missing semester

看完课程后补上的笔记。首页链接：计算机教育中缺失的一课 . lecture0.pdf。
视频链接：[自制双语字幕] 计算机教育缺失的一课(2020) - 第1讲 - 课程概览与 shell_ 哔哩哔哩 _bilibili。个人认为胜过youtube版本。

目录

1 Lecture1 CourseOverview and Shell

课程概览与 shell . lecture 1.pdf
本课程包含 11 个时长在一小时左右的讲座，每一个讲座都会关注一个特定的主题。
教学大部分是面向Linux的。课程会使用bash，最被广泛使用的shell。

当您打开终端时，您会看到一个提示符，它看起来一般是这个样子的：`missing:~$`
这是 shell 最主要的文本接口。它告诉你，你的主机名是 missing 并且您当前的工作目录 (cwd)或者说您当前所在的位置是 ~ (表示home)。 $符号表示您现在的身份不是 root 用户。

shell 基于空格分割命令并进行解析，然后执行第一个单词代表的程序，并将后续的单词作为程序可以访问的参数。如果您希望传递的参数中包含空格（例如一个名为 My Photos 的文件夹），您要么用使用单引号，双引号将其包裹起来，要么使用转义符号 \ 进行处理（ My\ Photos ）。

shell 是一个编程环境，所以它具备变量、条件、循环和函数。
环境变量是在shell启动是设置的东西。其中包含你的主目录路径以及用户名。
当你执行程序时，shell会在环境变量$PATH中寻找，直到找到相应的程序。
可以通过`which [arg]`命令来寻找指定程序的路径。

shell 中的路径是一组被分割的目录，在Linux和macOS上使用/分割，而在Windows上是\。
绝对路径是能完全确定文件位置的路径。。如果某个路径以/开头，那么它是一个绝对路径。
相对路径是指相对于当 前工作目录的路径。
当前工作目录可以通过`pwd(present working directory)`获取。
`cd -` 会前往上一个目录。

`ls -l` 可以更加详细地打印出目录下文件或文件夹的信息。
首先，行中第一个字符d表示该文件是一个目录。然后接下来的九个字符，每三个字符构成一组。它们分别代表了文件所有者， 用户组以及其他所有人具有的权限。 其中- 表示该用户不具备相应的权限。
注意，/bin目录下的程序在最后一组，即表示所有人的用户组中，均包含 x 权限，也就是说任何人都可以执行这些程序。
如果你对于一个文件有写权限，对于目录却没有写权限，那么你只能清空此文件，不能删除此文件。
简单来说：

表格 1. rwx

r	是否被允许看到这个目录中的文件
w	是否被允许在该目录中重命名，创建或修改文件
x	是否被允许进入该目录或是执行文件

`mv`命令可以用于重命名或是移动文件。
eg：`mv aaa bbb`将文件 aaa 改名为 bbb。 `mv aaa /bbb` 将文件aaa移入bbb目录。

`cp [origin] [target]`命令可以用于复制文件。 其需要两个参数， 一个是要复制的文件路径，一个是目标路径。

在 shell 中，程序有两个主要的“流”：它们的输入流和输出流。当程序尝试读取信息时，它们会从输入流中进行读取，当程序打印信息时，它们会将信息输出到输出流中。通常输入是键盘，输出则是屏幕终端，但是我们可以重定向这些流。

最简单的方法是使用<以及>符号。
例如使用`cat < aaa.txt > bbb.txt`。此命令会将aaa文件作为cat的输入，同时将cat的输出覆写到bbb文件中。
>>的作用则是追加内容，而非覆写。
|管道(pipes)则可以将一个程序的输出与另一个程序的输入连接起来。

`sudo`命令可以以superuser的身份执行操作。

`tee`命令从标准输入中复制到一个文件，并输出到标准输出。
eg: `ping google.com | tee output.txt`
输出内容不仅会被写入文件，也会被显示在终端中。

`touch`命令可以修改文件的参数，或是创建一个不存在的文件。

2 Shell Tools and Scripting

一般来说，shell脚本中的空格是用来作为参数分隔的。
在shell中，$1表示的是脚本的第一个参数。具体如下：

表格 2. bash特殊符

$0	脚本名
$1-$9	脚本第一到第九个参数
$@	所有参数
$#	参数的数量
$?	上一条指令的返回值
$$	现在脚本进程的pid
!!	上一条指令(包括参数)，用来sudo !!执行上一条没执行成功的指令时很有用
$_	上一条指令的最后一个参数

命令会通过STDOUT返回值，错误则通过STDERR。
0表示执行正确，其他值则是错误。

short-circuting[1]逻辑短路指使用计算机中的与或运算来跳过语句。
eg: `ture || echo "will not be printed"` and `false && echo "will not be printed"`

在shell中，我们可以将一个命令的输出作为一个变量处理。例如 `for file in $(ls)`。
我们也可以通过 `<(cmd)`的形式将命令的输出放入一个临时文件中。因为有些命令是从文件中获取输入而非STDIN标准输入，所以有时这很有效。`eg: diff <(ls aaa) <(ls bbb)`。

1. Short-circuit evaluation - Wikipedia。

当在bash中进行逻辑判断时，我们推荐使用[[ ]]而非[ ][2]。尽管其不兼容与sh，但这更安全。。

在bash中我们可以通过通配符来简化操作和参数。
通配符*与？。*匹配多位，？只匹配一位。
`eg: foo1,foo2,foo10. foo?-->foo1,foo2. foo*-->foo1,foo2,foo10`
花括号则可以用来拓展。
`eg: mv *{.py,.sh} will move all *.py and *.sh。 touch {foo,bar}/{a..h} will create files foo/a,foo/b...foo/h,bar/a...bar/h`。

你也可以通过其他语言写脚本。

我们可以通过文件顶部的shebang[3]来指定执行脚本的解释器。
我们可以在shebang中使用env[4]命令来解析系统中相关命令的位置，从而提高可移植性。
`eg: #!/usr/bin/env python`

man手册可以帮助我们了解命令的使用方法，但有时候man手册提供的内容过多了。这时我们可以通过tldr[5]来了解相关命令的例子从而更好的使用它们。

`find`命令可以用来寻找文件，同时也可以对这些文件进行操作。
`eg: find . -name '*.tmp' -exec rm {} \; delete all files with .tmp extension`
我们也可以应用更现代且语法更便捷的fd[6]来搜索文件。
`locate`命令。。感觉没啥用。

搜索文件中的内容可以使用grep命令。你也可以使用更现代的ripgrep。
`eg: rg -u --files-without-match "^#!"`
`it will find all files (including hidden files by using -u) without a shebang(by using --files.. to print those who don't match the pattern we give to command)`

可以通过`history`命令来快速的回归命令历史。也可以通过<c-r>来反向搜索命令历史。
fzf[7]是更高效的反向搜索工具。

快速的显现目录结构可以通过`ls -R` 或是 `tree`来实现。

3 Editors(Vim)

Vim是基于模式的。具体转换如下:

$$\text{normal} \overset{i}{\longleftrightarrow}^{\text{esc}} \text{insert}$$
$$\text{normal} \overset{r}{\longleftrightarrow}^{\text{esc}} \text{replace}$$
$$\text{normal} \overset{v}{\longleftrightarrow}^{\text{esc}} \text{visual}$$
$$\text{normal} \overset{\text{shift}+v}{\longleftrightarrow}^{\text{esc}} \text{visual-line}$$
$$\text{normal} \overset{<c-v>}{\longleftrightarrow}^{\text{esc}} \text{viusal-block}$$

2. BashFAQ/031 - Greg's Wiki (wooledge.org)
3. Shebang (Unix) - Wikipedia
4. env(1) - Linux manual page (man7.org)
5. tldr pages
6. sharkdp/fd: A simple, fast and user-friendly alternative to 'find' (github.com)
7. Configuring shell key bindings · junegunn/fzf Wiki (github.com)

$$\text{normal} \overset{:}{\longleftrightarrow}^{\text{esc}} \text{command}$$

表格 3. vim :

:q	退出当前窗口
:w	保存
:wq	保存并退出
:help {topic}	open help
:e {filename}	切换文件
:ls	显示打开的buffers

Vim会维持一组打开的文件，即*buffer*。
Vim具有很多的*tab*,每一个都对应一个或多个*window*。
每一个*window*显示出一个*buffer*。
一个*buffer*可以被多个*window*显示，即使这些*window*在同一个*tab*中。

表格 4. Vim movement

w	下一个词 next word
b	单词开头(向前) begining of word
e	单词尾部(向后) end of word
0	行开头
^	此行第一个非空字符
$	行尾
H	窗口顶部
M	窗口中部
L	窗口底部
<C-u>	向上滚动(up)
<C-d>	向下滚动(down)
gg	文件开头
G	文件尾部
{number}G	跳到相应行
%	在{，[，(上使用会跳转到对应的括号
f{character}	向后跳转到此字符
t{character}	向后跳转到此字符前
F{character}	向前跳转到此字符
T{character}	向前跳转到此字符后
/(word)	使用/进行单词搜索，按下回车后会跳转到相应处

表格 5. Vim modify

~	翻转大小写
ci[	修改[ ]中的内容
da(	删除包括( ) 在内的所有内容

使用~/.vimrc文件自定义你的Vim。
你可以在自己的shell中启用vim模式。
bash使用set -o vi。zsh使用bindkey -v。fish使用fish_vi_key_bindings。
你也可以不论shell类型直接使用export EDITOR=vim。但这也会改变其他程序的editor，比如git。Readline也可以使用 set editing-mode vi来进入Vim模式。例如python repl便会收到影响。

使用Vimium - Chrome Web Store (google.com)在chorme中启用Vim模式。
宏就不介绍了，一时半会说不清楚。
使用最少的操作完成文件处理:VimGolf - real Vim ninjas count every keystroke!。

4 Data Wrangling

将一种格式的数据变为另一种格式的都可以称为数据整理。
`less`命令可以用来分页查看文本。
使用sed这个流编辑器来处理文本。`eg: cat ssh.log | sed 's/.*Disconnected from //'`
s替换命令的格式是 s/REGEX/SUBSTITUTION