

2022第一次学习任务

内容比较多:)

A.环境配置

安装虚拟机（这部分我很早就完成了，具体步骤就不详细说明了）

- 安装vm及Linux系统镜像

这一部分步骤比较清晰，主要参考[VMware虚拟机安装Ubuntu 2022最新版详细图文安装教程](#)

如果个人更倾向于CentOS系统镜像也可以选择CentOS

两者的主要区别在于一些指令的使用上，比如CentOs中新建的普通用户是没有sudo权限的，

如果想让普通用户拥有sudo权限需要在/etc/sudoers文件中添加用户的权限，而Ubuntu系统

普通用户想要使用sudo权限 直接使用sudo +命令行的方式就可以了。还有一个就是在使用上

Ubuntu比CentOS更美观，图形化更友好一点。

- 安装Linux操作系统

Ubuntu:<https://zhuanlan.zhihu.com/p/141033713>

这里有一个要注意的地方是，虚拟机创建的时候内存最好找一个空硬盘，大小在100GB以上，

不然后面可能安装不下,若硬盘内存较小，建议安装时进行[虚拟机磁盘手动分区](#)，区别就在于增

加了一个磁盘分区的步骤。

安装OneAPI

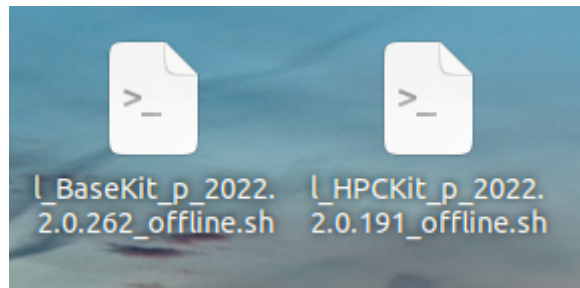
- OneAPI是什么

- 统一的开发工具组合和软件接口
- 可以让开发人员在CPU、GPU、FPGA、AI加速器等计算机架构上实现高效开发、任意扩展。

- 下载和安装OneAPI

- intel网站[下载地址](#)

注意需要下载base Toolkit 与 HPC Toolkit 两个文件



之后具体步骤参考[Intel® oneAPI Base Toolkit+Intel® oneAPI HPC Toolkit安装教程](#)

- 下载完毕后，需要添加环境变量

为避免每次使用都需要配置环境变量，可以将如图命令添加到~/.bashrc文件中。setvars.sh的具体路径以自己的为准

```
source /opt/intel/oneapi/setvars.sh intel64
```

环境变量加载信息如下：

```
:: initializing oneAPI environment ...
bash: BASH_VERSION = 5.1.16(1)-release
args: Using "$@" for setvars.sh arguments: intel64
:: advisor -- latest
:: ccl -- latest
:: clck -- latest
:: compiler -- latest
:: dal -- latest
:: debugger -- latest
:: dev-utilities -- latest
:: dnnl -- latest
:: dpcpp-ct -- latest
:: dpl -- latest
:: inspector -- latest
:: intelpython -- latest
:: ipp -- latest
:: ippcp -- latest
:: ipp -- latest
:: itac -- latest
:: mkl -- latest
:: mpi -- latest
:: tbb -- latest
:: vpl -- latest
:: vtune -- latest
:: oneAPI environment initialized ::
```

- 检验安装结果(只展示了icc)

```
non@non-virtual-machine:/$ gcc -v
gcc version 2021.6.0 (gcc version 11.2.0 compatibility)
```

这样的话OneAPI就算安装成功了

B.实操部分

Linux基础入门

这一部分学习了一些Linux基础知识，基本上是一些命令的实操。下面列出几块相对常用的命令。

1. 快捷键

快捷键

Tab：命令补全，当你忘记某个命令的全称时可以只输入它的开头的一部分，然后按下 Tab 键就可以得到提示或者帮助完成。

Ctrl+c：强行终止当前程序。

Ctrl+d：键盘输入结束或退出终端。

Ctrl+s：暂停当前程序，暂停后按下任意键恢复运行。

Ctrl+z：将当前程序放到后台运行，恢复到前台为命令fg。

Ctrl+a：将光标移至输入行头，相当于Home键。

Ctrl+e：将光标移至输入行末，相当于End键。

Ctrl+k：删除从光标所在位置到行末。

Alt+Backspace：向前删除一个单词。

Shift+PgUp：将终端显示向上滚动。

Shift+PgDn：将终端显示向下滚动。

帮助手册

在 Linux 环境中，如果你对某个命令感到疑惑，可以使用man命令来获取帮助，它是Manual pages的缩写。用户可以通过执行man命令调用手册页。

你可以使用如下方式来获得某个命令的说明和使用方式的详细介绍：

```
$ man <command_name>
```

2. 安装、更新与卸载

- 安装只需要执行以下指令

```
sudo apt-get install htop
```

htop是一款交互式进程软件

- 更新软件源

```
sudo apt-get update
```

- 更新软件包

```
sudo apt-get upgrade
```

- 卸载前面的htop软件

```
sudo apt-get remove htop
```

- 可以一并删除配置文件

```
sudo apt-get purge htop
```

3. 查看文件权限

- ls 命令不仅可以列出并显示当前目录的文件，还可以用来查看文件的权限。

```
$ ls -l
```

输出:

文件类型和权限 链接数 所有者 所属用户组 文件大小 最后修改时间 文件名

这里特别解释一下文件类型和权限:

第一位是指文件类型, 例如 'd' 表示目录, '-' 表示普通文件, 等等。

后面的九位可以分成三组, 每组三位分别表示读权限/写权限/执行权限, 三组分别对应的是 拥有者权限/所属用户组权限/其他用户权限。

4. 基本目录操作

- 常用 `cd` 来切换目录, 使用 `.` 表示当前目录, `..` 表示上一级目录, `~` 表示当前用户的home目录, `-` 表示上次所在目录。
- 使用 `pwd` 命令可以获得当前所在路径 (绝对路径), 例如进入上一级目录:

```
$ cd ..
```

5. 基本文件操作(创建、复制和删除等)

- `touch`命令可以创建一个新的空白文件, 例如创建一个名为myfile的文件。

```
$ touch myfile
```

- 使用 `mkdir` 命令可以创建一个新的目录, 例如创建一个newdir的目录。

```
$ mkdir newdir
```

- 使用 `cp` 命令复制一个文件到指定目录。

```
$ cp myfile ../../newdir
```

- 在复制文件的基础上加上 `-r`参数即可复制文件

```
$ cp -r dir newdir
```

- 用`rm` (remove files or directori) 命令删除文件。同样地, 删除目录也需要加上 `-r`参数。

```
$ rm myfile
```

- 移动及重命名文件都使用 `mv` 命令。

移动文件格式为: `mv 源目录文件 目的目录`

注意 此处的`mv`命令为剪切文件

```
$ mv myfile newdir
```

重命名文件为: `mv 旧的文件名 新的文件名`

```
$ mv myfile newfile
```

- 搜索 `find` 命令, 可以根据文件各个属性来搜索, 其格式为:

```
$ find [path] [option] [action]
```

例如:

```
$ sudo find /usr/ -name myfile
```

6. tar工具的使用 (仅部分)

- 创建一个tar包

```
tar -cf newtar.tar ./Desktop
```

其中, `-c`参数表示创建, `-f`参数表示指定文件名, 必须跟在文件名之前。这个tar包跟原大小相同。

- 查看一个tar包:

```
tar -tf newtar.tar
```

- 解包一个tar包到指定目录

```
tar -xf newtar.tar -C newdir
```

7. 终止

- 在参数上可以通过选择信号值来决定以何种方式终止程序，信号值可以这样查看

```
kill -l
```

- 可以这样终止程序：

```
kill -signal pid
```

Vim入门

这一部分主要学习了Vim编辑器的一些入门操作，以及普通模式、插入模式等多种模式。

一、六种模式

1. 普通模式

在普通模式中，用的编辑器命令，比如移动光标，删除文本等等。是Vim启动后的默认模式。在其他模式下，按下ESC即可返回普通模式。

2. 插入模式

在这个模式中，大多数按键都会向文本缓冲区中插入文本。在普通模式中可以按a（append / 追加）键或者i（insert / 插入）键进入插入模式。

3. 可视模式

这个模式与普通模式比较相似。但是移动命令会扩大高亮的文本区域。高亮区域可以是字符、行或者是一块文本。当执行一个非移动命令时，命令会被执行到这块高亮的区域上。在普通模式下，按下v, V, Ctrl键+v即可进入。

4. 选择模式

这个模式和无模式编辑器的行为比较相似（Windows标准文本控件的方式）。这个模式中，可以用鼠标或者光标键高亮选择文本，不过输入任何字符的话，Vim会用这个字符替换选择的高亮文本块，并且自动进入插入模式。

5. 命令行模式

在命令行模式中可以输入会被解释成并执行的文本。例如执行命令（: 键），搜索（/和? 键）或者过滤命令（!键）。在命令执行之后，Vim返回到命令行模式之前的模式，通常是普通模式。在普通模式下，按下: 即可进入。

6. Ex模式

这和命令行模式比较相似，在使用:visual命令离开Ex模式前，可以一次执行多条命令。在进入Vim时，加上-e参数即可进入，例如：vim -e

二、普通模式——游标移动

1. 字符级移动

在普通模式下，可以使用方向键或者h,j,k,l进行移动，如下表：

按键	方向
h, ←	左
l, →	右
j, ↓	下
k, ↑	上

2. 行内移动

在行内以移动游标时，可以使用下面的命令来实现：

按键	说明
w	右移到下一个单词的开头
e	右移到下一个单词的末尾
b	左移到前一个单词的开头
0	右移到本行的开始
\$	右移到本行的末尾
^	左移到本行的第一个非空字符

3. 页级移动

可以在相邻页之间移动：

按键	说明
<code>crtl + f</code>	前移一页
<code>crtl + b</code>	后移一页

4. 页内移动

当我们到达指定页时，可以在页内快速移动：

按键	说明
H	将光标移到该页的起始行
M	将光标移到该页的中间行
L	将光标移到该页的末尾行

5. 其他移动

Vim提供了很多的移动方式，这里再列举一些大范围移动：

按键	说明
*	将光标移到该单词的下一个位置
#	将光标移到该单词的上一个位置
g	将光标移到文件的第一行
G	将光标移到文件的末尾行

三、普通模式——文档编辑

1. 搜索

Vim的搜索方法非常简单，只需要输入/+内容就可以进行搜索，其命令具体如下：

指令	说明
<code>\ + str</code>	向下搜索字符串str
<code>n</code>	继续搜索
<code>N</code>	反向搜索
<code>? + str</code>	反向搜索字符串str

2. 替换

Vim的常规替换命令主要由s和c完成，但替换命令之后，会进入插入模式，按下ESC即可返回普通模式，替换命令部分如下：

指令	说明
<code>s</code>	用输入的正文替换游标所在的字符
<code>ns</code>	用输入的正文替换游标右侧n个字符
<code>c\$</code>	用输入的正文替换从游标开始到行尾的所有字符
<code>c0</code>	用输入的正文替换从本行开始到游标的所有字符

3. 删除

Vim的删除命令主要由d和x完成，如下：

指令	说明
x	删除光标所在的字符
X	删除光标所在前一个字符
dw	删除光标右的单词
dd	删除光标所在行
d0	删除光标到行的开始
d\$	删除光标到行的末尾

4. 删除

Vim中删除操作其实与剪切操作类似，会将内容复制到内存缓冲区，当然，也可以通过命令自行复制而不删除内容：

指令	说明
yy	复制光标所在行到缓冲区
nyy	复制n行到缓冲区

5. 粘贴

将内容复制到缓冲区后，可以使用命令将内容粘贴：

指令	说明
p	粘贴到光标后
P	粘贴到光标前

6. 撤销

通过指令，可以撤销上一次的操作：

指令	说明
u	撤销上次操作

7. 重复

通过指令，可以重复上一次的操作：

指令	说明
u	重复上次操作

四、插入模式

1. 进入插入模式

进入插入模式还可以通过命令：

指令	说明
i	在光标处插入
I	在行首插入
a	在光标右侧插入
A	在行末插入
o	在当前行后添加一行插入
O	在当前行前添加一行插入

2. 光标移动

当然，也可以尝试在插入模式时移动，但此时h,j,k,l已不能移动，只能使用方向键进行小幅移动：

按键	方向
←	左
→	右
↓	下
↑	上

3. 退出插入模式

进入插入模式后，在编辑内容时，往往需要应用其他命令，这时就需要退出插入模式，进入普通模式，使用命令进行操作：

指令	说明
ESC	退出插入模式
Ctrl + [退出插入模式

五、命令模式

输入 vim : 进入命令模式

1. 打开文件

在进入Vim之后，如需打开其他文件,则：

e 文件名

2. 保存文件

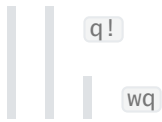
编辑之后，我们可以使用w命令保存文件，也可以另存为其他文件：

:w other-ehpc.txt

3. 退出Vim

退出Vim时，如果未作修改，需要使用进行退出，如果有修改内容，但不保存，则可以使用强制退出，如果想要保存内容，并退出，可以使用命令。

q



4. 行号操作

命令模式下，我们可以直接输入行号进行跳转

`:n`

跳转到第n行

通过对行号的使用，我们可以结合其他命令做到：

`:nw 文件名` 将行写入文件n

`:m,nw 文件名` 将到行写入文件m n

`:m,.w 文件名` 将到当前行写入文件m

`:.,$w 文件名` 将当前行到最后一行写入文件

5. 文件操作

前面指出了可以使用打开文件，写文件，Vim还有更多的文件操作：

`ew`

`:r 文件名` 读取文件并插入到光标之后

`:f 文件名` 重命名文件

`:f` 输出当前文件名称和状态

6. 内容替换

Vim除了普通模式下能够进行替换操作，命令模式下也可以进行替换操作：

`:%s/str1/str2/` 用替换行中首次出现的 `str2` 替换 `str1`

通过参数可以达到替换所有的操作：

用替换行中出现的 `g:s/str1/str2/gstr2str1`

还可以指定行数，例如：用字符串 替换正文中所有出现的字符串

上述操作也可以通过实现：`1,$ s/str1/str2/gstr2str1:g/str1/s//str2/g`

六、进阶命令

1. 文件恢复

恢复意外退出而没有保存的文件

`:recover`

2. 选项设置

通过命令可以设置选项，比如：`:set`

选项	说明
autoindent	自动缩进
number	显示行号
ignorecase	忽略正则表达式的大小写
ruler	显示光标位置

3. 分屏

Vim中可以使用命令进行分屏操作：

命令	说明
split ,sp	水平分屏
vsplit,vsp	垂直分屏

也可以在启动Vim时分屏，在启动时加入参数：

打开 file1 和 file2 ，垂直分屏

打开 file1 和 file2 ，水平分屏

各屏幕间移动需要使用快捷键：

```
vim -On file1 file2...vim -on file1 file2...
```

命令	说明
<code>Ctrl + w + h</code>	当前屏的左分屏
<code>Ctrl + w + l</code>	当前屏的右分屏
<code>Ctrl + w + j</code>	当前屏的下分屏
<code>Ctrl + w + k</code>	当前屏的上分屏

Shell入门教程


一、第一个shell脚本

- 打开终端，输入以下命令,创建一个新的shell脚本

```
vim hello.sh
```

- 在脚本中以插入模式输入以下代码,并强制保存后退出

```
#!/bin/bash
echo "Hello world !"
```



```
Terminal - ehpc@40908feec652: ~
File Edit View Terminal Tabs Help
#!/bin/bash
echo "Hello World !"

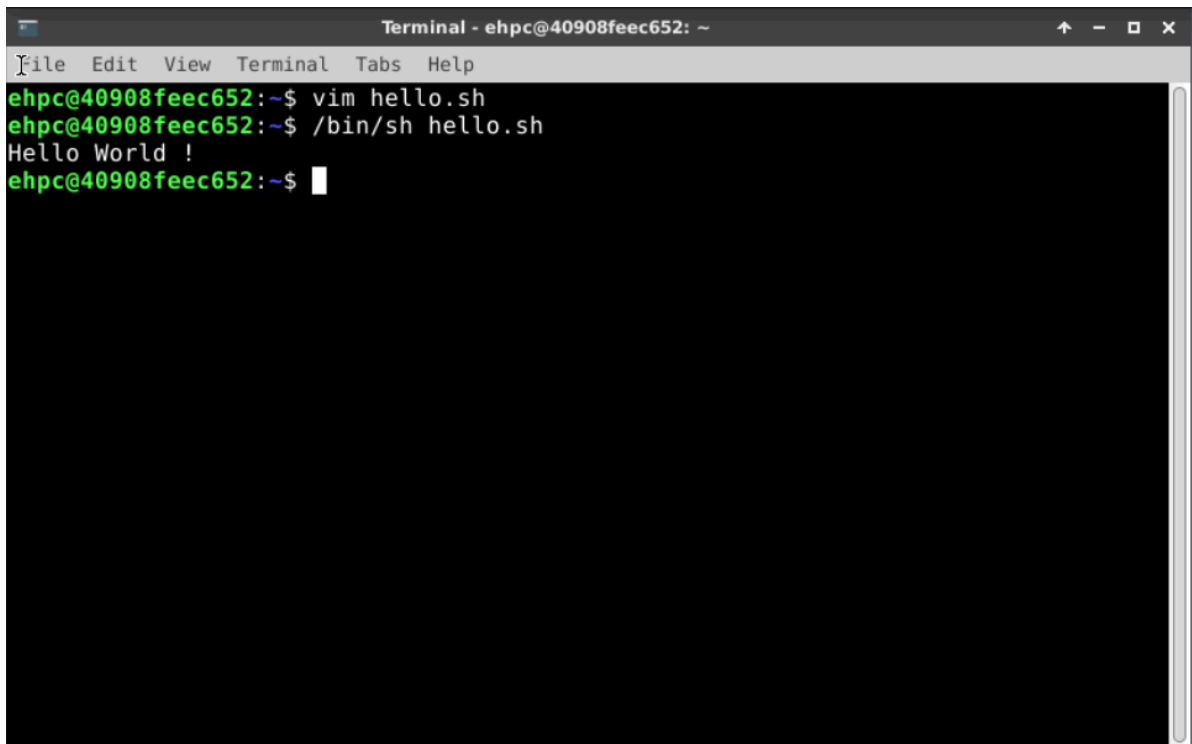
"hello.sh" 2L, 33C 2,20 All
```

脚本的第一行#! 是一个标记，它的用处是告诉系统这个脚本需要使用什么解释器来执行，我们这里选用的是bash。

第二行的echo命令是将想要输出的字符串打印到屏幕上，在我们的脚本是将hello world打印到屏幕上。

- 再输入运行命令，运行shell脚本文件

```
/bin/sh hello.sh
```



```
Terminal - ehpc@40908feec652: ~
File Edit View Terminal Tabs Help
ehpc@40908feec652:~$ vim hello.sh
ehpc@40908feec652:~$ /bin/sh hello.sh
Hello World !
ehpc@40908feec652:~$
```

成功运行！

二、Shell基本语法

1. shell的变量

变量名与等号之间不能有空格，命名只能使用英文字母、数字、下划线，不能使用Shell中的关键字。

```
#!/bin/bash

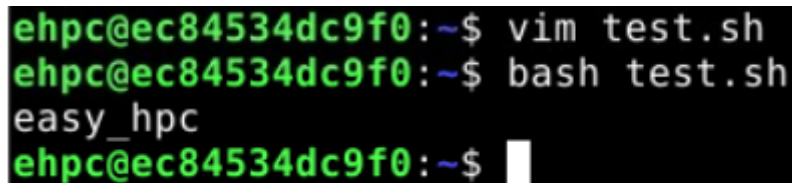
ehpc="easy_hpc"

echo $ehpc
```

首先我们使用以上代码来定义一个变量ehpc，然后用echo命令将它打印出来。

这里可以看到，使用一个定义的变量，只需要在变量名之前加美元符号。

shell脚本运行结果如下:



```
ehpc@ec84534dc9f0:~$ vim test.sh
ehpc@ec84534dc9f0:~$ bash test.sh
easy_hpc
ehpc@ec84534dc9f0:~$
```

2. shell的运算符

将之前的 test.sh 文件清空，添加以下代码保存后运行

```
#!/bin/bash
a=5
b=10

val=`expr $a + $b`
echo "a + b : $val"

val=`expr $a - $b`
echo "a - b : $val"

val=`expr $a \* $b`
echo "a * b : $val"

val=`expr $b / $a`
echo "b / a : $val"

val=`expr $b % $a`
echo "b % a : $val"
```

在上面的代码中，expr是一个表达式计算工具，使用它能够完成表达式的求值操作。

特别注意表达式前后的引号为反引号。所以可以看到，我们这五个例子分别展示的是加、减、乘、除、求余。

shell脚本运行结果如下:

```
ehpc@ec84534dc9f0:~$ vim test.sh
ehpc@ec84534dc9f0:~$ bash test.sh
a + b : 15
a - b : -5
a * b : 50
b / a : 2
b % a : 0
ehpc@ec84534dc9f0:~$
```

3. shell的输出命令

Shell中的echo命令也是类似PHP语言中的功能，即将字符串输出，在test.sh中输入以下代码

```
#!/bin/bash
echo "ehpc is easy hpc"
echo "\"ehpc is easy hpc\""
echo `date`
```

显示普通的字符串

显示转义字符

显示转义字符

shell脚本运行结果如下

```
ehpc@e4fb1a499578:~$ vim test.sh
ehpc@e4fb1a499578:~$ bash test.sh
ehpc is easy hpc
"ehpc is easy hpc"
Sun Oct 9 09:47:36 UTC 2022
ehpc@e4fb1a499578:~$
```

除了echo输出命令之外，还有一个输出命令printf，printf命令可以格式化字符串，还可以指定字符串的宽度、左右对齐方式，下面我们将用一个例子来展示printf的功能。>

vim打开 test.sh，清空之前的代码，输入以下代码：

```
#!/bin/bash
printf "%-10s %-8s %-4s\n" name gender heightcm
printf "%-10s %-8s %-4.2f\n" ehpc male 187.1234
```

这段代码中，%-10s 指一个宽度为10个字符，任何字符都会被显示在10个字符宽的字符内，如果不足则自动以空格填充，超过也会将内容全部显示出来。%-4.2f 指格式化为小数，其中.2指保留2位小数。

运行结果如下

```
ehpc@5cf594fcb1a2:~$ vim test.sh
ehpc@5cf594fcb1a2:~$ bash test.sh
name          gender    heightcm
ehpc          male      187.12
ehpc@5cf594fcb1a2:~$
```

4. shell数组的使用

在test.sh.中输入以下代码,在下面的代码中,我们建立了一个数组array_test,并依次输出各个元素。

```
#!/bin/bash
array_test=(E H P C)
echo "first element: ${array_test[0]}"
echo "second element: ${array_test[1]}"
echo "third element: ${array_test[2]}"
echo "fourth element: ${array_test[3]}"
```

运行结果如下

```
ehpc@5cf594fcb1a2:~$ vim test.sh
ehpc@5cf594fcb1a2:~$ bash test.sh
first element: E
second element: H
third element: P
fourth element: C
ehpc@5cf594fcb1a2:~$
```

三、流程控制

1. if-else语句

Shell 编程中的if else语句的语法格式如下所示:

```
if condition
then
    command
else
    command
fi
```

当条件为真时,执行then之后的代码,否则执行else之后的代码。

下面给出一个例子

```
#!/bin/sh
a=5
b=6
if [ $a == $b ]
then
    echo "a == b"
else
    echo "a !=b"
fi
```

运行结果如下

```
ehpc@5cf594fcb1a2:~$ bash test.sh
a !=b
ehpc@5cf594fcb1a2:~$
```

2. while 循环

```
while condition
do
    command
done
```

同样给出一个例子

```
#!/bin/sh
count=1
while(( $count<=3 ))
do
    echo $count
    let "count++"
done
```

初始化count变量为1，当count小于等于3时，打印出count值，并且增加count值；
当count大于3的时候，结束while循环。

运行结果如下

```
ehpc@5cf594fcb1a2:~$ vim text.sh
ehpc@5cf594fcb1a2:~$ bash text.sh
1
2
3
ehpc@5cf594fcb1a2:~$
```

3. for 循环

```
for var in item1 item2 ... itemN
do
    command
done
```

例子如下


```
#!/bin/sh
for str in 'ehpc is easy hpc'
do
    echo $str
done
```

依次输出字符串中的每一个字符

运行结果如下

```
ehpc@5cf594fcb1a2:~$ vim for.sh
ehpc@5cf594fcb1a2:~$ bash for.sh
ehpc is easy hpc
ehpc@5cf594fcb1a2:~$
```

4. until 循环

其格式如下所示

```
until condition
do
    command
done
```

例子如下:

```
#!/bin/bash
val=0
until [ ! $val -lt 5 ]
do
    echo $val
    val=`expr $val + 1`
done
```

这段代码将依次输出0到4，当val为5的时候，判断条件为真，结束循环。

运行结果如下

```
ehpc@5cf594fcb1a2:~$ vim until.sh
ehpc@5cf594fcb1a2:~$ bash until.sh
0
1
2
3
4
ehpc@5cf594fcb1a2:~$
```

四、函数的使用

1. 定义与调用函数

函数定义格式如下:

```
function_name ()
{
    statement
}
```

或者

```
function function_name ()
{
    statement
}
```

调用函数的格式如下所示，后面跟着的是传递给函数的参数。

```
function_name parm1 parm2...
```

同样地，我们可以用函数调用的形式编写hello world程序

```
function echo_hello()
{
    echo "hello world"
}
echo_hello
```

这段代码中我们还是实现了第一次课程中介绍的hello world程序，不同的是我们这次使用函数来实现。定义函数名为echo_hello，函数体内实现的是打印出hello world。定义完函数之后，我们直接键入函数名echo_hello来调用，最终也实现了在屏幕上打印hello world的效果。

2. 函数参数

这里给出一个例子

```
#!/bin/bash

test(){
    echo " $1 !"
    echo " $2 !"
    echo " $5 !"
    echo " $10 !"
    echo " ${10}!"
}

test 1 3 5 7 9 11 13 15 17 19
```

我们先定义了一个函数test，在函数体内，我们依次输出第一个参数、第二个参数、第五个参数和第十个参数。调用函数时，将参数跟在函数名之后。

这里特别注意，`$10` 并不能够获取第十个参数，获取第十个参数需要 `${10}`。

因为Shell规定当 $n \geq 10$ 时，需要使用 `${n}` 来获取参数。

于是可以得到运行结果如下

```
ehpc@3a6da61c7a61:~$ vim t.sh
ehpc@3a6da61c7a61:~$ bash t.sh
1 !
3 !
9 !
10 !
19!
ehpc@3a6da61c7a61:~$
```

五、输入输出重定向

1. shell 输出重定向

输出重定向是指命令的结果不再输出到显示器上，而是输出到其他地方，一般是文件中；这样做的好处就是可以将命令的结果保存起来。

符号	作用
<code>command >file</code>	以覆盖的方式，把 command 的正确输出结果输出到 file 文件中
<code>command >>file</code>	以追加的方式，把 command 的正确输出结果输出到 file 文件中

输出重定向的完整写法是：
其中 fd 为文件描述符，默认为1，也就是标准输出文件；

```
command fd>file
或者
command fd>>file
```

2. shell 输入重定向

输入重定向就是改变输入的方向，不再使用键盘作为命令输入的来源，而是使用文件作为命令的输入；

符号	说明
<code>command</code>	将 file 文件中的内容作为 command 的输入
<code>command <</code>	从标准输入（键盘）中读取数据，直到遇见分界符 END 才停止（分界符可以是任意的字符串，用户自己定义）
<code>command file1 file2</code>	将 file1 作为 command 的输入，并将 command 的处理结果输出到 file2

看下面例子，用 wc 命令来对文本进行统计，包括单词个数、行数、字节数，它的用法如下：

```
wc [选项] [文件名]
```

选项可为：
-c 统计字节数

-w 统计单词数

-l 统计行数

c语言入门和c++基础

由于大一整个学年都在学c和c++，自认为学得还可以了，这部分就默认会了吧

HPL和HPCG的安装

1.HPL

修改Make.test的参数

主要有以下三部分

```
ARCH          = test

#
# -----
# - HPL Directory Structure / HPL library -----
# -----
#
TOPdir         = /home/non/hpl-2.3
INCdir         = $(TOPdir)/include
BINDir         = $(TOPdir)/bin/$(ARCH)
LIBdir         = $(TOPdir)/lib/$(ARCH)
#
HPLlib         = $(LIBdir)/libhpl.a
##
```

```
# - Message Passing library (MPI) -----
# -----
# MPinc tells the C compiler where to find the Message Passing library
# header files, MPlib is defined to be the name of the library to be
# used. The variable MPdir is only used for defining MPinc and MPlib.
#
# MPdir        = /opt/intel/oneapi/mpi/latest
# MPinc        = -I$(MPdir)/include
# MPlib        = $(MPdir)/lib/release/libmpi.a
```

```
CC             = mpiicc
CCNOOPT       = $(HPL_DEFS)
OMP_DEFS      = -qopenmp
CCFLAGS       = $(HPL_DEFS) -O3 -w -ansi-alias -i-static -z noexecstack -z relro -z now -nocompchk -Wall
#
```

编译 make arch=test

！！切记要在之前设置环境变量

```
non@non-virtual-machine:~/hpl-2.3$ make arch=test
make -f Make.top startup_dir      arch=test
make[1]: 进入目录“/home/non/hpl-2.3”
mkdir -p include/test
mkdir -p lib
mkdir -p lib/test
mkdir -p bin
mkdir -p bin/test
make[1]: 离开目录“/home/non/hpl-2.3”
make -f Make.top startup_src      arch=test
make[1]: 进入目录“/home/non/hpl-2.3”
make -f Make.top leaf le=src/auxil      arch=test
make[2]: 进入目录“/home/non/hpl-2.3”
( cd src/auxil ; mkdir -p test )
( cd src/auxil/test ; \
    ln -fs /home/non/hpl-2.3/Make.test Make.inc )
make[2]: 离开目录“/home/non/hpl-2.3”
make -f Make.top leaf le=src/blas      arch=test
make[2]: 进入目录“/home/non/hpl-2.3”
( cd src/blas ; mkdir -p test )
( cd src/blas/test ; \
    ln -fs /home/non/hpl-2.3/Make.test Make.inc )
make[2]: 离开目录“/home/non/hpl-2.3”
make -f Make.top leaf le=src/comm      arch=test
make[2]: 进入目录“/home/non/hpl-2.3”
( cd src/comm ; mkdir -p test )
( cd src/comm/test ; \
    ln -fs /home/non/hpl-2.3/Make.test Make.inc )
make[2]: 离开目录“/home/non/hpl-2.3”
make -f Make.top leaf le=src/grid      arch=test
make[2]: 进入目录“/home/non/hpl-2.3”
( cd src/grid ; mkdir -p test )
( cd src/grid/test ; \
    ln -fs /home/non/hpl-2.3/Make.test Make.inc )
make[2]: 离开目录“/home/non/hpl-2.3”
make -f Make.top leaf le=src/panel      arch=test
make[2]: 进入目录“/home/non/hpl-2.3”
( cd src/panel ; mkdir -p test )
( cd src/panel/test ; \
    ln -fs /home/non/hpl-2.3/Make.test Make.inc )
make[2]: 离开目录“/home/non/hpl-2.3”
make -f Make.top leaf le=src/pauxil      arch=test
make[2]: 进入目录“/home/non/hpl-2.3”
( cd src/pauxil ; mkdir -p test )
( cd src/pauxil/test ; \
    ln -fs /home/non/hpl-2.3/Make.test Make.inc )
make[2]: 离开目录“/home/non/hpl-2.3”
make -f Make.top leaf le=src/pfact      arch=test
make[2]: 进入目录“/home/non/hpl-2.3”
( cd src/pfact ; mkdir -p test )
( cd src/pfact/test ; \
```

```

ln -fs /home/non/hpl-2.3/Make.test Make.inc )
make[2]: 离开目录"/home/non/hpl-2.3"
make -f Make.top leaf le=src/pgesv      arch=test
make[2]: 进入目录"/home/non/hpl-2.3"
( cd src/pgesv ; mkdir -p test )
( cd src/pgesv/test ; \
ln -fs /home/non/hpl-2.3/Make.test Make.inc )
make[2]: 离开目录"/home/non/hpl-2.3"
make[1]: 离开目录"/home/non/hpl-2.3"
make -f Make.top startup_tst      arch=test
make[1]: 进入目录"/home/non/hpl-2.3"
make -f Make.top leaf le=testing/matgen arch=test
make[2]: 进入目录"/home/non/hpl-2.3"
( cd testing/matgen ; mkdir -p test )
( cd testing/matgen/test ; \
ln -fs /home/non/hpl-2.3/Make.test Make.inc )
make[2]: 离开目录"/home/non/hpl-2.3"
make -f Make.top leaf le=testing/timer arch=test
make[2]: 进入目录"/home/non/hpl-2.3"
( cd testing/timer ; mkdir -p test )
( cd testing/timer/test ; \
ln -fs /home/non/hpl-2.3/Make.test Make.inc )
make[2]: 离开目录"/home/non/hpl-2.3"
make -f Make.top leaf le=testing/pmatgen arch=test
make[2]: 进入目录"/home/non/hpl-2.3"
( cd testing/pmatgen ; mkdir -p test )
( cd testing/pmatgen/test ; \
ln -fs /home/non/hpl-2.3/Make.test Make.inc )
make[2]: 离开目录"/home/non/hpl-2.3"
make -f Make.top leaf le=testing/ptimer arch=test
make[2]: 进入目录"/home/non/hpl-2.3"
( cd testing/ptimer ; mkdir -p test )
( cd testing/ptimer/test ; \
ln -fs /home/non/hpl-2.3/Make.test Make.inc )
make[2]: 离开目录"/home/non/hpl-2.3"
make -f Make.top leaf le=testing/ptest arch=test
make[2]: 进入目录"/home/non/hpl-2.3"
( cd testing/ptest ; mkdir -p test )
( cd testing/ptest/test ; \
ln -fs /home/non/hpl-2.3/Make.test Make.inc )
make[2]: 离开目录"/home/non/hpl-2.3"
make[1]: 离开目录"/home/non/hpl-2.3"
make -f Make.top refresh_src      arch=test
make[1]: 进入目录"/home/non/hpl-2.3"
cp makes/Make.auxil      src/auxil/test/Makefile
cp makes/Make.blas       src/blas/test/Makefile
cp makes/Make.comm       src/comm/test/Makefile
cp makes/Make.grid       src/grid/test/Makefile
cp makes/Make.panel      src/panel/test/Makefile
cp makes/Make.pauxil     src/pauxil/test/Makefile
cp makes/Make.pfact      src/pfact/test/Makefile
cp makes/Make.pgesv      src/pgesv/test/Makefile
make[1]: 离开目录"/home/non/hpl-2.3"
make -f Make.top refresh_tst      arch=test
make[1]: 进入目录"/home/non/hpl-2.3"

```

```
cp makes/Make.matgen    testing/matgen/test/Makefile
cp makes/Make.timer     testing/timer/test/Makefile
cp makes/Make.pmatgen   testing/pmatgen/test/Makefile
cp makes/Make.ptimer    testing/ptimer/test/Makefile
cp makes/Make.ptest     testing/ptest/test/Makefile
make[1]: 离开目录“/home/non/hpl-2.3”
make -f Make.top refresh_src    arch=test
make[1]: 进入目录“/home/non/hpl-2.3”
cp makes/Make.auxil      src/auxil/test/Makefile
cp makes/Make.blas       src/blas/test/Makefile
cp makes/Make.comm       src/comm/test/Makefile
cp makes/Make.grid       src/grid/test/Makefile
cp makes/Make.panel      src/panel/test/Makefile
cp makes/Make.pauxil     src/pauxil/test/Makefile
cp makes/Make.pfact      src/pfact/test/Makefile
cp makes/Make.pgesv      src/pgesv/test/Makefile
make[1]: 离开目录“/home/non/hpl-2.3”
make -f Make.top refresh_tst    arch=test
make[1]: 进入目录“/home/non/hpl-2.3”
cp makes/Make.matgen    testing/matgen/test/Makefile
cp makes/Make.timer     testing/timer/test/Makefile
cp makes/Make.pmatgen   testing/pmatgen/test/Makefile
cp makes/Make.ptimer    testing/ptimer/test/Makefile
cp makes/Make.ptest     testing/ptest/test/Makefile
make[1]: 离开目录“/home/non/hpl-2.3”
make -f Make.top build_src      arch=test
make[1]: 进入目录“/home/non/hpl-2.3”
( cd src/auxil/test;          make )
make[2]: 进入目录“/home/non/hpl-2.3/src/auxil/test”
make[2]: 对“all”无需做任何事。
make[2]: 离开目录“/home/non/hpl-2.3/src/auxil/test”
( cd src/blas/test;          make )
make[2]: 进入目录“/home/non/hpl-2.3/src/blas/test”
make[2]: 对“all”无需做任何事。
make[2]: 离开目录“/home/non/hpl-2.3/src/blas/test”
( cd src/comm/test;          make )
make[2]: 进入目录“/home/non/hpl-2.3/src/comm/test”
make[2]: 对“all”无需做任何事。
make[2]: 离开目录“/home/non/hpl-2.3/src/comm/test”
( cd src/grid/test;          make )
make[2]: 进入目录“/home/non/hpl-2.3/src/grid/test”
make[2]: 对“all”无需做任何事。
make[2]: 离开目录“/home/non/hpl-2.3/src/grid/test”
( cd src/panel/test;         make )
make[2]: 进入目录“/home/non/hpl-2.3/src/panel/test”
make[2]: 对“all”无需做任何事。
make[2]: 离开目录“/home/non/hpl-2.3/src/panel/test”
( cd src/pauxil/test;        make )
make[2]: 进入目录“/home/non/hpl-2.3/src/pauxil/test”
make[2]: 对“all”无需做任何事。
make[2]: 离开目录“/home/non/hpl-2.3/src/pauxil/test”
( cd src/pfact/test;         make )
make[2]: 进入目录“/home/non/hpl-2.3/src/pfact/test”
make[2]: 对“all”无需做任何事。
make[2]: 离开目录“/home/non/hpl-2.3/src/pfact/test”
```

```

( cd src/pgesv/test;          make )
make[2]: 进入目录“/home/non/hpl-2.3/src/pgesv/test”
make[2]: 对“all”无需做任何事。
make[2]: 离开目录“/home/non/hpl-2.3/src/pgesv/test”
make[1]: 离开目录“/home/non/hpl-2.3”
make -f Make.top build_tst      arch=test
make[1]: 进入目录“/home/non/hpl-2.3”
( cd testing/matgen/test;     make )
make[2]: 进入目录“/home/non/hpl-2.3/testing/matgen/test”
make[2]: 对“all”无需做任何事。
make[2]: 离开目录“/home/non/hpl-2.3/testing/matgen/test”
( cd testing/timer/test;      make )
make[2]: 进入目录“/home/non/hpl-2.3/testing/timer/test”
make[2]: 对“all”无需做任何事。
make[2]: 离开目录“/home/non/hpl-2.3/testing/timer/test”
( cd testing/pmatgen/test;    make )
make[2]: 进入目录“/home/non/hpl-2.3/testing/pmatgen/test”
make[2]: 对“all”无需做任何事。
make[2]: 离开目录“/home/non/hpl-2.3/testing/pmatgen/test”
( cd testing/ptimer/test;     make )
make[2]: 进入目录“/home/non/hpl-2.3/testing/ptimer/test”
make[2]: 对“all”无需做任何事。
make[2]: 离开目录“/home/non/hpl-2.3/testing/ptimer/test”
( cd testing/ptest/test;      make )
make[2]: 进入目录“/home/non/hpl-2.3/testing/ptest/test”
mpiicc -Dadd__ -DF77_INTEGER=int -DStringSunStyle -DHPL_DETAILED_TIMING -
DHPL_PROGRESS_REPORT -I/home/non/hpl-2.3/include -I/home/non/hpl-2.3/include/test
-I/opt/intel/oneapi/mkl/2022.1.0/mkl/include -O3 -w -ansi-alias -i-static -z
noexecstack -z relro -z now -nocompchk -Wall -qopenmp -mt_mpi -o /home/non/hpl-
2.3/bin/test/xhpl HPL_pddriver.o          HPL_pdinfo.o          HPL_pdtest.o
/home/non/hpl-2.3/lib/test/libhpl.a -
L/opt/intel/oneapi/mkl/2022.1.0/mkl/lib/intel64 -Wl,--start-group
/opt/intel/oneapi/mkl/2022.1.0/lib/intel64/libmkl_intel_lp64.a
/opt/intel/oneapi/mkl/2022.1.0/lib/intel64/libmkl_intel_thread.a
/opt/intel/oneapi/mkl/2022.1.0/lib/intel64/libmkl_core.a -Wl,--end-group -
lpthread -ldl
make /home/non/hpl-2.3/bin/test/HPL.dat
make[3]: 进入目录“/home/non/hpl-2.3/testing/ptest/test”
( cp ../HPL.dat /home/non/hpl-2.3/bin/test )
make[3]: 离开目录“/home/non/hpl-2.3/testing/ptest/test”
touch dexe.grd
make[2]: 离开目录“/home/non/hpl-2.3/testing/ptest/test”
make[1]: 离开目录“/home/non/hpl-2.3”

```

运行 mpirun ./xhpl

```

non@non-virtual-machine:~/hpl-2.3/bin/test$ mpirun ./xhpl
HPL ERROR from process # 0, on line 419 of function HPL_pdinfo:

>>> at least 4 processes for these tests <<<

HPL ERROR from process # 0, on line 621 of function HPL_pdinfo:

>>> Illegal input in file HPL.dat. Exiting ... <<<

```


2.HPCG

同样需要修改Make.Linux_MPI里面的参数，具体就不细说了，接着就是设置安装环境，开始安装的操作。

参考: <https://www.cnblogs.com/lijiayi/p/14283958.html>

！！ 同样要记得设置环境变量

```
non@non-virtual-machine:~/hpcg/setup/build$ /home/non/hpcg/configure Linux_MPI
non@non-virtual-machine:~/hpcg/setup/build$ make
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
-I/home/non/hpcg/src -I/home/non/hpcg/src/Linux_MPI -
-I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/main.cpp -o src/main.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
-I/home/non/hpcg/src -I/home/non/hpcg/src/Linux_MPI -
-I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/CG.cpp -o src/CG.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
-I/home/non/hpcg/src -I/home/non/hpcg/src/Linux_MPI -
-I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/CG_ref.cpp -o src/CG_ref.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
-I/home/non/hpcg/src -I/home/non/hpcg/src/Linux_MPI -
-I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/TestCG.cpp -o src/TestCG.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
-I/home/non/hpcg/src -I/home/non/hpcg/src/Linux_MPI -
-I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/ComputeResidual.cpp -o src/ComputeResidual.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
-I/home/non/hpcg/src -I/home/non/hpcg/src/Linux_MPI -
-I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/ExchangeHalo.cpp -o src/ExchangeHalo.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
-I/home/non/hpcg/src -I/home/non/hpcg/src/Linux_MPI -
-I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/GenerateGeometry.cpp -o src/GenerateGeometry.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
-I/home/non/hpcg/src -I/home/non/hpcg/src/Linux_MPI -
-I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/GenerateProblem.cpp -o src/GenerateProblem.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
-I/home/non/hpcg/src -I/home/non/hpcg/src/Linux_MPI -
-I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/GenerateProblem_ref.cpp -o src/GenerateProblem_ref.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
-I/home/non/hpcg/src -I/home/non/hpcg/src/Linux_MPI -
-I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/CheckProblem.cpp -o src/CheckProblem.o
```

```
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/MixedBaseCounter.cpp -o src/MixedBaseCounter.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/OptimizeProblem.cpp -o src/OptimizeProblem.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/ReadHpcgDat.cpp -o src/ReadHpcgDat.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/ReportResults.cpp -o src/ReportResults.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/SetupHalo.cpp -o src/SetupHalo.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/SetupHalo_ref.cpp -o src/SetupHalo_ref.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/TestSymmetry.cpp -o src/TestSymmetry.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/TestNorms.cpp -o src/TestNorms.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/WriteProblem.cpp -o src/WriteProblem.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/YAML_Doc.cpp -o src/YAML_Doc.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/YAML_Element.cpp -o src/YAML_Element.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/ComputeDotProduct.cpp -o src/ComputeDotProduct.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/ComputeDotProduct_ref.cpp -o src/ComputeDotProduct_ref.o
```

```
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/mytimer.cpp -o src/mytimer.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/ComputeOptimalShapexYZ.cpp -o src/ComputeOptimalShapexYZ.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/ComputeSPMV.cpp -o src/ComputeSPMV.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/ComputeSPMV_ref.cpp -o src/ComputeSPMV_ref.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/ComputeSYMGS.cpp -o src/ComputeSYMGS.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/ComputeSYMGS_ref.cpp -o src/ComputeSYMGS_ref.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/ComputeWAXPBY.cpp -o src/ComputeWAXPBY.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/ComputeWAXPBY_ref.cpp -o src/ComputeWAXPBY_ref.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/ComputeMG_ref.cpp -o src/ComputeMG_ref.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/ComputeMG.cpp -o src/ComputeMG.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/ComputeProlongation_ref.cpp -o src/ComputeProlongation_ref.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/ComputeRestriction_ref.cpp -o src/ComputeRestriction_ref.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/CheckAspectRatio.cpp -o src/CheckAspectRatio.o
```

```

/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/OutputFile.cpp -o src/OutputFile.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/GenerateCoarseProblem.cpp -o src/GenerateCoarseProblem.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/init.cpp -o src/init.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -c -DHPCG_NO_OPENMP -
I/Home/non/hpcg/src -I/Home/non/hpcg/src/Linux_MPI -
I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -qopenmp -mavx -I/home/non/hpcg/src
/home/non/hpcg/src/finalize.cpp -o src/finalize.o
/opt/intel/oneapi/mpi/2021.6.0/bin/mpiicpc -DHPCG_NO_OPENMP -I/Home/non/hpcg/src
-I/Home/non/hpcg/src/Linux_MPI -I/opt/intel/oneapi/mpi/2021.6.0/include -O3 -
qopenmp -mavx src/main.o src/CG.o src/CG_ref.o src/TestCG.o src/ComputeResidual.o
src/ExchangeHalo.o src/GenerateGeometry.o src/GenerateProblem.o
src/GenerateProblem_ref.o src/CheckProblem.o src/MixedBaseCounter.o
src/OptimizeProblem.o src/ReadHpcgDat.o src/ReportResults.o src/SetupHalo.o
src/SetupHalo_ref.o src/TestSymmetry.o src/TestNorms.o src/WriteProblem.o
src/YAML_Doc.o src/YAML_Element.o src/ComputeDotProduct.o
src/ComputeDotProduct_ref.o src/mytimer.o src/ComputeOptimalShapeXYZ.o
src/ComputeSPMV.o src/ComputeSPMV_ref.o src/ComputeSYMGS.o src/ComputeSYMGS_ref.o
src/ComputeWAXPY.o src/ComputeWAXPY_ref.o src/ComputeMG_ref.o src/ComputeMG.o
src/ComputeProlongation_ref.o src/ComputeRestriction_ref.o src/CheckAspectRatio.o
src/OutputFile.o src/GenerateCoarseProblem.o src/init.o src/finalize.o -o
bin/xhpcg

```

运行测试

```

non@non-virtual-machine:~/hpcg/setup/build/bin$ ls
hpcg20221010T141658.txt hpcg20221010T142012.txt HPCG-Benchmark_3.1_2022-10-
10_14-19-01.txt HPCG-Benchmark_3.1_2022-10-10_14-21-55.txt hpcg.dat xhpcg

```

可以发现在build目录里有一个.dat文件以及xhpcg可执行文件

接着运行即可 `./xhpcg`

```

non@non-virtual-machine:~/hpcg/setup/build/bin$ ls
hpcg20221010T141658.txt hpcg20221010T142012.txt HPCG-Benchmark_3.1_2022-10-10-
14-19-01.txt HPCG-Benchmark_3.1_2022-10-10_14-21-55.txt hpcg.dat xhpcg
non@non-virtual-machine:~/hpcg/setup/build/bin$ ./xhpcg
non@non-virtual-machine:~/hpcg/setup/build/bin$

```

总结一下

感觉第一次任务做得有点赶，主要是在配置部分花了太长时间，特别是HPL和HPCG安装，还可能有一部分原因是想学好一些linux、shell的基础和vim的用法，打好基础，笔记也记得算是详细，markdown也是比较熟悉的了。之前有学过git的用法，但是这次时间太赶了来不及push到GitHub上，希望下次能效率更高一点，更早一点开始做，但是有一点感触还是比较深的，就是要学会自己找资料，遇到问题要加强自己参考资料并解决的能力，希望这块能力能进一步提升吧。

下次我一定会记得设置环境变量