Linux入门实操

> 基本上是翻译了下**missing-semester**的1、3讲
>
> 写完这个发现missing-semester有**中文版**orz

Shell基础

shell使用

打开shell之后首先看到这样的提示符

```
zxdclyz@Lyz:~$
```

用户名@设备名:当前路径，`~`代表home家路径

此时就可以输入命令了

```
zxdclyz@Lyz:~$ date
Sat Jun 20 20:07:17 CST 2020
zxdclyz@Lyz:~$
```

通过输入date运行了date这个打印日期和时间的程序，然后提示符重新出现，告诉我们可以输入下一条命令

另一个十分基础的命令是echo

```
zxdclyz@Lyz:~$ echo hello world
hello world
```

echo的作用是直接打印它的参数，在shell中，命令是以空格隔断的，第一个词指定运行的程序，后面的是参数，如果想要在一个参数中包括空格，可以使用`'`或者`"`，也可以用`\(space)`

```
zxdclyz@Lyz:~$ echo 'hello world'
hello world
zxdclyz@Lyz:~$ echo hello\ world
hello world
```

不过在echo中看不出区别orz

那么我们输入date、echo运行的是什么程序呢？shell也是一个编程环境，类似python，其中也有变量，如果输入了程序关键字之外的字段，shell会去查找`$PATH`环境变量中的路径

```
zxdclyz@Lyz:~$ echo $PATH
/usr/local/sbin:/usr/local/bin
```

其中的路径用 `:` 分隔，可以用 `which` 命令查看运行的程序所在路径

```
zxdclyz@Lyz:~$ which echo
/bin/echo
```

如果程序所在路径不在环境变量里，也可以直接使用其完整路径来调用

```
zxdclyz@Lyz:~$ /bin/echo eesast
eesast
```

文件系统

有关文件操作， `pwd` 显示当前路径， `cd` 改变路径， `ls` 打印当前目录中的文件， `.` 指当前目录， `..` 指上级目录

```
zxdclyz@Lyz:~/tutorial$ ls
a.txt  b.py  c.js
zxdclyz@Lyz:~/tutorial$ cd ../
zxdclyz@Lyz:~$ pwd
/home/zxdclyz
zxdclyz@Lyz:~$
```

使用 `cd -` 可以跳转到刚才所在的路径，用 `mkdir` 创建文件夹， `mv` 移动文件， `cp` 复制文件， `rm` 删除文件， `touch` 创建文件

有关文件权限，参考课件

查看命令参数

可以使用 `man command` 来进入手册页，也可以直接使用--help参数查看

```
zxdclyz@Lyz:/bin$ ls --help
...
zxdclyz@Lyz:/bin$ man ls
```

连接程序

在shell中，程序也有输入流和输出流，我们可以用 `> file` 和 `< file` 来重新引导流

```
zxdclyz@Lyz:~/tutorial$ echo hello > hello.txt
zxdclyz@Lyz:~/tutorial$ echo bye >> hello.txt
```

使用 `>>` 可以进行追加写

另一个十分有用的工具是pipe `|`，它可以连接两个命令，将前一个命令的输出当做后一个命令的输入

```
zxdclyz@Lyz:~/tutorial$ cat hello.txt
hello
bye
hello world
zxdclyz@Lyz:~/tutorial$ cat hello.txt|grep world
hello world
```

Vim极简入门

如果你在使用服务器并且想简单地对一个文件进行编辑，使用vim是十分方便的（大量编辑我倾向于使用VSCode等远程连接），vim本身拥有十分强大的功能，但我自己也并不会，所以只在此展示最基础的用法

使用 `vim file` 打开文件进行编辑

刚进入时处于正常模式，而vim中有这些模式

- **Normal**: for moving around a file and making edits
- **Insert**: for inserting text
- **Replace**: for replacing text
- **Visual** (plain, line, or block): for selecting blocks of text
- **Command-line**: for running a command

我们在此用到insert模式，在正常模式下按 `i` 进入insert模式，此时可以像其他编辑器一样对文件进行编辑，然后按 `ESC` 退回到正常模式

在正常模式下输入 `:` 会进入命令模式，最基础的命令有

- `:q` quit (close window)
- `:w` save (“write”)
- `:wq` save and quit

事实上vim还有许多十分强大的功能，感兴趣的同学可以自己了解

tmux

我们在使用ssh方法登录到服务器的时候是和服务器进行一次会话，而当我们从服务器断开的时候，会话结束，会话中的各个进程也会被终止，而tmux是用来解绑会话与打开的终端窗口的，可以让你启动的进程在断开连接之后继续运行

直接输入tmux命令会新建一个只有编号的tmux窗口，我们可以这样来新建命名的窗口

```
$ tmux new -s <session-name>
```

在tmux窗口中，按下 `Ctrl+b` ，然后按 `d` ，可以从当前窗口分离，而窗口中的进程仍在后台运行

可以查看所有tmux会话

```
$ tmux ls
# or
$ tmux list-session
```

然后使用 `tmux attach` 来重新接入会话

```
# 使用会话编号
$ tmux attach -t 0

# 使用会话名称
$ tmux attach -t <session-name>
```

可以在会话中直接使用 `Ctrl+d` 或者输入 `exit` 来关闭，也可以在外部使用 `tmux kill-session` 命令杀死某个会话。

```
# 使用会话编号
$ tmux kill-session -t 0

# 使用会话名称
$ tmux kill-session -t <session-name>
```

以上为tmux的最基础操作，更多可以参考 **这里**