令 bash search.sh file int result，可以在当前目录下生成 result 文件，内容为 file 文件含有 int 字符串所在的行数，即若有多行含有 int 字符串需要全部输出。[匹配时大小写不忽略]

```
grep -n $2 $1 | awk -F: '{print $1}' > $3
```

在 Lab0 工作区的 csc/code 目录下，存在 fibo.c、main.c，其中 fibo.c 有点小问题，还有一个未补全的 modify.sh 文件，将其补完，以实现通过命令 bash modify.sh fibo.c char int，可以将 fibo.c 中所有的 char 字符串更改为 int 字符串。

```
#!/bin/bash
sed -i "s/$2/$3/g" $1
```

Lab0 工作区的 csc/code/fibo.c 成功更换字段后 (bash modify.sh fibo.c char int)，现已有 csc/Makefile 和 csc/code/Makefile，补全两个 Makefile 文件，要求在 csc 目录下通过命令 make 可在 csc/code 目录中生成 fibo.o、main.o，在 csc 目录中生成可执行文件 fibo，再输入命令 make clean 后只删除两个.o 文件。[注意：不能修改 fibo.h 和 main.c 文件中的内容，提交的文件中 fibo.c 必须是修改后正确的 fibo.c，可执行文件 fibo 作用是输入一个整数 n(从 stdin 输入 n)，可以输出斐波那契数列前 n 项，每一项之间用空格分开。比如 n=5，输出 1 1 2 3 5] 要求成功使用脚本文件 modify.sh 修改 fibo.c，实现使用 make 命令可以生成.o 文件和可执行文件，再使用命令 make clean 可以将.o 文件删除，但保留 fibo 和.c 文件。

csc/Makefile:

```
all:
    cd ./code && make fibo
.PHONY clean:
    rm ./code/*.o
```

csc/code/Makefile:

```
fibo:
    gcc -c fibo.c -I../include
    gcc -c main.c -I../include
    gcc fibo.o main.o -o ../fibo
```

sed 文本处理工具

Linux sed 命令是利用脚本来处理文本文件，sed 可依照脚本的指令来处理、编辑文本文件。Sed 主要用来自动编辑一个或多个文件、简化对文件的反复操作、编写转换程序等。

相关选项

sed 2 sed [选项] '命令' 输入文本 3 选项（常用）： 4 -n：安静模式，只显示经过 sed 处理的内容。否则显示输入文本的所有内容。 5 -i：直接修改读取的档案内容，而不是输出到屏幕。否则，只输出不编辑。 6 命令（常用）： 7 [行号]a[内容]： 新增，在行号后新增一行相应内容。行号可以是“数字”，在这一行之后新增， 8 也可以是“起始行，终止行”，在其中的每一行后新增。 9 当不写行号时，在每一行之后新增。使用 $ 表示最后一行。后面的命令同理。 10 [行号]c[内容]：取代。用内容取代相应行的文本。 11 [行号]i[内容]：插入。在当前行的上面插入一行文本。 12 [行号]d：删除当前行的内容。 13 [行号]p：输出选择的内容。通常与选项-n 一起使用。 14 s/re（正则表达式）/string：将 re 匹配的内容替换为 string。

-n, --quiet, --silent 取消自动打印模式空间 -e 脚本, --expression=脚本 添加“脚本”到程序的运行列表 -f 脚本文件, --file=脚本文件 添加“脚本文件”到程序的运行列表 --follow-symlinks 直接修改文件时跟随软链接 -i[扩展名], --in-place[=扩展名] 直接修改文件(如果指定扩展名就备份文件) -l N, --line-length=N 指定“l”命令的换行期望长度 --posix 关闭所有 GNU 扩展 -r, --regexp-extended 在脚本中使用扩展正则表达式 -s, --separate 将输入文件视为各个独立的文件而不是一个长的连续输入 -u, --unbuffered 从输入文件读取最少的数据，更频繁的刷新输出 --help 打印帮助并退出 --version 输出版本信息并退出 -a :新增， a 的后面可以接字串，而这些字串会在新的一行出现(目前的下一行)～ -c :取代， c 的后面可以接字串，这些字串可以取代 n1,n2 之间的行！ -d :删除，因为是删除啊，所以 d 后面通常不接任何咚咚； -i :插入， i 的后面可以接字串，而这些字串会在新的一行出现(目前的上一行)； -p :列印，亦即将某个选择的资料印出。通常 p 会与参数 sed -n 一起运作～ -s :取代，可以直接进行取代的工作哩！通常这个 s 的动作可以搭配正规表示法

用法:

查看功能 查看passwd文件的第5到第8行内容 sed -n '5,8 p' passwd 查看passwd文件中以roo开头的行 sed -n '/^roo/ p' passwd 忽略大小写，对含有root字符的行打印出来 sed -n '/root/I p' passwd

查找功能 查找passwd文件中有/bin/bash字符串的行 sed -n '%/bin/bash% p' passwd

以行为单位的新增 在文件passwd上的第四行后面添加新字符串it-test sed -e 4a\it-test passwd | head passwd第1前追加huawei sed '1 i\huawei ' passwd |head huawei

替换功能 passwd第三行替换为redhat sed '3 c\redhat' passwd

查找及替换功能 将passwd的5到10的bin字符串查找出来替换为aaaa sed -n '5,10 s/bin/aaaa/ p' passwd |head

对原文件操作 删除原文件第1行 sed -i '1 d' passwd

---

grep 文本搜索工具

文本搜索工具，根据用户指定的“模式（过滤条件）”对目标文本逐行进行匹配检查，打印匹配到的行.

相关选项

-i 忽略大小写 -o 仅显示匹配到的字符串本身 -v 显示不能被匹配到的行 -E 正则 -q 不输出任何信息 -n 显示匹配行与行数 -c 只统计匹配的行数

---

rm

rm -r 递归删除 rm -f 强制删除

man 查看帮助

diff 比较文件差异

diff [选项] 文件 1 文件 2 常用选项 -b 不检查空白字符的不同 -B 不检查空行 -q 仅显示有无差异，不显示详细信息