Shell

可视化界面(这里说是Visual Interface)实际上一般说GUI(Graphical User Interface)
Shell: 命令行界面(CLI,Command Line Interface)
　Windows： Powershell，Windows7以前没有pwsh。
　Linux，OSX：一般使用Bash(Bourne Again Shell)。
Terminal外观：单行
例如

```
[jon@xpanse missing-semseter]$
```

'$'左边这一串叫做**命令行提示符(Shell Prompt)**，通常包含用户名(username)，机器名称(name of machine)，路径(当前的PATH)。
'$'右边闪烁的光标是在请求你的输入。
命令通常是带着参数(argument)的执行程序。带着参数去执行，可以修改程序的行为。
参数：紧随程序名称后面，用空格分隔的东西。

请注意：后面的示例将不显示*Shell Prompt*

Shell怎么知道这些程序要做什么?

计算机操作系统通常拥有自己的**内置程序(Built-in)**，通常内嵌了终端程序，或者是Windows Explorer，浏览器这类内嵌了围绕终端工作的程序。
这些程序位于File System(文件系统)，让Shell有办法知道程序放在目录的什么地方。

date - 显示日期

举例

```
$date
Wed Aug 11 10:28:58 CST 2021
```

echo - 返回空格后面带的参数(arugument)

```
$echo hello
hello
```

或者

```
$echo "Hello World"
Hello World
```

单转义字符(Escape character)

```
$echo Hello\ World
Hello World
```

Environment Variable

环境变量，类似于程序设计语言的变量(variable)，也就是说，Shell或者说Bash本身就是一种程序设计语言。
它们的Prompt能带参数运行程序，还能写出while循环，for循环，条件循环语句或者是定义函数(详细的将在Shell Scripting说明)。

什么是环境变量?

环境变量是Shell本来就设定好的东西，无论何时去启动Shell都不需要你去重新设置。这类设置好的东西，例如 home目录，username用户名(路径PATH是做这类工作的)。
目录：这里说的是Shell寻找程序时所查找的目录，计算机会寻找与你输入指令名字相同的一个程序或者文件。例如你输入`date`或者`echo`，计算机遍历目录，直到找到它们。
如果你想知道程序是在哪个目录运行的，可以这样做

```
$which
# 如果是echo，就会返回echo所在的目录
/usr/bin/echo
```

说到目录，这里提及一下PATH。
PATH：描述你的计算机文件位置的东西。例如echo的示例，你会发现在/usr前面有一个根目录(root)，这是因为PATH的起点是根目录，它是整个文件系统的最顶层。
这里需要说明：
　Linux，OSX的目录都是绝对路径，也就是说，所有的东西都在一个命名空间(namespace)里。
　Windows比较特殊，它的每个分区都有一个根目录，比如`C:\`，每个驱动器下，都是独立的。这是Windows每个盘符都有一套独立的文件系统的层次结构。

print working directory(pwd)

打印当前工作目录(print working directory)

```
$pwd
/home/Shaymin
```

change directory(cd)

改变工作目录(change directory)

```
# 改变前目录/home/Shaymin
$cd /home
$pwd
/home
```

'.' & '..'

特殊字符单引号(.)和双引号(..)。

```
# '.'表示当前目录，'..'表示上一层目录。
# 当前目录/home
$cd ../
$pwd
/
# 返回根目录了。这里表示的都是相对路径。
$cd -
/home
# 如果输入上面指令，将会返回改变前目录。
$cd ~
/home
# 如果这样做，一定会返回/home目录。
```

这里需要说明，一般用户给出程序名称，Shell会用PATH去查找位置，默认在当前目录。

list(ls)

显示当前目录的文件

```
$ls
picture video download ...
# 输入文件或者文件夹，名称之间会用空格分隔。

# 如果想快速显示上一层目录的文件，键入以下任一指令
$cd ../
$ls
# or
$ls ../

# 如果这样做会显示当前目录的权限。
$ls -l
# 意思是 use a long listing format.
# 示例输出
drwxrwxrwx 1 shaymin shaymin 4096 Aug 11 08:35 missing-semester-notes
```

说明

d - directory
rwxrwxrwx(权限)：由后面的9个字母分成三组组成，它们分别代表了三个不同的用户组。
  1.计算机所有者
  2.拥有文件的用户组
  3.非所有者的其他人
r - read
w - write
x - excute
注意：只有文件w权限，没有它的整个路径目录的w权限，是不能够删除的。除了目录。

rename file(mv)

通过改变文件的所在目录和名称，可以进行重命名或者移动文件位置的操作。

```
$ls
test.txt
$mv /home/Shaymin/test.txt /home/Shaymin/hello.txt
# 在目录home/Shaymin/下的test.txt文本文件被重命名为hello.txt
$ls
hello.txt
```

copy file(cp)

复制文件，格式 [复制源路径] [目标路径]

```
$cp /home/Shaymin/hello.txt /home/hello.txt
# 此时文件复制了一份到/home 目录下。
```

remove file(rm)

移除文件(对目录无效)除非添加参数。

```
[shaymin@ubuntu2004 /home/Shaymin]$rm hello.txt
# 不会返回信息，但文件hello.txt已被删除。
```

make directory(mkdir)

创建一个文件夹目录

```
$ls
picture video download
$mkdir "my photos"
# 特别注意要用双引号引住，否则Shell会认为你要创建两个文件夹目录。
$ls
picture video doawnload my photos
```

manual pages(man)

man是一个程序的说明书，[pages]处输入你要查询说明的程序名称。

```
$man ls
# ls --help也有同样效果
LS(1)                                                   User Commands
                                LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about the FILEs (the current directory by default).
Sort entries alphabetically if none of
       -cftuvSUX nor --sort is specified.

       Mandatory arguments to long options are mandatory for short options too.
```

```
       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..

       --author
              with -l, print the author of each file

       -b, --escape
              print C-style escapes for nongraphic characters

       --block-size=SIZE
              with -l, scale sizes by SIZE when printing them; e.g., '--block-
size=M'; see SIZE format below
# 输入 :q 退出查看。
```

顺带一提，输入`clear`可以清楚当前终端，使用 control+L 也可以。

stream

流。
Stream分为 input stream 和 output stream
  input stream(keyboard)
  output stream(terminal)

用于重定向流的字符

'<' 重定向输入流
'>' 重定向输出流

```
$echo hello > hello.txt
# 此时 hello 的返回结果 将会输入到hello.txt这个文本文件内。
# 这里要在相对路径下生效。
```

如果这时候想要查看hello.txt的内容是否有了hello，需要使用`cat`指令。

cat

程序验证(cat)，用于打印文件内容。

```
$cat hello.txt
hello
# 说明重定向输出流成功。
```

cat也支持重定向流。

```
$cat < hello.txt
hello
# 重定向输入流成功。
```

同时，cat还具有copy功能。

```
$cat < hello.txt > hello2.txt
# 此时同样的内容会出现在hello2.txt中。
```

特殊字符'>>'，追加(append)，向文件末尾继续添加内容(覆写，overwrite)。

```
$cat < hello.txt > hello2.txt
$cat < hello.txt >> hello2.txt
$cat hello2.txt
hello
hello
# 第1行，第2行都是hello。后者是追加内容。
```

pipe('|')

管道操作符'|'，可以操作(io)流的过程。还可以处理二进制图片或者推流视频文件。

```
# 介绍指令 tail -n1 打印文件或者目录的最后(n = 1)行。 可以重定向。
$ls -l | tail -n1
drwxrwxrwx 1 shaymin shaymin 4096 Aug 11 08:35 missing-semester-notes
```

tee

读取输入，写入到文件并且写入到标准输出流。

```
[shaymin@ubuntu2004 /sys]$echo 1060 | sudo tee brightness
1060
# 这里调用权限修改了system目录下的屏幕亮度为1060(尼特)
# 这个操作在WSL(Windows Subsystems for Linux,Windows下的Linux子系统)无法完成。
```

注意：如果不了解，请不要随意修改根目录的文件内容。