

STAT243 Lecture 1.3 UNIX Command Line

Logic ▾

本教程链接: <https://computing.stat.berkeley.edu/tutorial-unix-basics/>

Logic ▾

本教程涵盖了类 UNIX (例如 Linux 或 MacOS) 环境的基础知识, 并特别介绍了 UNIX command line interface 的使用

这是一种在计算机上执行操作和自动化任务的有效方式, 熟悉 command line operation 将使你能够以可重复的方式更快地完成任务

本教程的教材见 [GitHub](#), 教学视频见 [YouTube video](#) 的 Episodes 1-3 (前 20 分钟)

1 Introduction

1.1 The Shell

1. Shell 是什么

- 在 UNIX 命令行上操作也被称为 "using the terminal" 和 "using the Shell"
- 在使用 UNIX-style operating system (例如 Linux 或 MacOS) 的 terminal window 时, 你与之交互的 UNIX program 即为 Shell
- Shell 位于你与操作系统之间, 是一个为你运行其他命令并显示结果的程序

2. Shell 的类型

- `bash` 非常常见, 并且在许多系统上是默认的
- 在较新的 MacOS 版本中, `zsh` 是默认的 Shell, 它是 `bash` 的扩展
- 其他 Shells: `sh`, `csh`, `tcsh`, `ksh`

1.2 Accessing a UNIX command line interface

访问 UNIX 命令行界面的方法如下:

- MacOS: 可以在 `Applications -> Utilities -> Terminal` 下找到 Terminal
- Windows:
 - If you have a sufficiently new version of Windows 10, you can use the [Windows Subsystem for Linux](#), which will provide you with an Ubuntu shell running bash on your own machine.
 - If you have access to remote machines running Linux, you can login to them using programs such as MobaXTerm and Putty. Once logged in, you'll find yourself in a Terminal window on the remote machine.
- JupyterHub: 可以在 "New" 下启动一个 Terminal session
- Cloud-based options: 可以尝试使用 Google [Cloud Shell](#) 等云服务

1.3 Getting started

进入终端后就会看到 "prompt", 表示 Shell 正在等待我们输入命令

有时 prompt 只有 `$`, 但它通常会包含 current user 和我们在 filesystem 中所在的 directory 的信息

- 如果看到的是 `>` 而不是 prompt, 这意味着 Shell 认为您还没有完成输入命令, 并且正在等待您输入更多信息
- 如果看到的是一个 newline 但没有任何其他内容, Shell 可能期望您输入一些文本以便它进行处理

Example ▾

这是一个显示当前用户是 "scflocal", 在名为 "gandalf" 的机器上, 位于 user home directory (通常用 `~` 表示) 中的 "tutorial-unix-basics" (sub) directory 中的 prompt:



Shell

```
1 scflocal@gandalf :~/tutorial-unix-basics>
```

⚠ Remark: Tutorial code formatting ▾

在教程的其余部分, 将不会显示命令前的 prompt, 输出 (如果有) 将跟随代码

⚠ Remark ▾

如果不确定该做什么, 可以按 `Ctrl-c` 返回到 usual prompt

1.4 查询当前用户名

`whoami` 会打印出当前用户的 username:



Shell

```
1 whoami
```

2 Files and directories

2.1 查询目录

2.1.1 查询当前工作目录

任何时候, 你在 UNIX 命令行中都有一个 **working directory**, 它是你当前在 file system 中的位置

`pwd` ("print working directory") 命令可以查询当前的工作目录:



Shell

```
1 pwd
```

```
1 /home/scflocal/tutorial-unix-basics
```

2.1.2 查询工作目录下的 files 和 subdirectories

`ls` 会列出工作目录中的文件 (和子目录)



Shell

```
1 ls
```

```
1 assets
2 _config.yml
3 example.text
4 example.txt
5 filename with spaces.txt
6 _freeze
```

```
7  _includes
8  index.qmd
9  index.rmarkdown
10 _layouts
11 mv_assets.sh
12 myfile
13 name of my file with spaces.txt
14 _quarto.yml
15 README.md
16 _sass
17 _site
```

2.2 访问目录

`cd` 可以使用 absolute path (绝对路径) 与 relative path (相对路径) 访问目录

2.2.1 访问 home directory

单独运行 `cd` 可以访问 home directory

```
>- Shell
1  cd
2  pwd

1  /home/scflocal
```

2.2.2 访问子目录

可以使用 `cd` 并加上子目录的名称, 这个子目录是相对于我们的工作目录找到的

```
>- Shell
1  cd tutorial-unix-basics
2  pwd

1  /home/scflocal/tutorial-unix-basics
```

2.2.3 访问嵌套子目录

可以使用 `cd` 并加上 nested subdirectories 的名称

```
>- Shell
1  cd
2  cd tutorial-unix-basics/assets
3  pwd

1  /home/scflocal/tutorial-unix-basics/assets
```

2.2.4 访问父目录

可以使用 `..` 访问任何目录的父目录

```
>- Shell
1  pwd
2  cd ..
3  pwd
```

```
1 /home/scflocal/tutorial-unix-basics/assets /home/scflocal/tutorial-unix-basics
```

可以使用相对路径让 `..` 的使用更复杂, 这里我们将先向上移动一个目录, 然后进入一个不同的子目录:

```
>- Shell
1 cd assets
2 cd ../_includes
3 pwd
```

```
1 /home/scflocal/tutorial-unix-basics/_includes
```

然后我们向上移动两级目录, 再进入另一个子目录

```
>- Shell
1 cd ../../Desktop # go up two directories and down
2 pwd
```

```
1 /home/scflocal/Desktop
```

⚠ Remark ▾

以上所有示例都使用了相对路径, 根据你运行命令时的 `current working directory` 进行导航

2.2.5 使用绝对路径进行访问

使用绝对路径的方法是让路径以 `/` 开头, 如 `/home/scflocal`, 如果路径不以 `/` 开头, 则会被解释为相对于 `current working directory` 的相对路径

```
>- Shell
1 cd /home/scflocal/tutorial-unix-basics/assets
2 pwd
```

```
1 /home/scflocal/tutorial-unix-basics/assets
```

⚠ Remark: Absolute paths are not robust ▾

在 `script` 中使用绝对路径通常是一个坏主意, 因为如果 `script` 在不同的机器上运行 (这些机器通常有不同的文件系统结构) 或以不同的用户身份运行 (这些用户通常有不同的主目录), `script` 可能无法正常工作

2.3 The filesystem

2.3.1 查询文件系统结构

文件系统本质上是一个 `upside-down tree`

可以使用 `tree` 查看树状的文件系统结构:

```
>- Shell
1 tree
```

```
1 .
```

```
2 |— assets
3 |   |— css
4 |     |— style.scss
5 |   |— fonts
6 |     |— Noto-Sans-700
7 |       |— Noto-Sans-700.eot
8 |       |— Noto-Sans-700.svg
9 |       |— Noto-Sans-700.ttf
10 |      |— Noto-Sans-700.woff
11 |      |— Noto-Sans-700.woff2
12 |     |— Noto-Sans-700italic
13 |       |— Noto-Sans-700italic.eot
14 |       |— Noto-Sans-700italic.svg
15 |       |— Noto-Sans-700italic.ttf
16 |       |— Noto-Sans-700italic.woff
17 |       |— Noto-Sans-700italic.woff2
18 |     |— Noto-Sans-italic
19 |       |— Noto-Sans-italic.eot
20 |       |— Noto-Sans-italic.svg
21 |       |— Noto-Sans-italic.ttf
22 |       |— Noto-Sans-italic.woff
23 |       |— Noto-Sans-italic.woff2
24 |     |— Noto-Sans-regular
25 |       |— Noto-Sans-regular.eot
26 |       |— Noto-Sans-regular.svg
27 |       |— Noto-Sans-regular.ttf
28 |       |— Noto-Sans-regular.woff
29 |       |— Noto-Sans-regular.woff2
30 |   |— img
31 |     |— logo.svg
32 |     |— ls_format.png
33 |   |— js
34 |     |— scale.fix.js
35 |   |— stat_bear.png
36 |   |— styles.css
37 |— _config.yml
38 |— example.text
39 |— example.txt
40 |— filename with spaces.txt
41 |— _freeze
42 |   |— index
43 |     |— execute-results
44 |     |— html.json
45 |   |— site_libs
46 |     |— clipboard
47 |     |— clipboard.min.js
48 |— _includes
49 |   |— toc.html
50 |— index.qmd
51 |— index.rmarkdown
52 |— _layouts
53 |   |— default.html
54 |— mv_assets.sh
55 |— myfile
56 |— name of my file with spaces.txt
57 |— _quarto.yml
58 |— README.md
59 |— _sass
60 |   |— fonts.scss
61 |   |— jekyll-theme-minimal.scss
62 |   |— jekyll-theme-minimal.scss.bak
```

```
63 |   └─ minimal.scss
64 |   └─ rouge-github.scss
65 |   └─ _site
66
67 18 directories, 46 files
```

- `.` 表示 current working directory (`tutorial-unix-basics`)
- 其中包含子目录 `asset` 、 `_includes` 、 `_layouts` 等
- 这些目录中又包含文件和进一步的子目录 (如 `assets` , 它包含名为 `css` 和 `fonts` 的子目录)

2.3.2 查询根目录 root directory

可以使用 `/` 表示整个文件系统的顶端, 也就是根目录

`/` 目录中包含许多子目录, 例如

- `/home` (包含了所有用户的 home directories (家目录), 属于某个用户的所有文件都存储在这里)
- `/bin` (包含 UNIX 程序, 也称为 "二进制文件")
- `/tmp` 是一个存放只需要短暂使用且不需要保存的临时文件的地方, 当机器重启时, 这些文件会消失



Shell

```
1  ls /
```

```
1  accounts
2  app
3  bin
4  boot
5  dev
6  etc
7  home
8  lib
9  lib32
10 lib64
11 libx32
12 lost+found
13 media
14 mirror
15 mnt
16 opt
17 pool0
18 proc
19 root
20 run
21 sbin
22 scratch
23 server
24 srv
25 swap.img
26 sys
27 system
28 tmp
29 usr
30 var
```

2.3.3 查询家目录 home directory

如果有一个名为 `scflocal` 的用户, 则与该用户相关的一切信息都会存储在该用户的 **home directory** 中, 即 `/home/scflocal` (在不同系统中具体位置可能会有所不同)

- 可以使用 `~scflocal` 来表示 `scflocal` 的家目录, 即 `/home/scflocal`
- 如果你是 `scflocal` 用户, 可以通过快捷方式 `~` 来访问家目录

```
>_ Shell
1 ls /home
```

```
1 scflocal
2 shiny
```

```
>_ Shell
1 cd /home/scflocal
2 pwd
```

```
1 /home/scflocal
```

```
>_ Shell
1 cd ~
2 pwd
```

```
1 /home/scflocal
```

```
>_ Shell
1 cd ~scflocal
2 pwd
```

```
1 /home/scflocal
```

2.3.4 访问 tmp 目录

```
>_ Shell
1 cd /tmp
2 ls
```

```
1 assets
2 assets.tgz
3 quarto-session71f9ea197eacbf29
4 RtmpMQcgrV
5 Temp-76c6318d-4f54-44d2-8bd9-d9a42eeeb7ce
6 test
```

2.3.5 返回最近所在的目录

可以使用 `cd -` 返回到最近所在的目录

```
>_ Shell
1 cd -
2 pwd
```

