

Week 1: POSIX Systems

1. System Administration

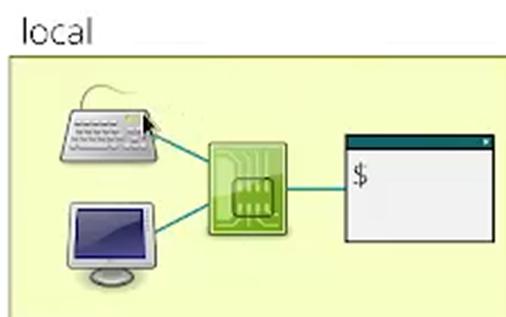
1.1 Secure Shell

安全外壳 (SSH) 是一种允许您远程登录另一台计算机（例如实验室机器）的协议。几乎每个使用 SSH 的人都使用免费的 OpenSSH 实现。

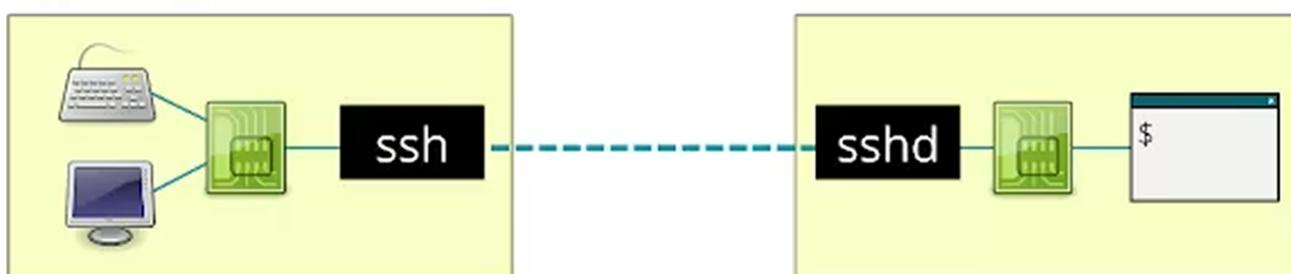
`sshd` 是 Secure Shell (SSH) 服务的守护进程，用于接受远程计算机通过 SSH 协议进行安全连接。SSH 是一种加密的网络协议，它允许通过不安全的网络（如互联网）建立安全的连接，用于远程登录和执行命令、文件传输等操作。

在大多数 Unix-like 操作系统中，包括 Linux 和 BSD 系统，`sshd` 是 SSH 服务的守护进程。它监听在默认的 SSH 端口（通常是 22），等待来自客户端的连接请求。一旦建立连接，`sshd` 负责验证用户身份，处理用户的命令和请求，并确保通信过程中的数据加密。

Shell



remote with ssh



\$ ssh [USERNAME@] HOSTNAME

- `ssh` 是客户端，您在自己的计算机上运行它以连接到另一台计算机。
- `sshd` 是服务器，或者是 UNIX 中的守护进程。它在您要连接的计算机的后台运行，需要由系统管理员安装。

Advanced note

SSH 默认使用 TCP 端口 22。

1. 连接到实验室 `uname -a` 这个命令通常用于显示操作系统的名称 `whoami ssh -J` jump through this host, 跳板连接是一种通过一个中间主机（跳板主机）来连接目标主机的方法，通常用于通过内部网络或防火墙访问无法直接连接的主机。
2. 设置 ssh 密钥 something you know (password or PIN), something you have (physical key, digital key, ID card, passport) something you are (biometrics)
 - 您可以使用用户名和密码登录，即“您知道的内容”。这是默认设置，但不是最安全的。
 - 您可以使用（数字）密钥登录，即“您拥有的东西”。这是更安全的，只要您的密钥（只是一个文件）不会落入坏人之手，而且也是最方便的，因为您可以登录实验室机器或复制文件而无需输入密码。
 - 您可以使用本身受密码保护的密钥文件登录。这将为您提供双因素身份验证。

这一节就是讲keys，keys需要密码，而且本身也是一个文件，所以是双重登陆

SSH 使用的密钥实现数字签名。每个密钥都作为一对文件出现：

- 通常命名为 CIPHER 的文件中的私钥（也称为秘密密钥）`id_CIPHER` 是正在使用的密钥。您需要保证其安全，并且仅将其存储在只有您有权访问的地方。
- 通常名为 的文件中的公钥 `id_CIPHER.pub`。你可以与全世界共享它，并且你需要在任何机器上或任何你想要登录的服务上存储它的副本（对于实验室，因为实验室机器都共享一个文件系统，你只需要存储一次 - 但 seis 有一个单独的文件系统，因此您需要在那里有一个单独的副本）。

使用的密钥是一对，一个private一个public

创建：`ssh keygen`

查看存储密钥的文件夹。`ls -l ~/.ssh` 将会执行此操作，除非您在创建它们时选择其他位置来存储它们：

```
-rw-----. 1 vagrant vagrant  411 Oct  7 10:50 id_ed25519
-rw-r--r--. 1 vagrant vagrant   98 Oct  7 10:50 id_ed25519.pub
-rw-r--r--. 1 vagrant vagrant 1597 Oct  7 11:54 known_hosts
```

`/home/if23696/.ssh/id_ed25519`

所以本地有三个东西：一个秘钥，一个公钥，一个`known_hosts`（OpenSSH 中用于存储远程主机的公钥信息的文件。当你首次连接到一个远程主机时，OpenSSH 会将该主机的公钥信息保存在你的 `~/.ssh/known_hosts` 文件中。这样，下次连接到相同的主机时，OpenSSH 将检查该主机的公钥是否与 `known_hosts` 文件中存储的相匹配。）3. 在 SEIS 上设置密钥访问

`ssh -A -J USERNAME@seis.bris.ac.uk USERNAME@rd-mvb-linuxlab.bristol.ac.uk` 4. 设置配置文件 `scp-secret`
`copies`: 用于在 Unix-like 操作系统之间安全地复制文件的命令。它是基于 SSH 协议的，因此提供了加密的数据传

输入和身份验证。scp 的名称代表 "Secure Copy Protocol"

```
scp ~/.ssh/id_ed25519.pub "USERNAME@seis.bris.ac.uk:~/.ssh/"
```

The general syntax of scp is `scp source destination` and source or destination may be of the form `[USERNAME@]HOSTNAME:PATH` - if it contains a colon (:), then it refers to a file on a different machine.

如果有这个东西，说明是另一个主机的地址 复制以过去之后将内容复制到authorized_keys文件中

```
cd .ssh  
cat id_ed25519.pub >> authorized_keys  
chmod 600 authorized_keys
```

chmod 600->八进制->110 000 000-> rw- --- ---

1.2 Vagrant

虚拟机

vagrant up start machine

vagrant ssh log in

vagrant halt stop machine

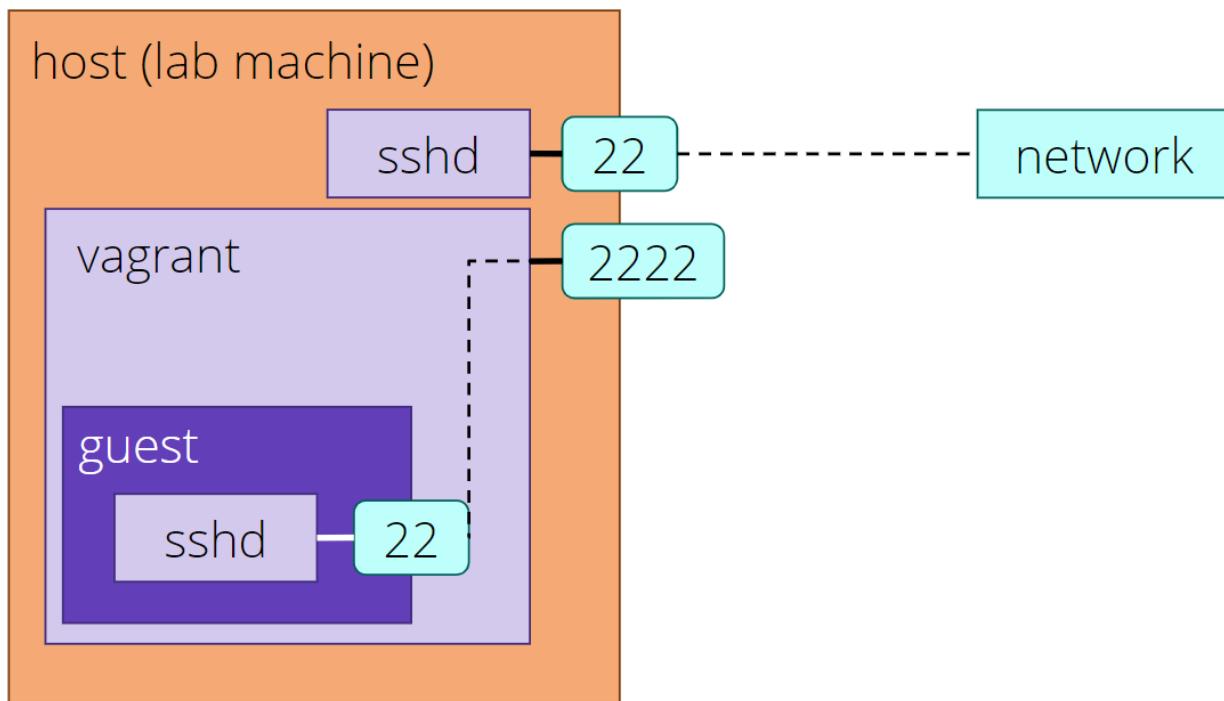
vagrant reload stop+start machine (for config update)

vagrant destroy delete machine

All commands require a Vagrantfile in the current directory.

需要有一个Vagrantfile

ssh and port forwarding



ssh默认使用的是**22**端口，但是ssh默认使用的是**2222**端口 Vagrant Box有默认的private key和secret key，一个新的默认更新一个

1.3 System administration

apt advanced package tool

finding packages

\$ **apt search [-v] [-d] STRING**

\$ **apt info [-a] PACKAGE**

\$ **apt list [-I] PACKAGE**

\$ **apt [COMMAND] --help**

- -V:选项用于显示详细的搜索结果，包括软件包的详细信息
- -d 选项用于显示每个软件包的详细描述
- -a:显示所有版本的信息，感觉是all
- -l:而 -l 是一个选项，用于显示已安装软件包的详细信息，包括版本号、架构等 单独的

sudo apt update 只是下载最新的package的清单，但是不安装任何东西 sudo apt upgrade 将所有已安装的 package 更新到最新版本 sudo apt install PACKAGE PACKAGE 安装

`dpkg-query -S /bin/bash` Debian系统及其衍生系统（

- `dpkg-query` 是一个用于查询 Debian 软件包数据库的工具。
- `-S` 选项用于指定一个文件路径，并查找包含该文件的软件包。
- `/bin/bash` 是要查询的文件路径，这里是 Bash shell 的二进制可执行文件路径。

因此，这个命令的含义是查找 `/bin/bash` 文件所属的软件包。在这个例子中，它会输出与 Bash shell 相关的软件包的信息，包括软件包的名称和版本号。

运行这个命令的输出可能是类似以下的形式：

javascript

 Copy code

```
bash: /bin/bash
```

这表示 `/bin/bash` 文件属于 `bash` 软件包。

- `dpkg-query` 是一个用于查询 Debian 软件包数据库的工具。
- `-S` 选项用于指定一个文件路径，并查找包含该文件的软件包。
- `/bin/bash` 是要查询的文件路径，这里是 Bash shell 的二进制可执行文件路径。

因此，这个命令的含义是查找 `/bin/bash` 文件所属的软件包。在这个例子中，它会输出与 Bash shell 相关的软件包的信息，包括软件包的名称和版本号。

然后给你讲一下linux系统 `ls` /列出根目录的东西 `/bin`binary, 都是各种系统指令,
`/bin` was only for binaries needed to start the system - or at least the most important
binaries that needed to live on the faster of several disk drives, like your shell.

小小的提了一嘴`which`:用于查找并显示指定命令的绝对路径。具体而言，它会告诉你系统中关于某个命令的执行文件所在的位置。一般都是系统层面的指令，比如说`mv`, `ls`这种最基本的 比如说`which ls` 和`pwd`不同的 是，`pwd`主要是当前所在位置，`find`是查找文件，然后`which`是查找指令

- `'/bin'`: 包含基本的系统命令，用于所有用户。
- `'/sbin'`: 包含系统管理员使用的系统命令，这些命令通常需要特权 (root) 用户才能执行。
- `'/usr/bin'`: 包含用户安装的软件包提供的二进制文件。
- `'/usr/sbin'`: 包含用户安装的软件包提供的系统管理员命令。

`/usr`存放用户安装软件以及系统共享资源的重要目录。和用户相关，但是基本都是只读文件 提到了绿色的是可执行文件，蓝色的是指向其他文件的soft link

```
root@Ada:~# ls -l /bin
lrwxrwxrwx 1 root root 7 Jan  3  2023 /bin -> usr/bin
```

软链接 (Symbolic Link) :

- 性质:** 软链接是一个指向另一个文件的特殊文件，可以理解为一个快捷方式或别名。
- 文件内容:** 软链接保存的是目标文件的路径，而非实际的文件内容。
- 大小:** 软链接本身是一个独立的文件，保存了目标文件的路径信息，因此占用磁盘空间较小。
- 更新:** 如果目标文件被删除或移动，软链接仍然存在，但它将失效（称为“悬空指针”）。
- 创建:** 使用 `ln -s` 命令创建软链接。

bash

 Copy code

```
ln -s target_file symlink_name
```

硬链接 (Hard Link) :

- 性质:** 硬链接是文件系统中对同一文件 inode 的多个名称。
- 文件内容:** 硬链接共享相同的 inode 和数据块，它们实际上是相同的文件。
- 大小:** 由于多个文件名指向同一 inode，因此每个硬链接都占用实际文件大小的磁盘空间。
- 更新:** 如果原始文件被删除，硬链接仍然可以访问文件的内容，因为硬链接和原始文件共享相同的 inode。
- 限制:** 硬链接只能在同一文件系统内创建，不能跨文件系统。

二/五.

所以这里提到了软连接和硬连接 **inode**: inode 是文件系统中的一个概念，用于表示文件或目录的元数据信

息。每个文件或目录在文件系统中都与一个唯一的 inode 相关联。inode 记录了文件或目录的诸多属性，但不包括文件名或路径。

/etc 系统层面的 configuration file，然后基本是只有 root 用户可以更改 /lib dynamic-libraries，比如 windows 的 .dll，linux 系统的 .so /home 用户的 home 文件夹 /sbin system binaries，只有 root 用户能用？ /tmp 系统文件，但是在 RAM 里不在 disk 上，所以重开机的时候会消失 /var logs, caches，这种存在很久的文件 /dev /sys /proc 虚拟文件系统？“虚拟”是指 /dev、/sys 和 /proc 这些目录并非存储在硬盘上，而是在内存中动态生成的，它们是内核为用户和应用程序提供的一种抽象

`/dev`、`/sys` 和 `/proc` 是 Linux 系统中的特殊目录，它们提供了对系统信息和硬件设备的访问。这些目录在 Linux 中扮演了重要的角色，用于与内核交互、访问硬件设备和获取系统信息。

1. `/dev` 目录：

- `/dev` 目录包含了设备文件，这些文件用于与系统上的硬件设备进行通信。例如，硬盘分区、终端设备、声卡等都在 `/dev` 目录下有相应的设备文件。通过这些设备文件，用户和应用程序可以访问和控制硬件设备。
- 例如，硬盘分区 `/dev/sda1` 可以代表硬盘的第一个分区。

2. `/sys` 目录：

- `/sys` 是一个用于与内核进行交互的虚拟文件系统。它提供了一种方式，通过读写这些文件，可以动态地配置和获取有关内核的信息。这对于管理和监控内核参数和设备状态非常有用。
- 例如，你可以通过 `/sys/class/power_supply` 目录下的文件获取电源供应信息。

3. `/proc` 目录：

- `/proc` 是一个虚拟文件系统，提供了对正在运行的内核和进程的信息的访问。通过 `/proc`，你可以查看系统信息、进程信息等。这些信息以文件和目录的形式呈现，用户可以通过读取这些文件获取有关系统状态和进程的信息。
- 例如，你可以通过 `/proc/cpuinfo` 查看有关 CPU 的信息，通过 `/proc/meminfo` 查看有关内存的信息。

这些特殊目录在 Linux 系统中具有重要的功能，它们为用户和系统工具提供了访问底层系统信息和硬件设备的途径。

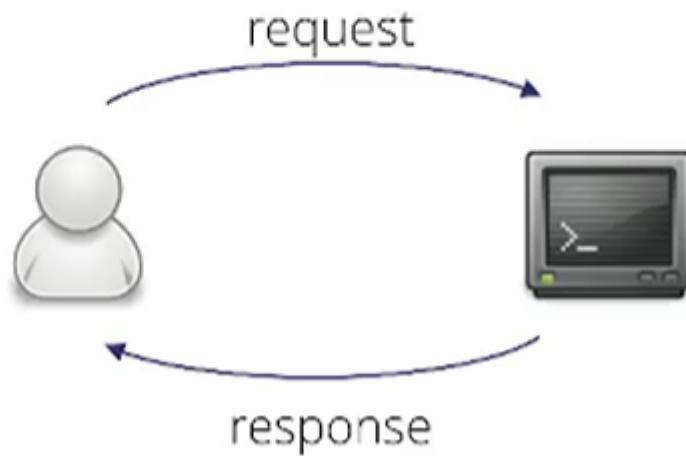
2.POSIX Shell

2.1 The Shell

Slides

Shell: CMD? Powershell?

shell workflow



prompt: 用户输入提示

示 但实际上

- \$ You are in a shell, most likely POSIX (sh) compatible.
- # You are in a root shell. With great power comes great responsibility.
- % You are probably in the C shell.
- > You are on a continuation line e.g. inside a string.

\$确实

```
[if23696@it075697 ~]$  
[if23696@it075697 ~]$ cd ..
```

是Linux相关的 >在‘’中换行

- [if23696@it075905 tools]\$ echo 'smile
> if
> you'
smile
if
you

builtin commands:

- [if23696@it075905 tools]\$ which cd
/usr/bin/cd
- [if23696@it075905 tools]\$ which ls
alias ls='ls --color=auto'
/usr/bin/ls

which cd是

- [if23696@it075905 tools]\$ ls
arguments 'silly name'
arguments.c 'silly name.c'
empty words
- [if23696@it075905 tools]\$ ls -a
. empty
.. 'silly name'
arguments 'silly name.c'
arguments.c words
- [if23696@it075905 tools]\$ ls -l
total 452
-rwxr-xr-x. 1 if23696 coms23t 18048 Jan 26 16:30 arguments
-rw-r--r--. 1 if23696 coms23t 152 Jan 26 16:31 arguments.c
drwxr-xr-x. 2 if23696 coms23t 4096 Jan 26 15:55 empty
-rwxr-xr-x. 1 if23696 coms23t 18048 Jan 26 16:32 'silly name'
-rw-r--r--. 1 if23696 coms23t 100 Jan 29 18:14 'silly name.c'
-rw-r--r--. 1 if23696 coms23t 409048 Sep 13 2004 words

一些flag什么的 -l: 长格式显示信息 -a: 显示所有文件，包括一些... 是隐藏文件

```
$ ls --help  
BusyBox v1.30.1 multi-call binary.
```

会列出各种flag代表

什么意思 **Manuals**:

```
man [选项] 命令或主题
```

例如，要查看`ls`命令的手册页，可以执行以下命令：

```
bash
```

```
man ls
```

实际上是在查看手册

Section 1 is shell commands, section 2 system calls,
section 3 the C library etc.

e.g. `man 1 printf` and `man 3 printf` are different.

Shell Expansions:

1. *: 匹配零个或多个字符 比如说: `ls *.txt` 匹配所有.txt开头的文件 `ls *abc` 匹配所有abc开头的文件 (所以*的位置和开头结尾有关)

2. ? 匹配一个字符

1. 匹配单个字符：

```
bash
```

 Copy code

```
echo file?.txt
```

这将匹配类似 `file1.txt`、`fileA.txt` 等的文件名，其中 `?` 匹配一个字符。

2. 在范围内使用：

```
bash
```

 Copy code

```
echo file[0-9]?.txt
```

这将匹配类似 `file1A.txt`、`file2B.txt` 等的文件名，其中 `[0-9]` 匹配一个数字，而 `?` 匹配一个字符。

3. 逐个字符匹配：

```
bash
```

 Copy code

```
echo file?.txt
```

这将匹配类似 `file1.txt`、`fileA.txt` 等的文件名，但不会匹配 `file10.txt`，因为 `?` 只匹配一个字符。

有点不确定字符的意思，可以是任何？

3. [ab]单个字符范围匹配

1. 单个字符匹配：

```
bash
```

 Copy code

```
echo [abc]
```

这将匹配字符集合中的任何一个字符，输出可能是 `a`、`b` 或 `c`。

2. 范围匹配：

```
bash
```

 Copy code

```
echo [0-9]
```

这将匹配任何数字字符，输出可能是 `0`、`1`、`2`，依此类推。

3. 反转字符集合：

```
bash
```

 Copy code

```
echo [!abc]
```

这将匹配不在字符集合中的任何字符。

4. 多个字符集合：

```
bash
```

 Copy code

```
echo [a-zA-Z]
```

这将匹配任何字母字符，无论是小写  或大写。

4. \$变量

```
my_variable="Hello"  
echo $my_variable
```

Shell Quoting "Double Quotes" 字符被原样输出，特殊字符被解释 'single Quotes' turn Off everything：所有东西都会被原样输出 * \? \[\] \\$ Do not treat as pattern 特殊字符不作为特殊字符处理

1. 单引号 (Single Quotes) :

使用单引号将一串字符括起来，单引号内的所有字符都会被原样输出，不进行任何解释。例如：

```
bash
```

 Copy code

```
echo 'Hello, $USER'
```

这将输出 `Hello, \$USER`，而不会替换 `\$USER`。

2. 双引号 (Double Quotes) :

使用双引号将一串字符括起来，双引号内的特殊字符会被解释，变量会被展开。例如：

```
bash
```

 Copy code

```
greeting="Hello"  
echo "$greeting, $USER"
```

这将输出 `Hello, [当前用户]`。

3. 反斜杠 (Backslash) :

使用反斜杠 `\\` 对特殊字符进行转义，使其失去特殊含义。例如：

```
bash
```

 Copy code

```
echo "This is a \"quoted\" word."
```

这将输出 `This is a "quoted" word.`。

cp mv find echo cp: copy

1. 简单的文件复制:

bash

 Copy code

```
cp file.txt /path/to/destination/
```

将 `file.txt` 复制到 `/path/to/destination/`。

2. 递归复制目录及其内容:

bash

 Copy code

```
cp -r directory /path/to/destination/
```

将 `directory` 及其内容复制到 `/path/to/destination/`。

3. 强制覆盖目标文件:

bash

 Copy code

```
cp -f file.txt /path/to/destination/
```

将 `file.txt` 强制复制到 `/path/to/destination/`，即使目标文件已经存在。

4. 交互式复制:

bash

 Copy code

```
cp -i file.txt /path/to/destination/
```

如果目标文件已经存在，将会询问用户是否覆盖。

mv:move

mv [-nf] SRC... DEST move files

- n** no overwrite
- f** force overwrite

find:search

```
find . -name "a*"
```

ChatGPT

您提供的命令是使用 `find` 命令在当前目录及其子目录中查找文件名以字母 "a" 开头的所有文件。以下是该命令的一些关键部分的解释：

- `.`.: 当前目录。`find` 命令从当前目录开始递归查找。
- `-name "a*"`: 用于指定匹配文件名的模式。在这里，它表示查找文件名以字母 "a" 开头的所有文件，后面的 `*` 表示任意字符（包括空字符）可以跟随 "a"。

因此，整个命令 `find . -name "a*"` 表示在当前目录及其子目录中查找所有文件名以字母 "a" 开头的文件。

echo:print

```
$ cp index.html style.css web  
$ cp * web
```

in empty folder:

```
$ cp * web  
cp: can't stat '*': No such file or  
directory
```

\$ **find** DIR [EXPRESSION]

find all files in directory (recursively)
that match an expression

e.g. **find** . -name "a*"

Exercise

whitespace:

- [if23696@it075905 tools]\$./arguments one two
Argument #0: [./arguments]
Argument #1: [one]
Argument #2: [two]
- [if23696@it075905 tools]\$./arguments "one two"
Argument #0: [./arguments]
Argument #1: [one two]
- [if23696@it075905 tools]\$./arguments "one two"
Argument #0: [./arguments]
Argument #1: [one two]

意思是，只有在“”里面，space也算 **Pattern matching**:

- [if23696@it075905 tools]\$./arguments *

Argument #0: [./arguments]
Argument #1: [arguments]
Argument #2: [arguments.c]

列出

所有的文件

- [if23696@it075905 empty]\$../arguments *
Argument #0: [../arguments]
Argument #1: [*]

只显示了*，“empty”文件夹为空，因此 ../arguments * 命令在该文件夹中执行时，shell 将 * 视为一个参数，而不是展开为文件名列表。

find three different ways to get the program to produce the following output:

```
Argument #0: [./arguments]
Argument #1: [*]
```

- [if23696@it075905 tools]\$./arguments *
 Argument #0: [./arguments]
 Argument #1: [*]
- [if23696@it075905 tools]\$./arguments "*"
 Argument #0: [./arguments]
 Argument #1: [*]
- [if23696@it075905 tools]\$./arguments '*'
 Argument #0: [./arguments]
 Argument #1: [*]

Files with spaces in their names

Files with spaces in their names

The command `touch FILENAME` creates a file. Create a file with a space in its name by typing `touch "silly named file"`. What would happen if you left the quotes off (you can try it, then do `ls`)?

Start typing `ls sill` and then press TAB to autocomplete. Assuming you have no other files whose name starts with `sill`, what happens? Use this method to get the arguments program to print the following:

```
Argument #0: [./arguments]
Argument #1: [Hello world!]
```

The command `rm` (remove) deletes files again. Use it to remove your file with spaces in its name, using one of several methods to get the shell to pass the spaces through to `rm`.

- [if23696@it075840 tools]\$ touch silly named file
- [if23696@it075840 tools]\$ ls
 arguments arguments.c empty file named silly 'silly named file'
- [if23696@it075840 tools]\$ ls silly
 silly
- [if23696@it075840 tools]\$ ls "silly named file"
 'silly named file'

touch "Silly named file"->创建一个文件夹? touch Silly named file->创建3个文件 Silly named file ls sill+tab:

```
[if23696@it075905 tools]$ ls silly\ named\ file
'silly named file'
```

rm: rm sill+tab 或者 rm "silly named file"

Shell variables 注意:

- ▶ [if23696@it075905 tools]\$ program="silly name"
- ▶ [if23696@it075905 tools]\$ gcc -Wall \$program.c -o \$program

```
]$ program="silly names"  
]$ gcc -Wall "$program.c" -o "$program"
```

上面那个无

法编译，因为被视作silly, name.c 意思就是，在program赋值和后续调用的时候都需要用到“”

2.2 Pipes

Slides

其实还是讲过unix philosophy的：

1. Each program should do one thing well.
2. Programs should be able to cooperate to perform larger tasks.
3. The universal interface between programs should be a text stream.

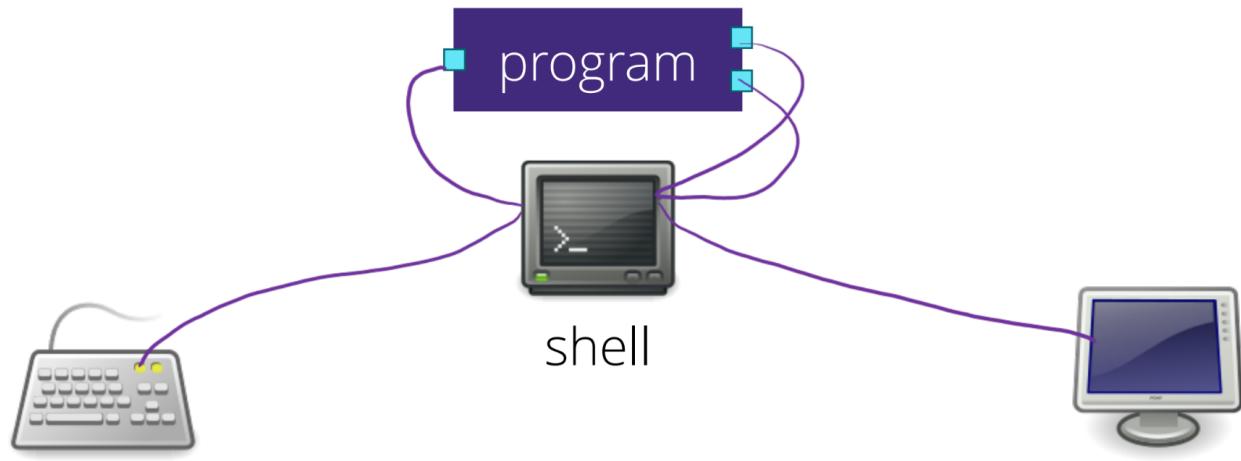
1.每个program都应该do one thing well 2.programs 可以联合起来完成更大的tasks 3.universal interface
shoulder be a text stream (文本流，是一种将文本数据视为连续流动的概念。Unix系统的设计哲学之一是将简单的工具通过标准输入 (stdin) 和标准输出 (stdout) 连接起来，形成一个数据处理管道。) 讲下标准输入和标准输出：

Each program starts with three file descriptors open:

- 0 = standard input
- 1 = standard output
- 2 = standard error



Running a program in the terminal:



在计算机科学中，管道（Pipe）是一种用于在两个进程之间传递数据的机制。它允许一个进程的输出（stdout）直接成为另一个进程的输入（stdin），从而使这两个进程协同工作。



ChatGPT

'Pipe' ('|') 和 'redirection' ('>', '>>', '<') 是在 Unix/Linux 系统中用于处理命令输入输出的不同机制。

1. Pipe ('|'):

- **作用：**允许将一个命令的输出传递给另一个命令的输入，从而创建一个命令链。
- **例子：**

```
bash                                     Copy code
                                         ↗
command1 | command2
```

- **解释：** `command1` 的输出成为 `command2` 的输入。这对于将多个命令连接起来以完成复杂的任务很有用。

2. Redirection ('>', '>>', '<'):

- **作用：**

- `>`：将命令的输出重定向到文件，如果文件存在则覆盖其内容。
- `>>`：将命令的输出追加到文件末尾，如果文件不存在则创建。
- `<`：将文件的内容作为命令的输入。

- **例子：**

```
bash                                     Copy code
                                         ↗
command > output.txt
command >> output.txt
command < input.txt
```

pipe是将一个文件的输出变成另一个文件的输入 > >> < 这些重定向（redirection）是将输出等追加到文件或者作为输入

错误输出 `COMMAND > FILE 2> FILE2` 将 `COMMAND` 的标准输出到 `FILE`, 然后错误输出到 `FILE2` `COMMAND>FILE 2>&1` `COMMAND` 的标准输出定向到 `FILE`, 然后错误也定向到 `&1` 所在的文件 `COMMAND >>FILE 2>&1` `COMMAND` 指令的标准输出和标准输入都到 `FILE` 中, 但是不覆盖

但是, 这个是错的 `COMMAND 2>&1>FILE` 没有先规定 1 到哪, 所以是从左到右顺序 (?) `COMMAND > /dev/null` ignore output

FILES and STREAMS `PROGRAM > FILE` 标准输出到 `FILE` `PROGRAM >> FILE` 标注输出到 `FILE` 但是不覆盖
`PROGRAM <FILE FILE` 作为标准 input 给 `COMMAND` 执行 `PROGRAM 2> FILE` 标准错误输出到 `FILE` 还有一个 subshell: `COMMAND $(SOMETHING)`

bash

```
# 假设 SOMETHING 的输出是一个字符串
```

```
result=$(SOMETHING)
```

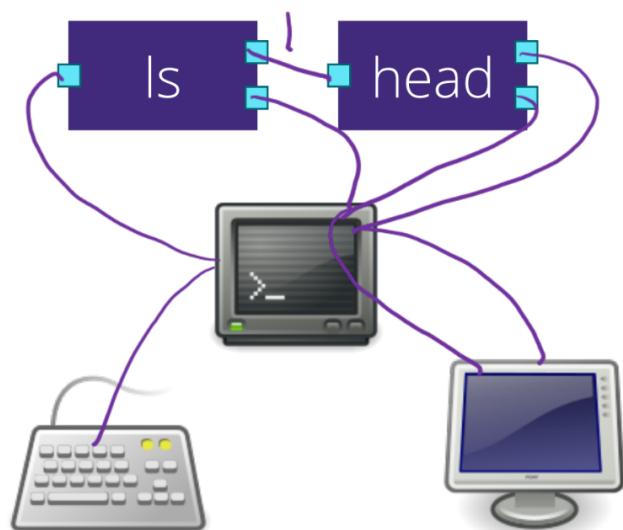
```
# 将子Shell 的输出作为参数传递给 COMMAND
```

```
COMMAND "$result"
```

Pipes

```
$ ls -1 | head
```

```
head [-n NUM]
tail [-n NUM]
```



列出头/尾巴 `ls -1 | head` 从头开始列 `ls -1| head -n 6` 从头开始列, 列 6 个结束 `ls -1 | tail`

在 `ls` 命令中，选项 `--1`（即数字 1）指定以单列的方式显示文件和目录，

grep sort uniq

1. grep: 文本搜索 "global regular expression parser"

```
bash
grep [options] pattern [file...]
```

- `pattern`: 要搜索的模式，可以是简单字符串或正则表达式。
- `file`: 要搜索的文件，如果省略，则默认从标准输入读取。

一些常用的 `grep` 选项：

- `--i`: 忽略大小写。
- `--r` 或 `--R`: 递归地搜索子目录。
- `--n`: 显示匹配行的行号。
- `--v`: 显示不匹配模式的行。
- `--c`: 只显示匹配的行数。

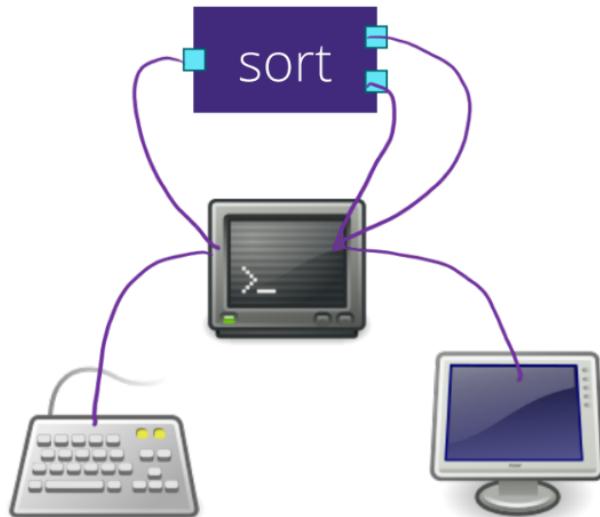
2. sort: read all lines into buffer, sort, output

\$ sort

aaa
ccc
bbb

^D
aaa
bbb
ccc

\$



`-n`: 按数字顺序排序。

`-r`: 降序排列。

`-u`: 仅显示唯一的行（删除重复行）。

`-k <key>`: 按指定键进行排序，例如 `-k 2` 表示按第二列进行排序。

默认：字母顺序

1. 按字母顺序排序：

```
bash  
  
sort file.txt
```

2. 按数字顺序排序：

```
bash  
  
sort -n numbers.txt
```

3. 降序排序：

```
bash  
  
sort -r file.txt
```

4. 按第二列排序：

```
bash  
  
sort -k 2 data.txt
```

3. uniq remove duplicates immediately following best used as: command | sort | uniq

一些常见的 `uniq` 选项包括：

- `'-c'`：显示每行重复出现的次数。
- `'-d'`：仅显示重复的行。
- `'-i'`：忽略大小写。
- `'-u'`：仅显示不重复的行。

1. 删除重复的行:

```
bash
```

```
sort file.txt | uniq
```

2. 显示每行重复出现的次数:

```
bash
```

```
sort file.txt | uniq -c
```

3. 仅显示重复的行:

```
bash
```

```
sort file.txt | uniq -d
```

4. 仅显示不重复的行:

```
bash
```

```
sort file.txt | uniq -u
```

5. 忽略大小写进行处理：

bash

```
sort file.txt | uniq -i
```

tee

`tee` 命令用于从标准输入读取数据并将其写入标准输出，同时将其复制到指定的文件或文件描述符。`tee` 命令通常与管道结合使用，以便查看命令的输出同时保存到文件中。

基本语法如下：

bash

 Copy code

```
command | tee filename
```

这会将 `command` 命令的输出同时显示在终端上并写入 `filename` 文件中。

一些例子：

L. 将命令的输出显示在终端上，并保存到文件：

bash

 Copy code

```
ls -l | tee filelist.txt
```

这个命令列出当前目录下的文件，并将输出显示在终端上，同时保存到名为 `filelist.txt` 的文件中。

感觉像是又保存到另一个文件夹里，又print到屏幕上 less 在shell中打开某个文本 sed 替换替换 用表达式 s/ONE/TWO/[g] 没有g只是替换第一个匹配到的，有了g就是全局替换 subshell

Exercise

1. **cat**: 打开文件 cat 命令用于将一个或多个文件的内容输出到标准输出。例如：cat file1.txt file2.txt 将显示 "file1.txt" 和 "file2.txt" 的内容。
2. **tail**: 显示末尾 tail 命令用于显示文件的末尾部分，默认显示前 10 行。例如：tail -n 20 file.txt 将显示文件 "file.txt" 的后 20 行。tail 命令用于显示文件的末尾部分，默认显示最后 10 行。例如：tail -n 30 file.txt 将显示文件 "file.txt" 的后 30 行。
3. **sort**: 排序 sort 命令用于对文本进行排序，默认按字典顺序排序。例如：sort file.txt 将对文件 "file.txt" 的内容进行排序。

4. **uniq**:删除重复行 uniq 命令用于删除或显示文本中的重复行。例如: sort file.txt | uniq 可以用来删除文件 "file.txt" 中的重复行。
5. **grep**:搜索匹配 grep 命令用于在文本中搜索匹配的行。例如: grep "pattern" file.txt 将显示文件 "file.txt" 中包含 "pattern" 的所有行。
1. **sed**:编辑 sed 命令用于对文本进行流编辑, 根据指定的命令进行转换。例如: sed 's/old/new/' file.txt 将把文件 "file.txt" 中的每个匹配到的 "old" 替换为 "new"。
6. **wc**:统计 wc 命令用于计算文件或标准输入中的字数、行数和字符数。例如: wc -l file.txt 将显示文件 "file.txt" 中的行数。

If English is not your native language, ignore the guessing part - it is not assessed.

1. The first word in the file. Can you guess what it will be? 打开文件: cat | 找头:head cat words | head -n 1

● [if23696@it075865 tools]\$ cat words|head -1
Aarhus

2. The last word in the file. Can you guess this one? cat words | tail -n 1

● [if23696@it075865 tools]\$ cat words|tail -1
Zurich

3. The number of words in the words file - there is one word per line. cat words | wc -l 统计: wc

● [if23696@it075865 tools]\$ cat words|wc -l
45402

4. The 6171st word in the file. Can you read my mind and guess this word directly? 搜索: grep(为什么是 sed?) sed -n '6171p' words

这个 `sed` 命令的含义如下:

```
bash
Copy code
sed -n '6171p' words
```

- ` -n`: 只打印模式匹配的行。
- `6171`: 选择文件中的第 6171 行。
- ` 'p' `: 打印模式匹配的行。

这个 `sed` 命令的目的是从文件 `words` 中选择并打印第 6171 行的内容。` -n` 选项会使 `sed` 只打印经过模式匹配的行, 而 ` '6171p' ` 是一个模式, 表示选择并打印第 6171 行。

● [if23696@it075865 tools]\$ cat words | sed -n '6171s/[[:space:]]\+/&/p'
Bristol

5. All words containing the letter Q, capitalised. (A regular expression containing a string of one or more letters matches all strings that contain the expression as a substring.) 替换: sed \$'s/q/Q/g'

`words sed -i 's/q/Q/g' your_file.txt` 不像显示打印就是这样

```
◎ [if23696@it075865 tools]$ awk '{ for (i=1; i<=NF; i++) if ($i ~ /Q/) $i = toupper($i) } 1' words
```

6. All words starting with the letter X. The regular expression X would match an X anywhere in the word, but ^X matches an X only at the start of the string. 查找: grep `grep '^X' words` sed -n `'/^X/p'` words -n 指只打印匹配行 -i 指不显示打印 s/[]/[]/gs: 替换; g: 全局替换(global) /^[]/p^: 是指开头是什么; p: 打印(`printf`)

- [if23696@it075865 tools]\$ grep -o '\bX\w*' words

Xanthus

Xavier

Xebec

Xenakis

Xenia

Xenix

Xerox

Xeroxed

Xeroxes

Xeroxing

Xerxes

Xhosa

7. All words ending in j. (The expression 'j\$' matches a j only at the end of the string, but you have to single-quote it to stop the shell from interpreting the dollar sign). 查找: grep `grep 'j$' words` sed -n `'/j$/p'` words

- [if23696@it075865 tools]\$ grep -o '\w*j\b' words

Cluj

8. The number of words containing the letter Q, ignoring case (e.g. capitalised or not). 查找: grep | 统计: wc `grep -i 'q' words | wc -l` -i 忽略大小写 ignore -o 仅输出匹配部分 only

- [if23696@it075865 tools]\$ grep -o -i '\w*Q\w*' words | wc -l

681

9. The first five words containing the letter sequence cl. 查找: grep `grep 'cl' words | head -n 5`

10. All words containing the sequence "kp", but not "ckp". Can you guess any of these? 感觉是正则表达式

- [if23696@it075865 tools]\$ grep -o '\w*cl\w*' words | head -n 5

acclaim

acclaimed

acclaiming

acclaims

acclamation

查找: grep `grep 'kp' words | grep -v 'ckp'` -v 代表反向匹配 (指排除?)

1. The last 15 words of exactly two letters. The expression . (period) matches a single character, and '^...\$' for example would match all strings of the format exactly three characters between start and end of string. You need to quote it because of the dollar sign. `grep '^..'$ | tail -n 15`

```
``` grep -o '^..'$ words | tail -n 15 ````
```

一个.代表一个字符

2. All words from the first 100 words on the list, which contain the letter y. `cat words | head -n 100 | grep 'y' cat words | head -n 100 | grep 'y'`

3. The first five words that are among the last 100 words on the list, and contain the letter y (whether capitalised or not). `tail -n 100 | grep -i 'y' | head -n 5 tail -n 100 words | grep -i 'y' | head -n 5`

4. All three-letter words with no vowels (aeiou).The regular expression '[aeiou]' matches any string that contains one of the bracketed characters; you need quotes to stop the shell from interpreting the brackets. Remember to exclude words with capitalised vowels as well. There are 343 of these. `grep '^...$' | grep -g -v '[aeiou]' words | wc -l`

- [if23696@it075903 tools]\$ grep '^...\$' words | grep -v '[aeiou]' | wc -l  
22

----22 grep -i -w -v '[aeiou]' words | grep -w '^...\$' | wc -l----536 ?????????? grep -Ei ^[aeiou]{3}\$ words

5. All words of exactly 7 letters, where the third one is an e and the word ends "-ded". This kind of search is really useful for crosswords. There are 14 words of this form, can you guess them? `grep -i '^...e.ded$' words`-----4????? grep -iE ^[a-z]{2}e[a-z]{1}ded\$ words -E是扩展正则表达式

## 2.2 Regular expressions

1. Study the documentation for the -w option. Contrive a file such that grep PATTERN FILE returns two different lines but grep -w PATTERN FILE returns only one line.

- [if23696@it075903 tools]\$ grep pattern ex1  
**pattern**  
**patterned**
- [if23696@it075903 tools]\$ grep -w pattern ex1  
**pattern**

```
plaintext
```

 Copy code

```
pattern
```

```
patterned
```

现在，如果运行命令 `grep 'pattern' file`，它会匹配两行，因为模式在两行中都出现。但是，如果运行命令 `grep -w 'pattern' file`，它只会匹配第一行，因为 `-w` 选项要求模式形成一个完整的单词，而在第二行中，'pattern' 是更大单词 'patterned' 的一部分。

就是 -w 是完全匹配，不加是部分匹配的意思

2. You'll have seen beforehand that you can count the results of a search with grep PATTERN FILE | wc -l. However, grep also has a -c option which counts matches. Can you find the situation where the wc -l approach and the -c approach produce different results? Can you explain why?  统计的是行数  
是的，有一些情况下，使用 `wc -l` 和 `grep -c` 可能会产生不同的结果。这通常发生在一行中包含多个搜索模式的情况下。

第一个命令 (`grep 'pattern' example.txt | wc -l`) 将计算包含模式 'pattern' 的行数，并返回 '1'，因为模式在一行上出现一次。

第二个命令 (`grep -c 'pattern' example.txt`) 将计算模式 'pattern' 的总出现次数，并返回 '2'，因为在同一行上有两个出现。

让我们考虑一个名为 `example.txt` 的示例文件：

```
plaintext
```

```
patternpattern
```

(????)但是实际输出是一样的哦 3. Some words have different spelling between British English and American English. For example, 'encyclopaedia' is valid in British English but not American. Can you write a regular expression that would match both of these words, but nothing else? How about matching both 'color' (American) and 'colour' (British)?  ex3

4. UK postcodes follow a general schema of two letters followed by one number, followed by an optional space, then another number, followed by two more letters. Can you write a regular expression that would match such sequences? `^[A-Z]{2}\d{1,2} ?\d[A-Z]{2}$`

在这里，`\\d{1,2}` 表示一个或两个数字，类似于使用 `...` 表达一个范围。其他部分与之前的解释一致。这个正则表达式可以匹配符合指定UK邮政编码格式的字符串。



You

`\\d`是什么



ChatGPT

`\\d` 是一个正则表达式中的元字符，表示匹配一个数字（0-9）。在正则表达式中，`\\d` 用于匹配任意数字字符。

因此，`\\d{1,2}` 表示匹配一个或两个数字。如果要匹配一个数字，可以使用 `\\d`，如果要匹配两个数字，可以使用 `\\d\\d` 或 `\\d{2}`。



5. In practice, the above is a simplified version of the system, and a better UK postcode validator regex is known to be `^(([A-Z]{1,2}[0-9][A-Z0-9]?|ASCN|STHL|TDCU|BBND|[BFS]IQQ|PCRN|TKCA) ?[0-9][A-Z]{2}|BFPO ?[0-9]{1,4}|(KY[0-9]|MSR|VG|AI)[-]?[0-9]{4}|[A-Z]{2} ?[0-9]{2}|GE ?CX|GIR ?0A{2}|SAN ?TA1)$`. Try breaking apart this monster to understand what is being tested, and find an example that would match the schema described for the fourth question but fail to match this expression.

```
^(
 ([A-Z]{1,2}[0-9][A-Z0-9]?|ASCN|STHL|TDCU|BBND|[BFS]IQQ|PCRN|TKCA) ?[0-9][A-Z]{2}
 |BFPO ?[0-9]{1,4}
 |(KY[0-9]|MSR|VG|AI)[-]?[0-9]{4}
 |[A-Z]{2} ?[0-9]{2}
 |GE ?CX|GIR ?0A{2}|SAN ?TA1
)$
```

## Lesson Notes

abc... *Letters*

123... *Digits*

`\\d` *Any Digit*

\D	<i>Any Non-digit character</i>
.	<i>Any Character</i>
\.	<i>Period</i>
[abc]	<i>Only a, b, or c</i>
[^abc]	<i>Not a, b, nor c</i>
[a-z]	<i>Characters a to z</i>
[0-9]	<i>Numbers 0 to 9</i>
\w	<i>Any Alphanumeric character</i>
\W	<i>Any Non-alphanumeric character</i>
{m}	<i>m Repetitions</i>
{m,n}	<i>m to n Repetitions</i>
*	<i>Zero or more repetitions</i>
+	<i>One or more repetitions</i>
?	<i>Optional character</i>
\s	<i>Any Whitespace</i>
\S	<i>Any Non-whitespace character</i>
^ ... \$	<i>Starts and ends</i>
(...)	<i>Capture Group</i>
(a(bc))	<i>Capture Sub-group</i>
(.*)	<i>Capture all</i>
(abc def)	<i>Matches abc or def</i>

• `(\w+ (\d+))`:

- `\w+`: 匹配一个或多个字母、数字或下划线字符。
- `( )`: 括号用于捕获匹配的内容，这里将整个表达式包裹在括号中，以捕获整个匹配。
- 空格：匹配一个空格字符。
- `(\d+)`: 与前面的类似，匹配一个或多个数字。

[https://regexone.com/lesson/nested\\_groups](https://regexone.com/lesson/nested_groups) 总之可以玩玩这个

Exercise 8: Matching Optional Characters

Task	Text
Match	1 file found?
Match	2 files found?
Match	24 files found?
Skip	No files found.

\d+ files? found\?

Continue >

## Week 2: Version Control

### 3.Git

#### Slides

git我个人理解大概就是一个版本控制的东西大概分为以下几类

#### 1.个人本地仓库

一般是`git init`初始化, `git add`之后`git commit`一下然后查看`git status`状态, 和`git log`查看版本控制日志`git checkout HASH`退回版本`git tagname` specific commits, 只是给commits打上tag, 但是不修改commits内容`commits相关内容` `git checkout HEAD -- hello.c` 放弃对文件的修改, 回到上一个head的地方`git reset --hard HEAD` `git clean -dfx` 这一系列命令将工作目录重置为与 HEAD 提交相匹配的状态, 放弃所有更改(包括暂存和未暂存的更改), 并删除未跟踪的文件。`reset`有三种形式`--soft`\默认\`--hard`

##### 1. Soft Reset:

- `git reset --soft <commit>` 移动 HEAD 指针到指定的`<commit>`, 但是不修改暂存区和工作目录。这使得你可以重新提交这些更改, 而不需要重新编辑 commit 信息。暂存区和工作目录中的更改仍然保留, 只是它们现在包含在新的提交中。

##### 2. Mixed Reset(默认):

- `git reset <commit>` 是默认的`reset`行为。它移动 HEAD 指针到指定的`<commit>`, 同时更新暂存区以反映该 commit 的状态。但它不修改工作目录, 所以你会失去工作目录中未暂存的更改。这相当于取消暂存区的更改, 但保留工作目录中的更改。

##### 3. Hard Reset:

- `git reset --hard <commit>` 是最强烈的`reset`形式。它移动 HEAD 指针、更新暂存区, 并彻底丢弃工作目录中的所有更改, 使其与指定 commit 一致。使用这种方式要非常小心, 因为它会永久删除未提交的更改。

暂存区(staging area/index): `git add .`之后文件就会放在staging area 工作目录: 工作目录的改变就是, 比如我用vscode修改了某个代码, 然后我用`git reset --hard`, 实际上我vscode里面的代码也会变回去, 这个就是工作目录`reset`和`checkout`的区别就在于修改的是什么东西

- `git checkout HEAD` 主要用于撤销工作目录中对文件的修改，不会修改暂存区和 commit。
- `git reset --soft HEAD` 将 `HEAD` 指针回退一个 commit，但保留工作目录和暂存区的更改。
- `git reset` 默认是 mixed mode，移动 `HEAD` 指针到指定 commit，更新暂存区，但不修改工作目录。

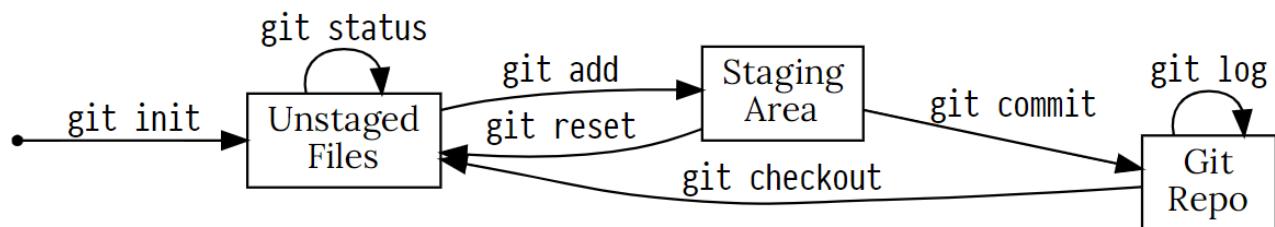
`git checkout HEAD~1` 是 `HEAD` 的上个，同理如果是 `HEAD~2` 的话就是上上个 `git revert HEAD`

1. 当你运行 `git revert HEAD` 后，Git 会创建一个新的提交，该提交包含了撤销 `HEAD` 指向的最近一次提交所做的更改。
2. 这个新的提交将被添加到分支上，成为新的最新提交。

这种方式与 `git reset` 不同，`git reset` 会直接修改分支的历史，而 `git revert` 会创建一个新的提交来撤销之前的提交。这对于共享仓库的协作是更安全的，因为它不会改变历史记录，而是在历史上添加一个新的撤销提交。

总的来说，`git revert HEAD` 是一种撤销最近提交的安全方式，适用于已经推送到共享仓库的场景，避免修改已有的提交历史。

对于远程仓库更加安全？



所以status和log是这个意思

## 2.branch的意思

一般有 `main/master` 这种主枝干和 `branch` 这类分支，更像是团队合作的东西 `git branch NAME` (创建一个新的分支) `git branch NEW main` 从 `main` 当前 `HEAD` 分一条 `branch` 出来 `git checkout NAME` (切换到这个分支)

## git cherry-pick [hashcode]

`git cherry-pick` 是一个用于选择并应用其他分支上的单个提交的 Git 命令。这个命令将指定的提交复制到当前所在的分支上。它的语法如下：

bash

 Copy code

```
git cherry-pick <commit-hash>
```

其中 `<commit-hash>` 是要应用的提交的哈希值。

使用 `git cherry-pick` 的主要场景是在不合并整个分支的情况下，将其他分支上的特定提交应用到当前分支。这在需要引入特定功能或修复特定 bug 时很有用。

步骤如下：

1. 切换到目标分支：`git checkout <target-branch>`
2. 运行 `git cherry-pick <commit-hash>`，将目标分支上的特定提交复制到当前分支。
3. 如果存在冲突，需要手动解决冲突并使用 `git cherry-pick --continue` 继续 cherry-pick 过程。

请注意，`git cherry-pick` 会创建新的提交，而不是简单地将提交复制过去。这是因为提交的哈希值是由其内容和父提交的哈希值计算而来的，所以不同的分支上可能存在相同的提交，但它们的哈希值是不同的。



## 3. 和远程仓库链接

git clone 本身是本地文件拷贝 然后 ppt 大概讲了下本地修改的两种方法：Bob 下载了 Alice 的文件，并进行修改 Bob's end: git format-patch origin/main \--to=alice@bristol.ac.uk 0001-Fixes-spelling-mistake.patch Alice's end: git am ../bob/0001-Fixes-spelling-mistake.patch git log --oneline  
这个其实我觉得不重要...主要是后面远程仓库 git remote add origin <远程仓库地址> 连接到远程仓库 git remote -v 验证链接 git push -u origin main 推送到远程仓库，然后 -u 简化以后的 push 和 pull

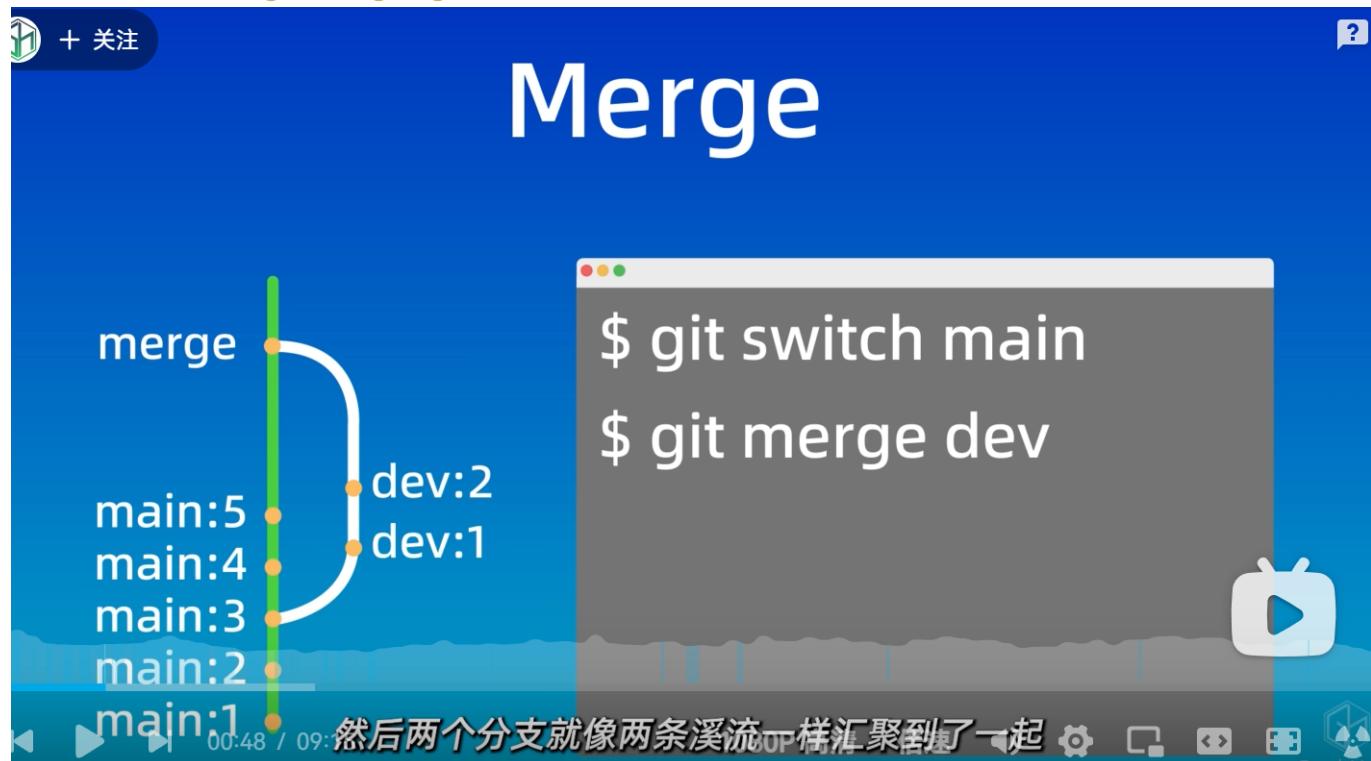
这个命令使用 `git format-patch` 生成一个补丁文件，该文件包含了你当前分支与 `origin/main` 分支之间的变更。生成的补丁文件随后会被发送到一个邮箱地址 (`alice@bristol.ac.uk`)。

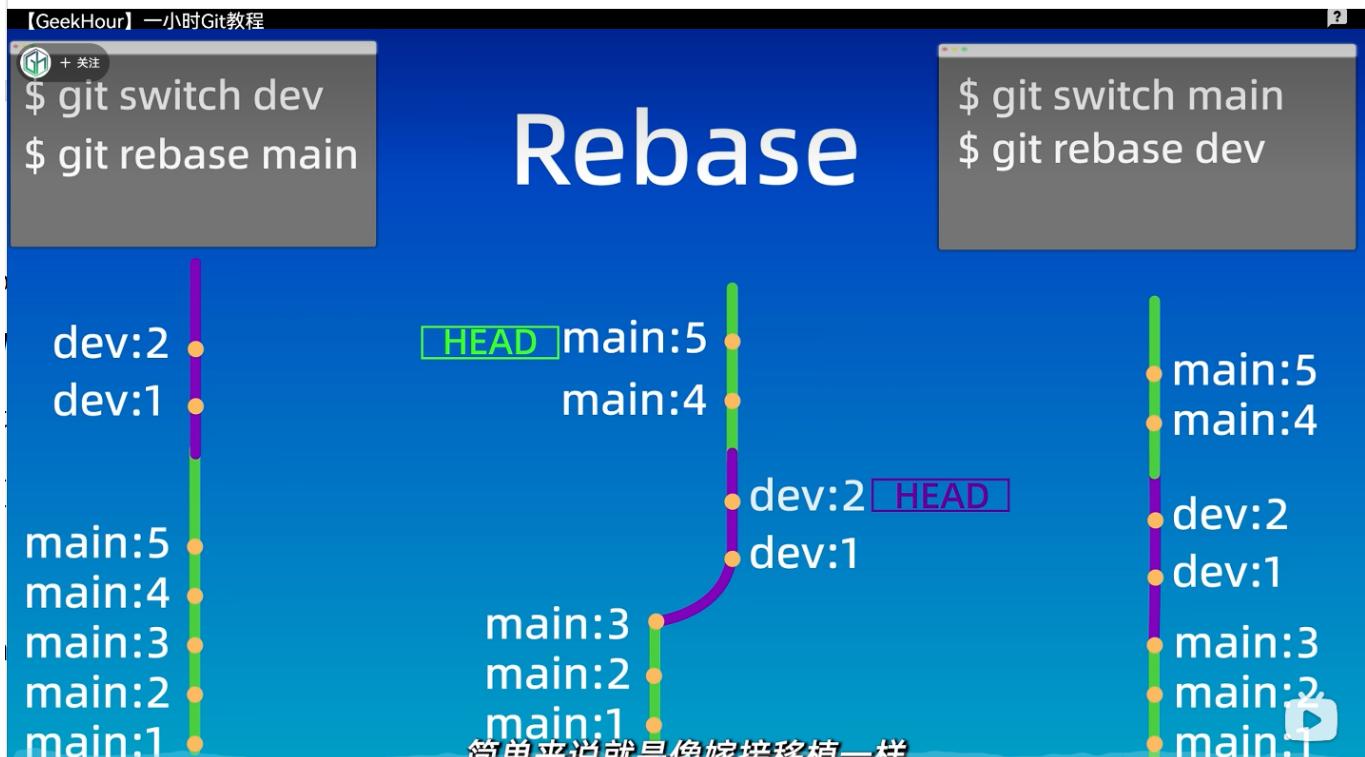
下面是命令的具体解释：

- `git format-patch origin/main`：这个命令为你当前分支与 `origin/main` 分支之间的变更生成一系列的补丁文件。每个提交都会生成一个补丁文件。
- `--to=alice@bristol.ac.uk`：此选项指定将补丁文件发送到的邮箱地址。生成的补丁文件将作为附件随邮件发送。
- `0001-Fixes-spelling-mistake.patch`：这是给补丁文件命名的方式。它以 `0001` 开头，因为它是生成的第一个补丁文件。通常，名称是根据提交消息或提交的信息生成的。

#### 4.branch的合并

其实昨天大概看到了 git merge 和 git rebase 的区别





rebase是找到相交的地方点

`git merge --no-ff bob/main` 将Bob/main中的修改合并到当前分支，主要是`--no-ff`是什么呢 首先是ff: fast-forward:

Master : A---> B --->

There are two commits on it, A and B There is another feature branch, from B, it has the following commit History

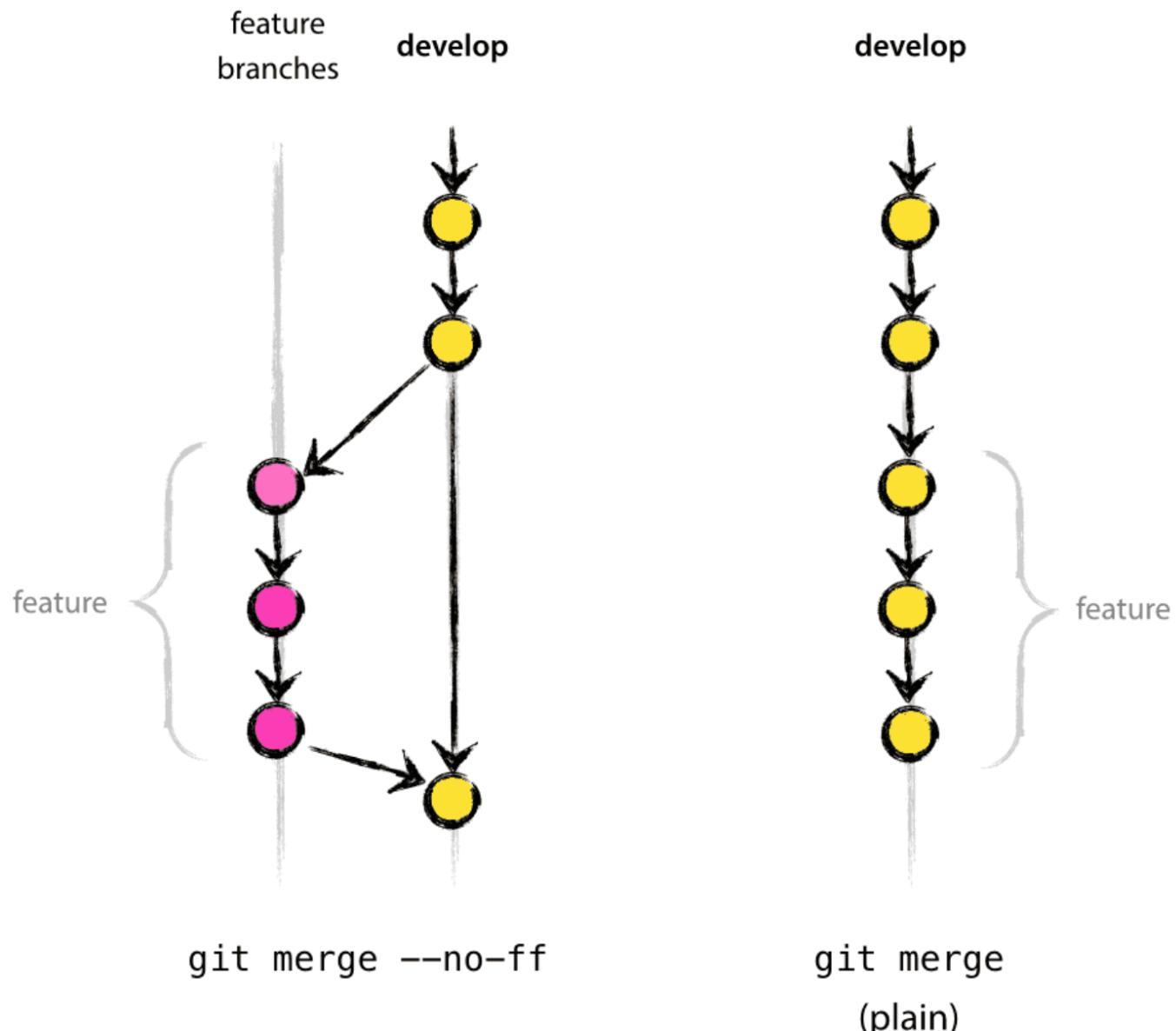
Feature : C--->D---->E

So the present it looks like this

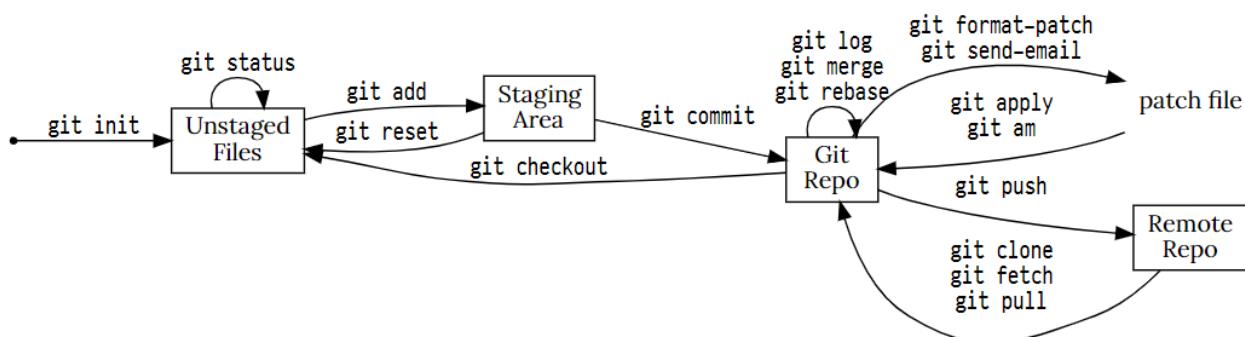
```
'A--->B'
'C--->D--->E'
```

So when you want to merge it, you can do it by fast forward merge by moving the head from B to E and incorporating the feature branch in the main branch and it looks like this

A--->B--->C--->D--->E with the header at E.



一般喜欢使用no-fastforward的意思就是可以保留branch上的分支记录



## Exercise

老三套 `git init` 初始化 `git status` 检查状态 创建一个 `file.gitignore` 将想忽略的扔进去

```
Liang@Ada MINGW64 /e/EDUCATION/硕士-布里斯托大学/Term 2/Tools/project1 (master)
$ ls
program.exe* temp.c

Liang@Ada MINGW64 /e/EDUCATION/硕士-布里斯托大学/Term 2/Tools/project1 (master)
$ git status
on branch master

No commits yet

Changes to be committed:
 (use "git rm --cached <file>..." to unstage)
 new file: temp.c

Untracked files:
 (use "git add <file>..." to include in what will be committed)
 .gitignore
```

`git add .` 将所有变动的地方加到 master branch 里 `git commit -m "Initial commit"` 添加 comments `git log` 检查状态

```
$ git commit -m "Initial commit"
[master (root-commit) 109cc27] Initial commit
 2 files changed, 8 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 temp.c
```

```
Liang@Ada MINGW64 /e/EDUCATION/硕士-布里斯托大学/Term 2/Tools/project1 (master)
$ git log
commit 109cc27daa257969130095364a5bbf81433eb8ec (HEAD -> master)
Author: Ada <liangjingtimes@gmail.com>
Date: Fri Feb 2 14:25:30 2024 +0000
```

Initial commit

`git log|less` 记下退回状态的的 hash 数字 `git checkout HASH`

```
Liang@Ada MINGW64 /e/EDUCATION/硕士-布里斯托大学/Term 2/Tools/project1 ((109cc27...))
$ git checkout 109cc27daa257969130095364a5bbf81433eb8ec
HEAD is now at 109cc27 Initial commit
```

确实退过去了(??!)这就是版本控制 `git checkout master` to return to the latest version of your files, and git

will set up the HEAD pointer again ready to accept new commits **Part 2: Git forges** ssh `git clone` SSH

**Practice the push workflow** `rebase merge` 会有 merge request 什么的 **Resolve a fake conflict, part one**

## Fake conflicts, part two

### Resolving a real conflict

# Feature branches

Every team member now independently tries the following:

- Make a new branch with `git checkout -b NAME`, choosing a unique name for their feature branch.
- Make a few commits on this branch.
- Push your feature branch with `git push --set-upstream origin NAME`.
- Make a few more commits.
- Do a simple `git push` since you've already linked your branch to `origin`.

实际上 `git checkout -b NAME` 执行了两个操作，创建branch并且切换到对应的地方 更好记的话应该是 `git`

## branch

### 1. 创建分支：

```
bash
```

```
git branch branch_name
```

上述命令会在当前提交的基础上创建一个名为 `branch\_name` 的新分支。

### 2. 列出本地分支：

```
bash
```

```
git branch
```

该命令会列出所有本地分支，当前所在的分支前会有一个 `\*` 符号。

### 3. 切换分支：

```
bash
```

```
git checkout branch_name
```

同时，`git push -u origin NAME` 将 branch 和远程联系起来，但是更好记（之后就可以美美地使用 `git push` 和 `git pull` 了）。

## Week 3: Shell Scripting & Build Tools

---

### 4. Shell Scripting

#### 4.1 File permissions

## Systems have users!

- ▶ The UNIX DAC lets you set file permissions!
- ▶ `setuid` and `setgid` programs exist!
- ▶ Root's firstname is *Charlie*!

`chmod` to change permissions

`chown` to change file owners

一开始是root:

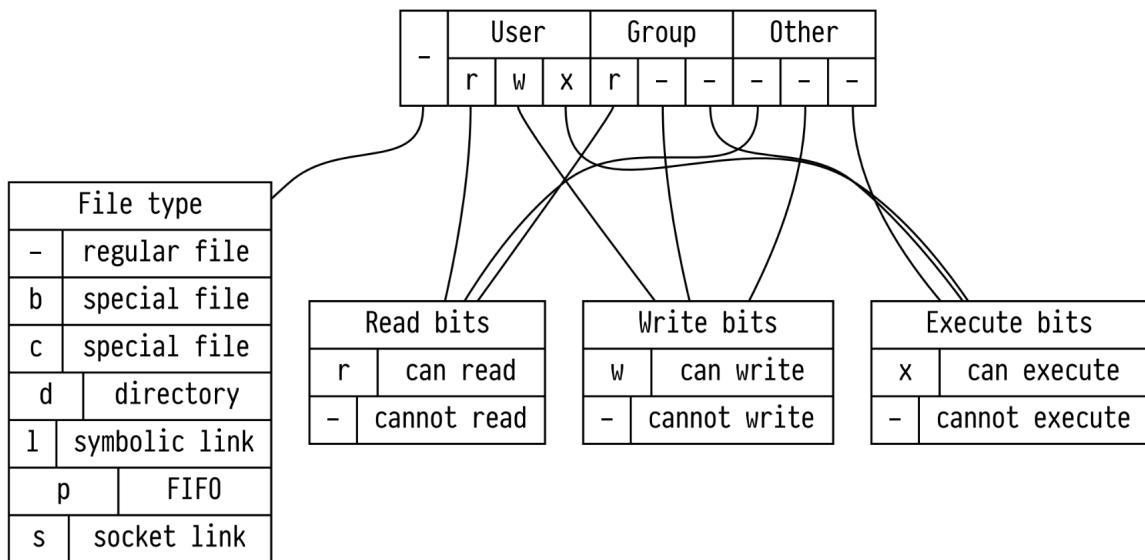
And it was good.

The root user is all powerful.

- ▶ The *super user*
- ▶ The *system administrator*
- ▶ UID 0

大概讲了下group、owner和others的东西，还有密码储存的位置

And the owner of each file could set the permissions for each file



文件权限是操作系统用来控制对文件（和目录）访问的一种机制。在类Unix系统中，文件权限通常由三组权限表示，分别是用户（owner）、组（group）和其他人（others）的权限。每组权限包括读取（read，简写为 "r"）、写入（write，简写为 "w"）和执行（execute，简写为 "x"）权限。

每个文件都有一个所有者和一个关联的组。以下是文件权限的常见表示方式：

- `r` 表示读权限，允许查看文件内容和文件属性。
- `w` 表示写权限，允许修改文件内容或删除文件。
- `x` 表示执行权限，对于目录来说，允许进入目录；对于文件来说，允许执行。

文件权限可以通过如下格式的字符来表示：

- `r` 表示读权限，用数字表示为 4。
- `w` 表示写权限，用数字表示为 2。
- `x` 表示执行权限，用数字表示为 1。
- `-` 表示没有相应的权限。

例如，`rw-r--r--` 表示文件所有者有读写权限，而组和其他人只有读权限。

在命令行中，可以使用 `chmod` 命令来更改文件权限，例如：

```
bash ⌂ Copy code
chmod 755 filename
```

### 1. Owner (所有者) :

- 所有者是文件或目录的创建者或拥有者。
- 所有者通常是创建文件的用户，可以是具有管理员权限的用户或其他普通用户。
- 所有者对文件有特定的权限，这些权限控制着他们对文件的读、写和执行等操作。

### 2. Group (组) :

- 每个文件都关联一个组，这个组通常是文件的创建者所属的用户组。
- 除了所有者之外的一些用户可能属于相同的组。
- 组权限控制着组内成员对文件的访问权限，通常这些用户是共享相似权限的。

### 3. Others (其他人) :

- "Others" 表示文件的其他用户，既不是所有者也不属于与文件相关联的组的用户。
- 这是系统中除了文件所有者和组成员之外的所有其他用户。

这三个实体 (Owner、Group、Others) 共同构成了文件权限模型。每个实体都有一组权限，例如读取、写入和执行。这种模型允许细粒度地控制不同用户对文件的访问，以保护文件的安全性和隐私。

数字表示法使用三个八进制数字来表示文件的权限。每个数字代表一个权限组（所有者、组、其他人）。每个数字由三个比特组成，分别对应读（4）、写（2）和执行（1）权限。例如：

- `4` 表示读权限。
- `2` 表示写权限。
- `1` 表示执行权限。

通过将这些数字相加，可以创建一个三位数的数字，表示各个权限的组合。例如，`chmod 755 file` 表示：

- 所有者有读、写、执行权限 (`4+2+1 = 7`)。
- 组有读、执行权限 (`4+1 = 5`)。
- 其他人有读、执行权限 (`4+1 = 5`)。

所以这个755是这么来的 **weird bit** 它上面提到过sticky point

### 1. Setuid (SUID) :

- **用法:** `chmod u+s file`
- **作用:** 当文件被设置为 Setuid 时，执行该文件的用户将暂时拥有文件所有者的权限。这通常用于提供一些需要特权的系统命令，例如 `passwd` 命令。

### 2. Setgid (SGID) :

- **用法:** `chmod g+s file`
- **作用:** 当文件被设置为 Setgid 时，执行该文件的用户将暂时拥有文件组的权限。通常用于共享目录，确保新创建的文件具有相同的组所有权。

### 3. Suid、Sgid、Sticky bit 用数字表示:

- Setuid: `chmod 4000 file`
- Setgid: `chmod 2000 file`
- Sticky bit: `chmod 1000 file`
- 数字表示法将对应的位设置为 1。



韩顺平：

1. 文件有三个文件模型：owners, group, others

2. 创建用户，是下面的那种

3. 创建一个文件并且修改权限：

#### ● 查看文件/目录所在组

##### ✓ 基本指令

**ls -ahl**

✓ 应用实例，使用 fox 来创建一个文件，看看该文件属于  
-rw-r--r--. 1 fox monster 0 11月 5 12:50 ok.txt

#### ● 修改文件所在的组

##### ✓ 基本指令

**chgrp 组名 文件名**

##### ✓ 应用实例

使用root用户创建文件 orange.txt ,看看当前这个文件属  
fruit组。

1. **groupadd fruit**

2. **touch orange.txt**

3. 看看当前这个文件属于哪个组 **ls -l > root组**

4. **chgrp fruit**

#### 4. 改变用户所在组

**改变用户所在组**

- 在添加用户时，可以指定将该用户添加到哪个组中，同样的用root的管理权限可以改变某个用户所在的组。
- 改变用户所在组
  - `usermod -g 组名 用户名`
  - `usermod -d 目录名 用户名` 改变该用户登陆的初始目录。

#### 5. bits(rwx)

**0-9位说明**

- 第0位确定文件类型(d, -, l, c, b)**
  - l**是链接，相当于windows的快捷方式
  - d**是目录，相当于windows的文件夹
  - c**是字符设备文件，鼠标，键盘
  - b**是块设备，比如硬盘
- 第1-3位确定所有者（该文件的所有者）拥有该文件的权限。---User**
- 第4-6位确定所属组（同用户组的）拥有该文件的权限，---Group**
- 第7-9位确定其他用户拥有该文件的权限 ---Other**

**ls -l 中显示的内容如下：**

```
-rwxrw-r-- 1 root root 1213 Feb 2 09:39 abc
```

#### 6. 修改权限

**u:所有者 g:所有组 o:其他人 a:所有人(u、g、o的总和)**

- 1) chmod u=rwx,g=rx,o=x 文件/目录名**
- 2) chmod o+w 文件/目录名**
- 3) chmod a-x 文件/目录名**

#### 案例演示

1) 给abc文件的所有者读写执行的权限，给所在组读执行权限，给其它组读执行权限。

`chmod u=rwx,g=rx,o=rx abc`

2) 给abc文件的所有者除去执行的权限，增加组写的权限

`chmod u-x,g+w abc`

3) 给abc文件的所有用户添加读的权限

`chmod a+r abc`

`chmod u=rwx,g=rx,o=rx abc` `chmod u-x,g+w abc` `chmod a+r abc` 或者用数字来 `read (4) write (2)`

`execute (1) (bit)` 所以才有什么`chmod 755 file` (`owner (4+2+1)` , `group (4+1)` , `others (4+1)`)

#### 7. 修改文件所有人-chown

**chown newowner 文件/目录 改变所有者**  
**chown newowner:newgroup 文件/目录 改变所有者和所在组**

- R 如果是目录 则使其下所有子文件或目录递归生效**

修改文件或目录所在组-chgrp

## chgrp newgroup 文件/目录 改变所有组

栗子

police , bandit

jack, jerry: 警察

xh, xq: 土匪

1. 创建组

2. 创建用户

3. jack 创建一个文件，自己可以读写，本组人可以读，其它组没人任何权限

4. jack 修改该文件，让其它组人可以读，本组人可以读写

5. xh 投靠 警察，看看是否可以读写。

sudo groupadd police sudo groupadd bandit sudo useradd jack sudo useradd jerry sudo useradd xh sudo useradd xq sudo usermod -aG police jack sudo usermod -aG police jerry 或者这样：

groupadd police

groupadd bandit

useradd -g police jack

已存在

userdel -r jack

useradd -g police jack

useradd -g police jerry

useradd -g bandit xh

useradd -g bandit xq

修改权限就是usermod

- 改变用户所在组

1. usermod -g 新组名 用户名

2. usermod -d 目录名 用户名 改变该用户登陆的初始目录。特别说明：用户需要有进入到新目录的权限。

**小总结** useradd添加用户 groupadd添加组 usermod -g groupname username用户添加到组里 usermod -d reponame username用户初始登录文件位置 passwd设置用户密码

chgrp 组名 文件名修改文件所在的组 chmod 权限 文件修改权限 chown newowner file/repo改变文件或者目录的所有者

### Exercise

直接用unix系统做，需要在lab上跑一遍 vagrant up -> vagrant ssh **Create a user and a group** sudo adduser NAME加名字

```
[vagrant@localhost ~]$ sudo adduser ADA
[vagrant@localhost ~]$ sudo passwd ADA
Changing password for user ADA.
```

New password:

BAD PASSWORD: The password is a palindrome

Retype new password:

passwd: all authentication tokens updated successfully

```
[vagrant@localhost ~]$ su ADA
```

Password:

```
[ADA@localhost vagrant]$
```

创立新的user，设置密码，切换到该user下面 `su ADA` 切换到ADA user下面 `pwd` 检查确实在user/home/下面  
`exit` 回到host 然后再次创建一个，并添加到一个组里面 `sudo addgroup USER` 添加组USER（反正addgroup和groupadd总有一个）

```
[vagrant@localhost ~]$ sudo usermod -aG USER ADA
[vagrant@localhost ~]$ sudo usermod -aG USER brian
[vagrant@localhost ~]$ getent group USER
```

将ADA和brian扔到组里

```
[vagrant@localhost ~]$ getent group USER
USER:x:1004:ADA,brian
```

是一个组里了

这样就

### Explore file permissions

Check, by using `su USERNAME` to log in as the different users, that:

- `nigel` can view Brian's home directory but not create files there;
- `nigel` can view but not edit Brian's `readme` file;
- `vagrant` cannot list files in or enter Brian's home directory at all. What happens when you try?

## 1. 通过使用 `su` 命令以不同的用户登录，检查以下情况：

- `nigel` 能够查看 `Brian` 的主目录，但不能在其中创建文件。
- `nigel` 能够查看但不能编辑 `Brian` 的 `readme.txt` 文件。
- `vagrant` 无法列出或进入 `Brian` 的主目录。尝试时会发生什么？

请注意，`vagrant` 可以使用 `sudo` 来绕过所有这些限制。权限不能保护允许成为 root 的任何人。

## 2. 以 `Brian` 用户身份，在主目录中创建一个私有子目录，只有你自己能够访问（读取、写入或执行）。使用 `nano` 命令在 `private/secret.txt` 中创建一个文件。不要更改 `secret.txt` 本身的所有权限。

以 `Nigel` 的身份检查，确保你能够看到这个文件夹本身，但不能进入它，也不能列出文件。检查即使知道文件名 (`cat /home/brian/private/secret.txt`)，`Nigel` 也无法读取它。

## 3. 使用 `ls -l` 命令，以 `Brian` 用户身份在 `~` 和 `~/private` 目录中比较以下条目的权限：

- `~/readme.txt`
- `~/private/secret.txt`
- `~/private` 目录

为什么两个文件的组不同？

请注意，尽管 `secret.txt` 默认认为所有人提供了读权限，但 `Nigel` 无法读取它。规则是你需要对从 `/` 到文件的整个路径具有访问权限才能访问它。



例如，为了确保 `private` 目录只有 `Brian` 用户可以访问，你可以执行以下操作：

```
bash
```

Copy code

```
chmod 700 ~/private
```

这将为 `private` 目录设置读取、写入和执行权限，而其他用户将没有任何权限。这确保了只有 `Brian` 用户可以访问该目录。

~~~~~  
这个做的我好烦躁 (

## 4.2 Shell Scripting

### Slides

## ● 脚本的常用执行方式

### ✓ 方式1(输入脚本的绝对路径或相对路径)

说明：首先要赋予helloworld.sh 脚本的+x权限，再执行脚本  
比如 ./hello.sh 或者使用绝对路径 /root/shcode/hello.sh

### ✓ 方式2(sh+脚本)

说明：不用赋予脚本+x权限，直接执行即可。

比如 sh hello.sh

还讲了一下环境变量的东西，但是我觉得不一定是重点，考都没考 x shellcheck vim xxx.sh ctrl+v:vision mode 光标选择+d

## 变量

Shell语言中的变量往往更像宏变量。

► 对于未定义的变量使用没有惩罚。

bash

```
NAME='Joe'
unset NAME
echo "Hello, '${NAME}'"
```

输出为："Hello,""

unset是取消赋值 其他的比如\$?是exit code，检查和输出错误信息

Shell scripts are written by chaining commands together

**A; B** run A then run B

**A | B** run A and feed its output as the input to B

**A && B** run A and if successful run B

**A || B** run A and if not successful run B

## How does it know if its successful?

Programs return a 1 byte exit value (e.g. C main ends with `return 0;`)

- ▶ This gets stored into the variable  `${?}`  after every command runs.
- ▶ `0` indicates success (usually)
- ▶ `>0` indicates failure (usually)

这个就是，`0`其实表示success，而大于`0`的是错误信息？

- `$0` - 脚本名
- `$1` 到 `$9` - 脚本的参数。 `$1` 是第一个参数，依此类推。
- `$@` - 所有参数
- `$#` - 参数个数
- `$?` - 前一个命令的返回值
- `$$` - 当前脚本的进程识别码
- `!!` - 完整的上一条命令，包括参数。常见应用：当你因为权限不足执行命令失败时，可以使用 `sudo !!` 再尝试一次。
- `$_` - 上一条命令的最后一个参数。如果你正在使用的是交互式 shell，你可以通过按下 `Esc` 之后键入 `.` 来获取这个值。

命令通常使用 `STDOUT` 来返回输出值，使用 `STDERR` 来返回错误及错误码，便于脚本以更加友好的方式报告错误。返回码或退出状态是脚本/命令之间交流执行状态的方式。返回值`0`表示正常执行，其他所有非`0`的返回值都表示有错误发生。

- **shell变量的定义**
- ✓ **基本语法**
- **1. 定义变量 : 变量名=值**
- **2. 撤销变量 : unset 变量**
- **3. 声明静态变量 : readonly 变量 , 注意 : 不能unset**

标准变量

**`\${0}`** Name of the script

**`\${1}`, `\${2}`, `\${3}`...** Arguments passed to your script

**`\${#}`** The number of arguments passed to your script

**`\${@}` and `\${\*}`** All the arguments

### ● 基本语法

**\$n** (功能描述：n为数字，\$0代表命令本身，\$1-\$9代表第一到第九个参数，十以上的参数需要用大括号包含，如\${10})

**\$\*** (功能描述：这个变量代表命令行中所有的参数，\$\*把所有的参数看成一个整体)

**\$@** (功能描述：这个变量也代表命令行中所有的参数，不过\$@把每个参数区分对待)

**\$#** (功能描述：这个变量代表命令行中所有参数的个数)

**\$\$** (功能描述：当前进程的进程号 ( PID ) )

**\$!** (功能描述：后台运行的最后一个进程的进程号 ( PID ) )

**\$?** (功能描述：最后一次执行的命令的返回状态。如果这个变量的值为0，证明上一个命令正确执行；如果这个变量的值为非0 (具体是哪个数，由命令自己来决定)，则证明上一个命令执行不正确了。)

### if 判断语句

执行语句

fi

### for 循环语句

```
for n in 1 2 3 4 5; do seq 5 seq -s, 5
 echo -n "${n} " 1 2 3 4 5 1,2,3,4,5
done
1 2 3 4 5 for n in $(seq 5); do # IFS = In Field Separator
 echo -n "${n} " IFS=' '
 done
 1 2 3 4 5 for n in $(seq -s, 5); do
 echo -n "${n} " echo -n "${n} "
 done
 1 2 3 4 5
```

```
for i in { };do
 执行语句
done
```

### case 跳转语句

```
3 # Remove everything upto the last / from ${SHELL}
case "${SHELL##*/}" in
 bash) echo "I'm using bash!" ;;
 zsh) echo "Ooh fancy a zsh user!" ;;
 fish) echo "Something's fishy!" ;;
 *) echo "Ooh something else!" ;;
esac
```

### 3) 按照文件权限进行判断

-r 有读的权限

-w 有写的权限

-x 有执行的权限

### 4) 按照文件类型进行判断

-f 文件存在并且是一个常规的文件

-e 文件存在

-d 文件存在并是一个目录

## Basename and dirname

- basename: 提取文件的名字

其中 `PATH` 是文件的路径。例如：

```
bash

$ basename /path/to/filename.txt
filename.txt
```

这个命令还可以结合使用后缀来去除文件扩展名：

```
bash

$ basename /path/to/filename.txt .txt
filename
```

去除后缀名字实际上也说了？

```
for f in *.jpg; do
 convert "${f}" "$(basename "${f}" .jpg).png"
done
```

这个的意

思就是，找到所有后缀为.jpg的文件，然后去除后缀，加上.png convert是一个库ImageMagick里面内置的东西x

- dirname: 提取目录的名字

2. `dirname`：这个命令用于提取文件路径中的目录部分。它的基本语法是：

```
bash
dirname PATH
```

其中`PATH`是文件的路径。例如：

```
bash
$ dirname /path/to/filename.txt
/path/to
```

**Extra Pipes** 给了第一个例子：`ps -A | grep -i firefox` ps是查看进程，-A是显示全部

|       |    |      |         |                                |              |           |
|-------|----|------|---------|--------------------------------|--------------|-----------|
| 43172 | ?? | SpU  | 0:10.69 | /usr/local/bin/firefox         |              |           |
| 59551 | ?? | Sp   | 0:00.06 | /usr/local/lib/firefox/firefox | -contentproc | -appDir   |
| 7023  | ?? | SpU  | 0:06.10 | /usr/local/lib/firefox/firefox | -contentproc | {a032331} |
| 59478 | ?? | SpU  | 0:00.21 | /usr/local/lib/firefox/firefox | -contentproc | {3cd651d} |
| 47320 | ?? | SpU  | 0:00.60 | /usr/local/lib/firefox/firefox | -contentproc | {50d5261} |
| 26734 | ?? | SpU  | 0:00.18 | /usr/local/lib/firefox/firefox | -contentproc | {68aa722} |
| 308   | ?? | SpU  | 0:00.16 | /usr/local/lib/firefox/firefox | -contentproc | {bd6ff5f} |
| 42479 | ?? | SpU  | 0:00.14 | /usr/local/lib/firefox/firefox | -contentproc | {d874750} |
| 45572 | ?? | Rp/2 | 0:00.00 | grep                           | -i           | firefox   |

然后如果太多了，就加上：`ps -A | grep -i firefox | awk '{print $1,$5}'`仅显示第一列和第五列

```
43172 /usr/local/bin/firefox
59551 /usr/local/lib/firefox/firefox
7023 /usr/local/lib/firefox/firefox
59478 /usr/local/lib/firefox/firefox
47320 /usr/local/lib/firefox/firefox
26734 /usr/local/lib/firefox/firefox
308 /usr/local/lib/firefox/firefox
42479 /usr/local/lib/firefox/firefox
5634 grep
```

## 2. 内置变量:

- `\${}`: 表示整个输入行。
- `\$1`, `\$2`, ...: 表示输入行的第一、第二个字段等。
- `NF`: 当前行中字段的数量。
- `NR`: 当前记录(行)数。

## 3. 字段分隔符:

- 使用`-F`选项指定字段分隔符，默认为空格。

```
bash
Copy code
awk -F'，' '{ print $1 }' data.csv
```

## 4. 条件和比较:

```
bash
Copy code
awk '$3 > 50 { print $2 }' data.txt
```

- 上述示例表示，如果第三列的值大于50，则打印第二列的值。

ps -A | grep -i firefox | awk '{print \$1,\$5}' | ghead -n -1 去掉最后一个的grep进程 ( ps -A | grep -i firefox | awk '{print \$1, \$5}' | ghead -n -1 | wc -l 只是想知道进程数量

- ▶ The `|` pipe copies standard output to standard input...
- ▶ The `>` pipe copies standard output to a named file... (e.g. `ps -A >processes.txt`, see also the `tee` command)
- ▶ The `>>` pipe appends standard output to a named file...
- ▶ The `<` pipe reads a file into standard input... (e.g. `grep firefox <processes.txt`)
- ▶ The `<<<` pipe takes a string and places it on standard input
- ▶ You can even copy and merge streams if you know their file descriptors (e.g. appending `2>&1` to a command will run it with standard error merged into standard output)

又对pipe进行一个总结 | standard output used as another ones' standard input, 标准输出作为另一个指令的标准输入 >standard output stored in a file将标准输出输出到一个文件中 >>append the standard output to a file but not overwritten the previous content将标准输出添加到一个文件的后面，但是不覆盖文件本来的内容 <read a file used as standard input读一个文件并将其作为标准输入 <<<将一个字符串传递给标准输入

```
bash
```

```
echo "Hello, World!" | grep "Hello"
```

上述命令使用 `echo` 将字符串 "Hello, World!" 输出，并通过管道 | 将其作为标准输入，然后通过 `grep` 命令以查找包含 "Hello" 的行。

使用 `<<<` 可以实现相同的效果：

```
bash
```

```
grep "Hello" <<< "Hello, World!"
```

## Exercise

<https://missing-semester-cn.github.io/2020/shell-tools/> 参考一下这个

2. 编写两个bash函数 `marco` 和 `polo` 执行下面的操作。每当你执行 `marco` 时，当前的工作目录应当以某种形式保存，当执行 `polo` 时，无论现在处在什么目录下，都应当 `cd` 回到当时执行 `marco` 的目录。为了方便 debug，你可以把代码写在单独的文件 `marco.sh` 中，并通过 `source marco.sh` 命令，（重新）加载函数。
3. 假设您有一个命令，它很少出错。因此为了在出错时能够对其进行调试，需要花费大量的时间重现错误并捕获输出。编写一段bash脚本，运行如下的脚本直到它出错，将它的标准输出和标准错误流记录到文件，并在最后输出所有内容。加分项：报告脚本在失败前共运行了多少次。

```
#!/usr/bin/env bash

n=$((RANDOM % 100))

if [[n -eq 42]]; then
 echo "Something went wrong"
 >&2 echo "The error was using magic numbers"
 exit 1
fi

echo "Everything went according to plan"
```

## Compile helper exercise

Write a shell script in a file called `b` (for build) that does the following:

- Your script should run under any Bourne-compatible shell (e.g. not just `bash`), and it should be written so that you can call it with `./b`.
- `./b compile NAME` should compile the file of the given name, so for example `./b compile hello` should run `gcc -Wall -std=c11 -g hello.c -o hello`.
- However, your script should accept both `./b compile hello` and `./b compile hello.c` as input, and do the same thing in both cases, namely compile `hello.c`. The output file for gcc in both cases should be called just `hello`.
- If the source file you provided as an argument does not exist (adding `.c` if necessary) then the script should print an error message and return a nonzero exit status - *not* invoke the C compiler.
- `./b run NAME` should run the program, assuming it exists in the current folder, so both `./b run hello` and `./b run hello.c` should run `./hello`. If it does not exist, again print an error message and exit with a nonzero status, don't try and run the program.
- `./b build NAME` should first compile the C source file, and then if the compile was successful it should run the program. If the compile failed, it should not try and run the program.
- If you call `./b` without any parameters, or `./b COMMAND` with a command other than compile or run or build, it should print some information on how to use it. If you call `./b compile` or another command with no filename at all, then the script should print an error message and exit with a nonzero exit status.

You now have a useful tool for when you are developing C programs. Of course you can add other features of your own like a `debug` command that compiles the file and launches it in `gdb`.

写一个脚本called b 1.在各种shell里都通用的 2.compile NAME 一个C文件，同时接受hello.c 和hello 3.如果该文件不存在，则报错 4.run NAME 5.build NAME 6.COMMAND 或者./b 打印手册

```
#!/bin/sh

if [$# -eq 0]; then
 echo "Usage: $0 COMMAND [NAME]"
 echo "Commands: compile, run, build"
 exit 1
fi

command="$1"
name="$2"

case "$command" in
compile)
 if [-z "$name"]; then
 echo "Error: Missing filename for compile command."
 exit 1
 fi

 if [! -f "$name"]; then
 if [-f "$name.c"]; then
 name="$name.c"
 else
 echo "Error: File $name does not exist." > /dev/null
 exit 1
 fi
 fi

 gcc -Wall -std=c11 -g $name -o $name
 if [$? -ne 0]; then
 echo "Error: Compilation failed." > /dev/null
 exit 1
 fi
 ;;
run)
 if [-f "$name"]; then
 ./"$name"
 else
 echo "Error: File $name does not exist." > /dev/null
 exit 1
 fi
 ;;
build)
 if [-f "$name"]; then
 ./"$name"
 else
 echo "Error: File $name does not exist." > /dev/null
 exit 1
 fi
 ;;
*)
 echo "Usage: $0 COMMAND [NAME]"
 echo "Commands: compile, run, build"
 exit 1
esac
```

```

 else
 echo "Error: Source file '$name' not found."
 exit 1
 fi
 fi

 gcc -Wall -std=c11 -g "$name" -o "${name%.*}"
;;
run)
if [-z "$name"]; then
 echo "Error: Missing program name for run command."
 exit 1
fi

if [! -x "$name"]; then
 if [-x "$name.c"]; then
 name="$name.c"
 else
 echo "Error: Program '$name' not found."
 exit 1
 fi
fi

./"$name"
;;
build)
if [-z "$name"]; then
 echo "Error: Missing filename for build command."
 exit 1
fi

./b compile "$name"
if [$? -eq 0]; then
 ./b run "${name%.*}"
fi
;;
*)
echo "Usage: $0 COMMAND [NAME]"
echo "Commands: compile, run, build"
exit 1
;;
esac

exit 0

```

这个主要看cheat sheet 写也可以? ? ? **Strict Mode** 这里提到了一个叫Strict Mode的东西 就是类似C里面那个所有warning都当error处理 `set -euo pipefail`就是shell scripting的严格模式 下面就是给你大概解释一下每个指令都是干什么的 对了shell scripting就是 return 0代表正确执行 (`set -e`强制退出, 如果指令 (return non-zero) `set -u`任何没有未被定义的variable就当是错误的 `set -o pipefail`如果pipeline中command错误 (return non-zero) 直接跳到exit code

## 5. Building Tools

这个我看都是在教安装（考过一道题，makefile的，就是因为用过（比如说如果make之后没有显示出任何东西，就说明没有新更新的内容就算新更新的东西，也只会显示更新的那个东西里面（？？？

dialects: BSD Make: 更老式，POSIX (什么是POSIX? 一个操作系统接口标准，目的是使不同UNIX系统之间的软件可移植性更强。它定义了一组标准接口和命令，以确保应用程序在符合POSIX标准的不同UNIX和UNIX-like操作系统上能够一致运行) GNU Make: default on Linux, 一般来说用的是GNU Make

这里也提到过make的使用：

```
hello: hello.c library.o
 cc -o hello hello.c library.o
```

```
library.o: library.c
 cc -c -o library.o library.c
```

```
coursework.zip: coursework
 zip -r coursework.zip coursework
```

1. Makefiles 会递归调用

```
$ make hello
cc -c -o library.o library.c
cc -o hello hello.c library.o
```

```
$ make hello
cc -o hello hello.c library.o
```

2. Making Changes 如果只改变了hello.c

只会改变含有hello.c的

3. Phony Targets (虚拟目标)

```
make
.PHONY: clean

clean:
 rm -f *.o
```

在上面的例子中，`clean` 被声明为虚拟目标，它并不代表实际的文件。当你运行 `make clean` 时，它会执行 `rm -f \*.o` 命令，删除所有 `\*.o` 文件。

这样做的好处是，即使存在一个同名的文件，`make` 也不会因为文件已存在而认为 `clean` 目标不需要执行。声明为 `.PHONY` 的目标会被 `make` 强制执行，无论文件是否存在。

Often a `Makefile` will include a phony:

`all` typically first rule in a file (or marked `.default`): depends on everything you'd like to build

`clean` deletes all generated files

`install` installs the program

```
$ make
cc -c -o library.o library.c
cc -o hello.o hello.c library.o
zip -r coursework.zip coursework
dot -Tpdf flowchart.dot -O flowchart.pdf
```

```
.PHONY: all clean
all: hello coursework.zip flowchart.pdf
clean:
 git clean -dfx
hello: hello.c library.o
 cc -o hello hello.c library.o
library.o: library.c
 cc -c -o library.o library.c
coursework.zip: coursework
 zip -r coursework.zip coursework
flowchart.pdf: flowchart.dot
 dot -Tpdf flowchart.dot -O flowchart.pdf
```

意思就是被认为是phony的目标会被强制执行？ 声明的目标不受文件系统中是否存在同名文件的影响，它们将被强制执行

`CC=clang`

`CFLAGS=-Wall -O3`

`.PHONY: all clean`

`all: hello coursework.zip flowchart.pdf`

`clean:`

`git clean -dfx`

`hello: hello.c library.o extra-library.o`

`% .o: %.c`

`$(CC) $(CFLAGS) -c -o $@ $<`

`%: %.c`

`$(CC) $(CFLAGS) -o $@ $<`

`% .zip: %`

`zip -r $@ $<`

`% .pdf: %.dot`

`dot -Tpdf $< -O $@`

#### 4. Pattern Rules

前面规定了C的编译器，

和flag 然后是虚拟目标强制执行all clean all、 clean、 hello 然后下面的意思是：

## 6. 模式规则:

makefile

Copy code

```
% .o: %.c
 $(CC) $(CFLAGS) -c -o $@ $<
%: %.c
 $(CC) $(CFLAGS) -o $@ $<
```

- 第一个规则用于构建 `.`o` 文件，第二个规则用于构建可执行文件。这两个规则使用了模式规则，可以匹配任何符合 `.`o` 或 `.`c` 后缀的文件。

## 7. %.zip 和 %.pdf 目标:

makefile

Copy code

```
% .zip: %
 zip -r $@ $<
%.pdf: %.dot
 dot -Tpdf $< -O $@
```

- 这两个规则分别用于构建 `.`zip` 和 `.`pdf` 文件。它们也使用了模式规则，可以匹配任何符合 `.`zip` 和 `.`pdf` 后缀的文件。

总体来说，这个的意思是 `\$@` 表示目标文件，`\$<` 表示第一个依赖文件。这个规则用于链接每个 .c 文件生成可执行文件 但是C本来就有：

.PHONY: all clean

```
all: hello coursework.zip flowchart.pdf
clean:
 git clean -dfx
```

```
hello: hello.c library.o extra-library.o
```

```
% .zip: %
 zip -r $@ $<
```

```
%.pdf: %.dot
 dot -Tpdf $< -O $@
```

所以直接这样就可以了

Suppose we wanted to add more figures... we could add dependencies on `all` to build them or...

```
.PHONY: all clean
figures=$(patsubst .dot,.pdf,$(wildcard *.dot))

all: hello coursework.zip ${figures}
clean:
 git clean -dfx

hello: hello.c library.o extra-library.o

%.zip: %
 zip -r $@ $<

%.pdf: %.dot
 dot -Tpdf $< -O $@
```

甚至可以直接这样

**Commonlisp** ASDF and Quicklisp

**Go** Gobuild

**Haskell** Cabal

**Java** Ant, Maven, Gradle...

**JavaScript** NPM

**Perl** CPAN

**Python** Distutils and `requirements.txt`

**R** CRAN

**Ruby** Gem

**Rust** Cargo

**LATEX** CTAN and TeXlive

**automate building**

举的例子是Maven:

```
mvn package
[INFO] Scanning for projects\ldots[]
[INFO]
[INFO] -----< uk.ac.bristol.cs:hello >-----
[INFO] Building hello 1.0-SNAPSHOT
[INFO] -----[jar]-----
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ hello ---
[WARNING] Using platform encoding (US-ASCII actually) to copy filtered resources, i.e. build is
[INFO] skip non existing resourceDirectory /tmp/src/hello/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ hello ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding US-ASCII, i.e. build is platf
[INFO] Compiling 1 source file to /tmp/src/hello/target/classes
[INFO]
[INFO] -----
[ERROR] COMPILATION ERROR :
[INFO] -----
[ERROR] Source option 5 is no longer supported. Use 7 or later.
[ERROR] Target option 5 is no longer supported. Use 7 or later.
[INFO] 2 errors
```

### 1. **Source option 5 is no longer supported. Use 7 or later.**

- 源代码选项 5 不再受支持，建议使用版本 7 或更高版本。
- 这表示你的 Maven 项目中使用的 Java 编译器版本太低，需要升级到 Java 7 或更高版本。

### 2. **Target option 5 is no longer supported. Use 7 or later.**

- 目标选项 5 不再受支持，建议使用版本 7 或更高版本。
- 这也是与 Java 编译器版本相关的错误，建议将目标版本升级到 Java 7 或更高版本。

所以修改的方案是：

解决这个问题的一种方法是在 Maven 项目的配置文件（通常是 `pom.xml` 文件）中设置合适的 Java 版本。你可以在 `` 部分添加如下配置：

```
xml
Copy code

<build>
 <plugins>
 <plugin>
 <groupId>org.apache.maven.plugins</groupId>
 <artifactId>maven-compiler-plugin</artifactId>
 <version>3.8.1</version> <!-- 使用适当的版本号 -->
 <configuration>
 <source>7</source>
 <target>7</target>
 </configuration>
 </plugin>
 </plugins>
</build>
```

这里，`<source>` 和 `<target>` 的值都设置为 7，你也可以根据需要设置为更高的 Java 版本。确保 Maven 编译器插件的版本号适用于你的项目。在上述配置中，版本号是 `3.8.1`，你可以根据需要更改为最新的版本。

(It's either that or installing an ancient Java compiler...)

```
ed /tmp/src/hello/pom.xml <<EOF
10i
<properties>
 <maven.compiler.source>17</maven.compiler.source>
 <maven.compiler.target>17</maven.compiler.target>
</properties>
.
wq
EOF
```

639 778

1. `Could not find artifact...`
  - **原因:** Maven 无法找到所需的依赖项。
  - **解决方法:** 检查你的项目依赖关系，确保所需的库和版本在 Maven 存储库中可用。如果是网络问题，检查网络连接。
2. `[ERROR] Failed to execute goal org.apache.maven.plugins:maven-compiler-plugin...`
  - **原因:** 编译插件执行失败。
  - **解决方法:** 检查源代码是否符合 Java 语法，检查编译插件配置。
3. `[ERROR] No compiler is provided in this environment`
  - **原因:** Maven 缺少合适的编译器。
  - **解决方法:** 安装或配置 Java 编译器，确保 `JAVA\_HOME` 环境变量正确设置。
4. `[ERROR] Unknown packaging`
  - **原因:** Maven 无法识别项目的打包类型。
  - **解决方法:** 确保在项目的 `` 元素中指定了正确的打包类型，例如 `jar`、`war`。
5. `[ERROR] Plugin org.apache.maven.plugins:maven-compiler-plugin... not found`
  - **原因:** Maven 无法找到指定的插件。
  - **解决方法:** 检查插件的坐标和版本是否正确，确保 Maven 仓库中有该插件。
6. `[ERROR] Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin...`
  - **原因:** 单元测试失败。
  - **解决方法:** 检查测试代码，确保测试用例正确，或者调整测试插件配置。
7. `[ERROR] Connection timed out`
  - **原因:** Maven 无法连接到 Maven 仓库，可能是网络问题。
  - **解决方法:** 检查网络连接，确保 Maven 仓库可访问，或者更换镜像源。
8. `[ERROR] Could not resolve dependencies...`
  - **原因:** Maven 无法解决项目依赖。
  - **解决方法:** 检查项目的依赖关系，确保版本和坐标正确。

`mvn test` run the test suite

`mvn install` install the JAR into your local JAR packages

`mvn clean` delete everything

And if I'm being a bit snarky...

<https://gradle.org> A better Java build tool

# Week 4: DEBUGGING

主要是GDB，这里直接看PPT来着 **gdb**

1. compile 的时候就要加入-g

```
gcc -g -o your_program your_program.c
```

```
gdb ./your_program
```

2. gdb, 启动！

3. 在 GDB 中运行程序:

- 在 GDB 提示符中输入 `run` 命令，后面可以加入程序需要的命令行参数。

```
bash
```

Copy code

```
run arg1 arg2
```

4. 设置断点:

- 在代码中设置断点，使程序在执行到断点时停下。

```
bash
```

Copy code

```
break your_function # 在函数 your_function 处设置断点
```

5. 运行程序:

- 使用 `continue` 命令继续执行程序，直到遇到断点或程序结束。

```
bash
```

Copy code

```
continue
```

6. 查看变量的值:

- 在程序执行到断点时，可以使用 `print` 命令查看变量的值。

```
bash
```

Copy code

```
print your_variable
```



- 3.

**strace** 跟踪程序系统调用的工具，它在Linux和类Unix系统上广泛使用。它的主要作用是捕获和记录一个进程的系统调用以及接收和发送的信号

```
$ strace ./journal2 <<<'Hello'
execve("./journal2", ["./journal2"], 0x7ffd3ef71360 /* 36 vars */) = 0
brk(NULL) = 0x154f000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffc01b1b610) = -1 EINVAL (Invalid argument)
access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=74509, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 74509, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ff1ca8ef000
close(3) = 0
```

考过这个execve call的是 `./journal2 ltrace` 检查library的？就是查看用了哪些library，然后那些function被哪个library用了，是dynamic library track

```
$ ltrace ./journal3 documents/log.txt <<<hello
fopen("documents/log.txt", "a+")
printf("Type your log: ")
getline(0x7fffebcc0fc8, 0x7fffebcc0fc0, 0x7f4bfccf40aa0, 0)
time(nil)
localtime(0x7fffebcc0f38)
strftime("20", 256, "%C", 0x7f4bfccf47640)
fprintf(nil, "%s: %s\n", "20", "hello\n" <no return ...>
--- SIGSEGV (Segmentation fault) ---
+++ killed by SIGSEGV +++
```

**valgrind** 这个主要是内存相关的，检查内存是否泄露吧。

## Exercise

来人，直接上卷子 past paper1

**Q14.** Bob is trying to run a command but it isn't working. To debug their issue they run their command with strace. What command are they trying to run?

- A. bash
  - B. cd
  - C. execve
  - D. /etc/ld.so.preload

接说的“cd”，所以是B？

这里直

**Q15.** Bob is trying to run a command but it still isn't working. Using the trace from before: what type of program does strace try and run, in the first instance?

- A. A compiled binary command
- B. A shell script
- C. A shell builtin command
- D. Impossible to say

[1 mark]

**Q16.** Bob is trying to run a command but it still isn't working. Using the trace from before: what's the problem?

- A. The command they've tried to run doesn't exist
- B. The permissions are set to prevent their command running
- C. The filesystem and its inodes are corrupt.
- D. A folder they've tried to access doesn't exist

[1 mark]

strace是一个跟踪系统调用的东西，然后能搞出那么一堆所以是shell script，选B

```
write(2, "/usr/bin/cd: line 2: cd: /home/j"..., 66)
 /home/jhl8636: No such file or directory
) = 66
read(255, "", 26) = 0
```

看这个，所以是D

### past paper2

```
execve("/usr/bin/ls", ["ls", "/hoem"], 0x7ffef4661138 /* 45 vars */) = 0
brk(NULL) = 0x1ca1000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
 0x7fa9994c8000
access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
```

A. 为什么这道题不是上面那个cd，是因为ls是一个external command，类似的还有：

1. `ls`: Lists the contents of a directory.

```
bash
```

Copy code

```
execve("/usr/bin/ls", ["ls", "/some/directory"], ...)
```

2. `cp`: Copies files or directories.

```
bash
```

Copy code

```
execve("/bin/cp", ["cp", "file1", "file2"], ...)
```

3. `rm`: Removes (deletes) files or directories.

```
bash
```

Copy code

```
execve("/bin/rm", ["rm", "file1"], ...)
```

4. `grep`: Searches for patterns in text.

```
bash
```

Copy code

```
execve("/bin/grep", ["grep", "pattern", "file"], ...)
```

5. `awk`: A text processing tool for pattern scanning and processing.

```
bash
```

Copy code

```
execve("/usr/bin/awk", ["awk", '{print $1}', "file"], ...)
```

message ChatGPT...



5. `awk`: A text processing tool for pattern scanning and processing.

```
bash
```

 Copy code

```
execve("/usr/bin/awk", ["awk", '{print $1}', "file"], ...)
```

6. `gcc`: The GNU Compiler Collection for compiling and linking programs.

```
bash
```

 Copy code

```
execve("/usr/bin/gcc", ["gcc", "source.c", "-o", "executable"], ...)
```

7. `python`: Executes Python scripts or enters the interactive Python shell.

```
bash
```

 Copy code

```
execve("/usr/bin/python", ["python", "script.py"], ...)
```

8. `java`: Executes Java programs.

```
bash
```

 Copy code

```
execve("/usr/bin/java", ["java", "MyProgram"], ...)
```

A. ls /hoem

B. ls /

C. ls

D. ls /Users/

因为是 /home 而不是 /hmoe

**Q16.** Bob is trying to run a command but it still isn't working. Using the trace above: what's the problem?

- ~~D~~
- A. They've made a typo in their file path
  - B. The program they're trying to run doesn't exist ~~X~~
  - C. The permissions are set to prevent their command running
  - D. Localisation library files are missing

[1 mark]

**Q17.** What is ltrace and what is it used for?

- ~~T~~
- A. It traces which library functions a program uses
  - B. It is an interactive debugger that provides breakpoints and stepping functionality
  - C. It traces what system calls a program uses
  - D. It is the *Linux / Tux Race* game: a modern 3D games program for the Linux enthusiast

A.ltrace的概念

## Week 5: SQL

---

### Slides

Am I being paid to store/process this data?

- ▶ Yes? Use a database.
- ▶ No? Use a spreadsheet (or a database)

Does the data need to be accessed remotely?

- ▶ Yes? Use a server-style database (MySQL/MariaDB)
- ▶ No? Use a file-style database (SQLite)

Am I just playing with data or is my data tiny (gigabytes in size)?

- ▶ Yes? Use a database or plain text data storage (i.e. CSV).

Is my data *really* big (petabytes in size)?

- ▶ Yes? Use a NoSQL database (beyond scope of this course)

Does my data contain recursive data structures (i.e. lists of lists of arbitrary length)

- ▶ Yes? Use Prolog or Datalog. (or abuse a database ;-))

### 什么时候选择什么数据库呢

1. 给钱就用数据库，没给就用表格
2. 是否需要远程储存数据？server-style database(MySQL/MariaDB) 没有的话就是SQLite
3. 数据很小 用数据库或者plain text data storage(CSV)
4. 数据特别大！可以用非SQL的database (Reddit) 那种吗
5. 是否有recursive类型的数据 Prolog，或者Datalog这类

### Primary Key & Foreign Key 主键：

- 主键的值必须是唯一的，且不能为空（即不允许NULL值）。
- 在数据库中，每个表只能有一个主键，它用来确保表中的每一行都能被唯一标识。
- 主键通常用于建立表与表之间的关系，作为其他表的外键参考。

## 外键：

- 外键创建了两个表之间的引用关系，通过外键，可以在一个表中引用另一个表中的数据。

Primary Key有两个属性：唯一&非空 **SQL basics**

## Basic

**SELECT** 选择多少列 **AS**

```
SELECT * FROM album
LIMIT 5;
```

**FROM** 从哪个 **WHERE** 一个条件语句，标准日期是"YYYY-MM-DD" **AND OR NOT BETWEEN**:**AND**多项选择 **IN**:**OR**多项选择

```
USE sql_store;

SELECT *
FROM customers
WHERE quantity IN (49,38,72)
```

**LIKE %:** 无论有多少个 **REGEXP**使用正则表达式

```
SELECT artist.name AS artist
FROM album
JOIN artist
ON album.artistid=artist.artistid
WHERE album.title LIKE '%Rock%'
LIMIT 5;
```

就是title带rock的，然后限制5个

**ORDER BY** **ORDER BY** **albums DESC**降序 **ORDER BY** **albums**升序 **LIMIT**

```
USE sql_store;

SELECT *
FROM customers
WHERE customer_id=1
ORDER BY first_name
```

## GROUP BY

```
SELECT artist.name AS artist,
COUNT(album.title) as albums
FROM album
JOIN artist
ON album.artistid=artist.artistid
WHERE album.title LIKE '%Rock%'
GROUP BY artist
LIMIT 5;
```

这里的count, 然后GROUP BY和 HAVING

**NULL** SELECT \* FROM fruit WHERE fruit IS NULL; SELECT \* FROM fruit WHERE fruit IS NOT NULL;

## Join

```
SELECT column1, column2, ...
FROM table1
JOIN table2 ON condition;
```

INNER JOIN: 如果表中有至少一个匹配, 则返回行  
 LEFT JOIN: 即使右表中没有匹配, 也从左表返回所有的行  
 RIGHT JOIN: 即使左表中没有匹配, 也从右表返回所有的行  
 FULL JOIN: 只要其中一个表中存在匹配, 则返回行

```
SELECT *
FROM fruit
FULL OUTER NATURAL JOIN recipes;
```

| Person | Fruit  | Dish          |
|--------|--------|---------------|
| Joseph | Lime   | Daiquiri      |
| Matt   | Apple  | Apple crumble |
| Partha |        |               |
|        | Banana | Banana split  |
|        | Cherry |               |

即使是NULL也给你全返回咯 **Statistic**

COUNT() AVG() SUM() SQRT()

```
SELECT AVG(stars) AS Average FROM ranking;
```

Average  
3.0

```
SELECT SUM(stars)/COUNT(fruit) AS Average
FROM ranking;
```

Average  
2

Remember computers are *awful*

- ▶ Multiply count by 1.0 to "fix"?
- ▶ Also number of stars is *ordinal* data so the *mean* shouldn't be used anyway...

A. `SELECT SUM(ranking)/COUNT(ranking) FROM soupranking;`

- 这是整数除法，如果 `SUM(ranking)` 不能被 `COUNT(ranking)` 整除，结果将是一个整数。这可能导致结果不如预期。

B. `SELECT AVG(ranking) FROM soupranking;`

- 这是使用 AVG 函数计算平均值，通常是以浮点数形式返回结果。这是一个推荐的方法。

C. `SELECT (1.0\*SUM(ranking))/COUNT(ranking) FROM soupranking;`

- 这使用了浮点数除法，确保了精确的计算。这也是一个有效的方法。

D. `SELECT SUM(ranking)/(1.0\*COUNT(ranking)) FROM soupranking;`

s 哦！对应这道题！1.0\*用了浮点数所以是对的

## 内连接

1. 跨数据库连接：

```
USE sql_store;

SELECT *
FROM order_items oi
```

```
JOIN sql_inventory.products p
 ON oi.product_id=p.product_id
```

## 2. 自链接 使用不同的别名

```
USE sql_hr;
SELECT e.employee_id,
 e.first_name,
 m.first_name AS manager
FROM employees e
JOIN employees m
 ON e.reports_to=m.employee_id
```

## 3. 多表链接

```
USE sql_store;

SELECT
 o.order_id,
 o.order_date,
 c.first_name,
 c.last_name,
 os.name AS status
FROM orders o
JOIN customers c
 ON o.customer_id=c.customer_id
JOIN order_statuses os
 ON o.status=os.order_status_id
```

注意这里的链接condintion，做了exercise才发现有哪里不对= =

还有一个natural join：就是让数据库自己检测 cross join：第一个表的所有，和第二个表的所有

```
SELECT
 c.first_name AS customers
 p.name AS product
 FROM customers c
CROSS JOIN products p
 ORDER BY c.first_name
```

## 4. Unions

```
USE sql_store;
SELECT
 order_id,
```

```

order_date,
'Active' AS status
FROM orders
WHERE order_date >'2019-01-01'
UNION
SELECT
order_id,
order_date,
'Archived' AS status
FROM orders
WHERE order_date <='2019-01-01';

```

UNION: MySQL UNION 操作符用于连接两个以上的 SELECT 语句的结果组合到一个结果集合，并去除重复的行。UNION 操作符必须由两个或多个 SELECT 语句组成，每个 SELECT 语句的列数和对应位置的数据类型必须相同

## Column

INT VARCHAR DATE CHAR PK:primary key NN:not NULL(选了就是不允许有null) AI:自动增递  
default/expression: NULL或者'0'

| Column Name  | Datatype      | PK                                  | NN                                  | UQ                       | B                        | UN                       | ZF                       | AI                                  | G                        | Default/Exp |
|--------------|---------------|-------------------------------------|-------------------------------------|--------------------------|--------------------------|--------------------------|--------------------------|-------------------------------------|--------------------------|-------------|
| order_id     | INT           | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> |             |
| customer_id  | INT           | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/> |             |
| order_date   | DATE          | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/> |             |
| status       | TINYINT       | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/> | '1'         |
| comments     | VARCHAR(2000) | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/> | NULL        |
| shipped_date | DATE          | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/> | NULL        |
| shipper_id   | SMALLINT      | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/> | NULL        |

## 1.插入单行

```

INSERT INTO customers(
first_name,
last_name,
birth_date,
address,
city,
state
)
VALUE(
'Hannah',
'Smith',
'address',
'city',
'CA')

```

是有DEFAULT这个东西的,但是如果上面不写的话,实际上是可以去掉的,如果上面是这样的话 2. 插入多行

```
INSERT INTO shippers(name)
VALUES ('shippers'),
 ('shippers2'),
 ('shippers3')
```

### 3. 插入分层行 在多张表中插入行 parent-child relationship LAST\_INSERT\_ID()

`LAST\_INSERT\_ID()` 是一个 MySQL 数据库中的函数, 用于获取最近插入操作生成的自增主键的值。在执行插入操作时, 如果表中的主键是一个自增 (AUTO\_INCREMENT) 类型的列, `LAST\_INSERT\_ID()` 函数可以返回刚刚插入的行所生成的自增主键值。

### 4. 创建表复制

```
INSERT INTO orders_archived
--archived是一个新的表
```

```
SELECT *
FROM orders
WHERE order_date < '2019-01-01'
```

### 5. 更新

```
UPDATE invoices
SET payment_total=10,
payment_date='2019-03-01'
WHERE invoice_id=1
```

### 6. 删除

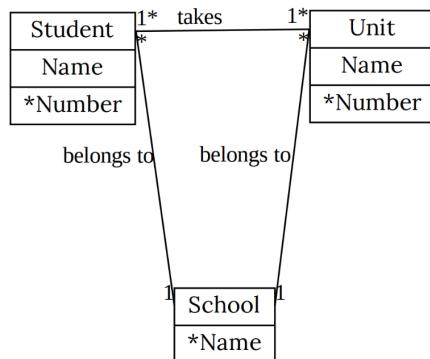
```
WHERE invoice_id =
 (SELECT *
 FROM clients
 WHERE name='Myworks')
```

到他的第五章了, 之后在慢慢看, 现在开始做school的内容

## School Tutorial

## CREATE TABLE

In the last lecture we had the following Entity relationship diagram:



```

CREATE TABLE IF NOT EXISTS student (
 name TEXT NOT NULL,
 number TEXT NOT NULL,
 PRIMARY KEY (number));

CREATE TABLE IF NOT EXISTS unit (
 name TEXT NOT NULL,
 number TEXT NOT NULL,
 PRIMARY KEY (number));

CREATE TABLE IF NOT EXISTS school (
 name TEXT NOT NULL,
 PRIMARY KEY (name));

CREATE TABLE IF NOT EXISTS class_register (
 student TEXT NOT NULL,
 unit TEXT NOT NULL,
 FOREIGN KEY (student) REFERENCES student(number),
 FOREIGN KEY (unit) REFERENCES unit(name),
 PRIMARY KEY (student, unit));

```

Lets build it in SQL

5 / 22

```

CREATE TABLE IF NOT EXISTS student(
 name TEXT NOT NULL,
 number TEXT NOT NULL,
 PRIMARY KEY(number)
);

CREATE TABLE IF NOT EXISTS unit(
 name TEXT NOT NULL,
 number TEXT NOT NULL,
 PRIMARY KEY(number)
);

CREATE TABLE IF NOT EXISTS unit(
 name TEXT NOT NULL,
 number TEXT NOT NULL,
 PRIMARY KEY(number)
);

```

sql

 Copy code

```
CREATE TABLE IF NOT EXISTS student(
 name TEXT NOT NULL,
 number VARCHAR(100) NOT NULL,
 PRIMARY KEY(number)
);
```

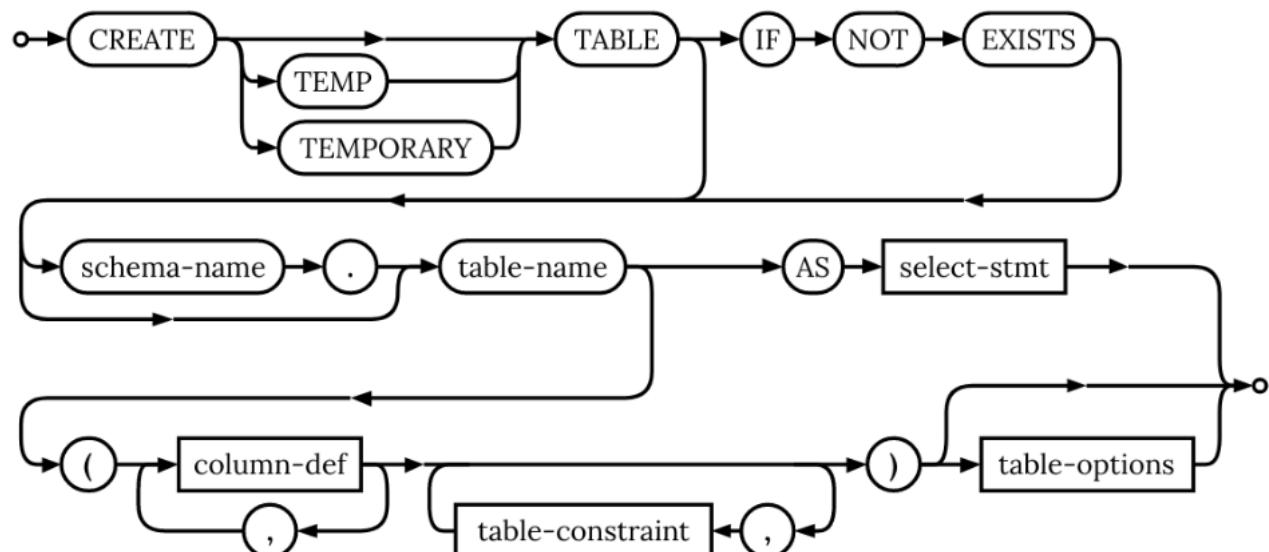


sql

 Copy code

```
CREATE TABLE IF NOT EXISTS student(
 name TEXT NOT NULL,
 number TEXT NOT NULL,
 PRIMARY KEY(number(255))
);
```

但是打下来会报错是因为number作为主键没有设置大小



## 删除表格

```
DROP TABLE IF EXISTS class_register;
```

~~DROP TABLE IF EXISTS~~

~~class\_register~~;

数据类型

**INTEGER** whole numbers

**REAL** lossy decimals

**BLOB** binary data  
(images/audio/files...)

**VARCHAR(10)** a string of 10 characters

**TEXT** any old text

**BOOLEAN** True or false

**DATE** Today

**DATETIME** Today at 2pm

但是实际上SQLite会直接

|                    |                                        |
|--------------------|----------------------------------------|
| <b>INTEGER</b>     | whole numbers                          |
| <b>REAL</b>        | lossy decimals                         |
| <b>BLOB</b>        | binary data<br>(images/audio/files...) |
| <b>VARCHAR(10)</b> | <i>actually TEXT</i>                   |
| <b>TEXT</b>        | any old text                           |
| <b>BOOLEAN</b>     | <i>actually INTEGER</i>                |
| <b>DATE</b>        | <i>actually TEXT</i>                   |
| <b>DATETIME</b>    | <i>actually TEXT</i>                   |

(others may exist... *read the manual!*)

#### Table constraints

**NOT NULL** can't be NULL

**UNIQUE** can't be the same as another row

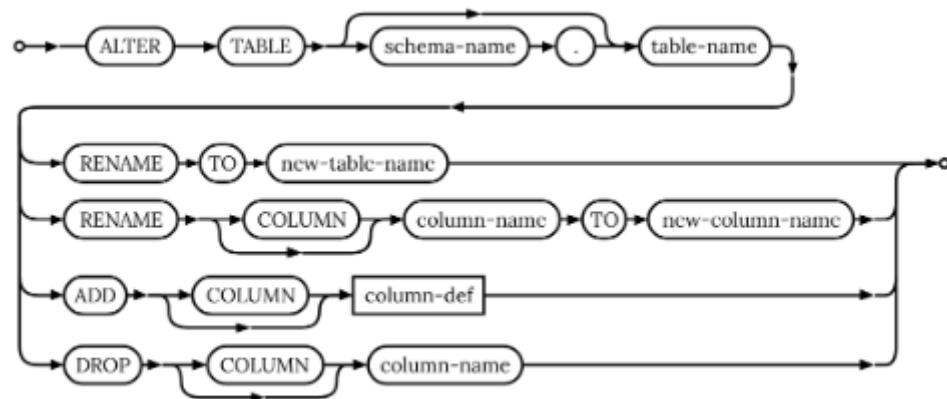
**CHECK** arbitrary checking (including it conforms to a regular expression)

**PRIMARY KEY** unique, not NULL and (potentially) autogenerated

**FOREIGN KEY (IGNORED BY MARIADB)** other key must exist

CHECK: 在创建表格的时候就对数据有一个约束

```
CREATE TABLE employee (
 id INTEGER PRIMARY KEY,
 age INTEGER CHECK (age >= 18),
 salary REAL CHECK (salary >= 0),
 name TEXT
);
```



**add constraints later**

USE <database>; 切换到该数据库 SHOW TABLE; 显示表格 DESCRIBE <TABLE>; 查看具体类型

| Field | Type         | Null | Key | Default | Extra          |
|-------|--------------|------|-----|---------|----------------|
| id    | int(11)      | NO   | PRI | NULL    | auto_increment |
| name  | varchar(100) | NO   | UNI | NULL    |                |
| party | int(11)      | YES  | MUL | NULL    |                |
| ward  | int(11)      | YES  | MUL | NULL    |                |
| votes | int(11)      | YES  |     | NULL    |                |

5 rows in set (0.001 sec)

**PRIMARY KEY: UNI KEY**唯一约束：唯一约束确保一列或一组列中的所有值都是唯一的。与主键不同，唯一约束允许包含NULL值（除非在受约束的列或列组中）

```
CREATE TABLE employees (
 employee_id INT PRIMARY KEY,
 email VARCHAR(255) UNIQUE,
 department_id INT
);
```

**MUL外键** 外键是一个表中一列或一组列，它引用另一张表的主键。它建立了两个表之间的链接，创建了一种关系。它确保引用完整性，意味着外键列中的值必须与引用表的主键匹配。

```
CREATE TABLE courses (
 course_id INT PRIMARY KEY,
 course_name VARCHAR(255)
);

CREATE TABLE enrollments (
 enrollment_id INT PRIMARY KEY,
 student_id INT,
 course_id INT,
 FOREIGN KEY (student_id) REFERENCES students(student_id),
 FOREIGN KEY (course_id) REFERENCES courses(course_id)
);
```

## Normal Forms

### 1. 第一范式 (1NF) :

- 表中的每一列都包含不可再分的原子数据（即每个单元格中的数据不可再分）。

- 确保每个单元格中的数据是原子的，没有包含集合、数组或其他非原子的数据类型。

| 学号  | 姓名 | 所选课程   |
|-----|----|--------|
| 001 | 张三 | 数学, 物理 |
| 002 | 李四 | 化学     |
| 003 | 王五 | 英语, 历史 |

因为所选课程是文本，而不是集合、数组

| 学号  | 姓名 | 年龄 | 性别 |
|-----|----|----|----|
| 001 | 张三 | 20 | 男  |
| 002 | 李四 | 22 | 女  |
| 003 | 王五 | 21 | 男  |

## 2. 第二范式 (2NF) :

- 必须符合第一范式。
- 所有非主键列完全依赖于整个主键，而不是仅依赖于主键的一部分。

- 确保表中的每一列都直接与主键相关，而不是间接与主键相关。

考虑一个"订单"表，包含订单号、产品号、顾客姓名以及产品数量的情况：

plaintext

Copy code

| 订单号 | 产品号 | 顾客姓名 | 产品数量 |
|-----|-----|------|------|
| 1   | A   | 张三   | 10   |
| 1   | B   | 张三   | 5    |
| 2   | A   | 李四   | 8    |
| 3   | C   | 王五   | 12   |

在这个表中，订单号和产品号构成了复合主键。然而，顾客姓名只与订单号相关，而不是与整个复合主键（订单号和产品号）相关。因此，这个表不满足第二范式，因为非主键列（顾客姓名）只依

| 图书编号 | 作者ID | 作者姓名 | 书名       |
|------|------|------|----------|
| 1    | 101  | 张三   | 数据库设计    |
| 2    | 102  | 李四   | Java编程   |
| 3    | 101  | 张三   | Python入门 |

这个就是符合的

### 3. 第三范式 (3NF) :

- 必须符合第二范式。
- 非主键列之间不存在传递依赖关系。换句话说，如果  $A \rightarrow B$  且  $B \rightarrow C$ ，则不能存在  $A \rightarrow C$ 。
- 确保表中的非主键列之间没有冗余的传递依赖关系。

| 订单号 | 顾客ID | 顾客姓名 | 产品ID | 产品名称 | 产品价格   |
|-----|------|------|------|------|--------|
| 1   | 101  | 张三   | A    | 商品A  | 100.00 |
| 1   | 101  | 张三   | B    | 商品B  | 50.00  |
| 2   | 102  | 李四   | A    | 商品A  | 100.00 |
| 3   | 103  | 王五   | C    | 商品C  | 75.00  |

在这个表中，订单号是主键。现在来看第三范式的要求：

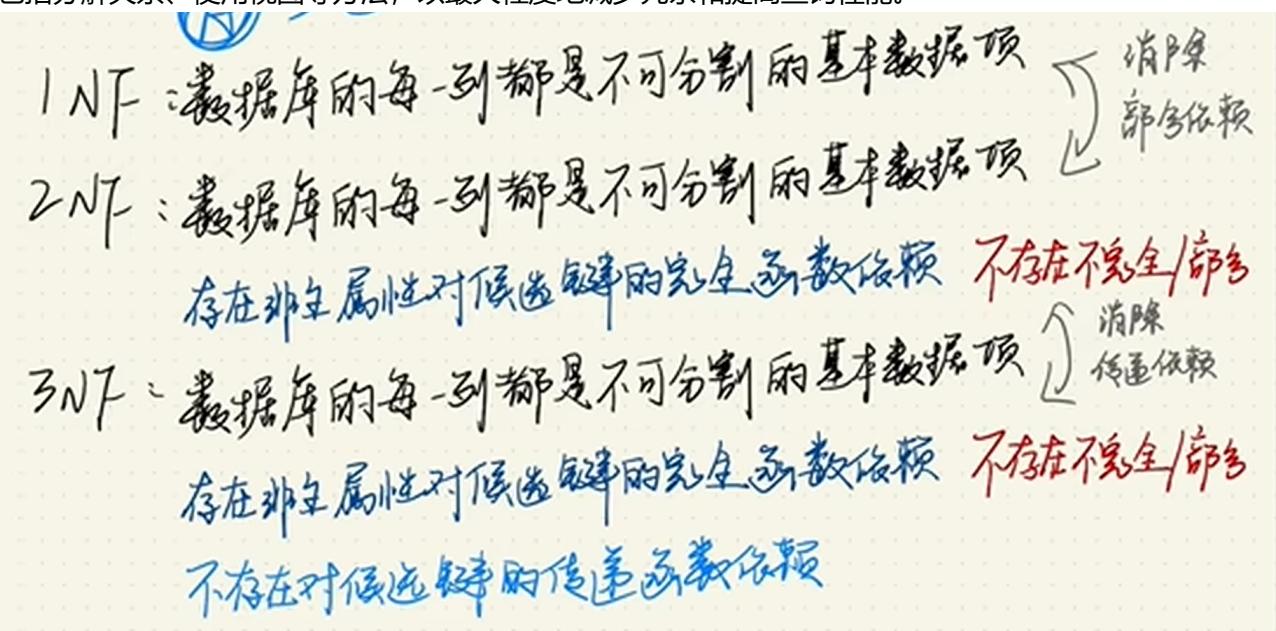
1. 表必须满足第二范式（2NF）：这个表中的每个非主键列都完全依赖于主键。顾客ID、顾客姓名、产品ID、产品名称以及产品价格都直接依赖于主键（订单号）。
2. 非主键列之间没有传递依赖：在这个表中，顾客姓名与顾客ID直接相关，而不是通过其他非主键列间接相关。同样，产品名称和产品价格也直接依赖于产品ID，而不是通过其他非主键列间接相关。

因此，这个表满足第三范式的要求。如果存在传递依赖关系，例如产品名称依赖于订单号而不是直接依赖于产品ID，就需要进一步拆分表以符合第三范式。

#### 4. 巴斯-科德范式（Boyce-Codd Normal Form, BCNF）：

- 是第三范式的一种强化形式。
- 要求表中的每一个非主键列完全依赖于主键，而不是依赖于主键的某一部分。
- 主要用于解决多候选键引起的问题，确保每一个非主键列都完全依赖于候选键。

#### 5. 第3.5范式：不是一个正式的范式，通常用于描述在某些情况下对表结构进行进一步优化的技术。可以包括分解关系、使用视图等方法，以最大程度地减少冗余和提高查询性能。



第一范式->第二范式：消除部分依赖（比如AB->C, AB推导出C，到2NF就是A也能直接推导出C） 第二范式->第三范式：消除传递依赖（传递依赖就是A->B）

## Exercise

### election

1. List the names of all parties that stood in the election, ordered alphabetically by name.

```
MariaDB [elections]> SELECT * FROM Party
 -> ORDER BY Party.name;
+---+
| id | name
+---+
| 7 | Conservative
| 4 | Green
| 3 | Independent
| 8 | Independents for Bristol
| 6 | Labour
| 2 | Liberal Democrat
| 9 | Respect
| 1 | Trade Unionists and Socialists Against Cuts
| 5 | UKIP
+---+
9 rows in set (0.000 sec)
```

2. List the names of all parties that stood in the Bedminster ward.

```
MariaDB [elections]> SELECT DISTINCT Party.name
 -> FROM Candidate
 -> JOIN Party ON Candidate.party = Party.id
 -> JOIN Ward ON Candidate.ward = Ward.id
 -> WHERE Ward.name = 'Bedminster';
+---+
| name
+---+
| Trade Unionists and Socialists Against Cuts |
| Liberal Democrat
| Conservative
| Green
| Labour
+---+
5 rows in set (0.001 sec)
```

SELECT DISTINCT 是不列重复项的意思 把party、ward对应的数字变成对应的name，然后查找 Ward.name是bedminister的

```
SELECT * FROM Candidate
JOIN Party ON Party.id=Candidate.party
JOIN Ward ON Ward.id=Candidate.ward
WHERE Ward.name='Bedminster';
```

Simplify:

```
SELECT Party.name FROM Candidate
JOIN Party ON Party.id=Candidate.party
JOIN Ward ON Ward.id=Candidate.ward
WHERE Ward.name='Bedminster';
```

1. How many votes did Labour get in the Stockwood ward?

```
MariaDB [elections]> SELECT votes
 -> FROM Candidate
 -> WHERE party = (SELECT id FROM Party WHERE name = 'Labour')
 -> AND ward = (SELECT id FROM Ward WHERE name = 'Stockwood');
+-----+
| votes |
+-----+
| 855 |
+-----+
1 row in set (0.000 sec)
```

要的是votes，所以在Candidate里面选择votes 然后两个子查询： 条件1是：Party是Labour->Candidate的都是id，所以要在Party里面找 name='Labour'，并且返回对应的id 条件2是：Ward是Stockwood->Candidate的都是id，所以要在Ward里面找 name='Stockwood'，并且返回对应的id

2. List the names, parties and number of votes obtained for all candidates in the Southville ward. Order the candidates by number of votes obtained descending (winner comes first).

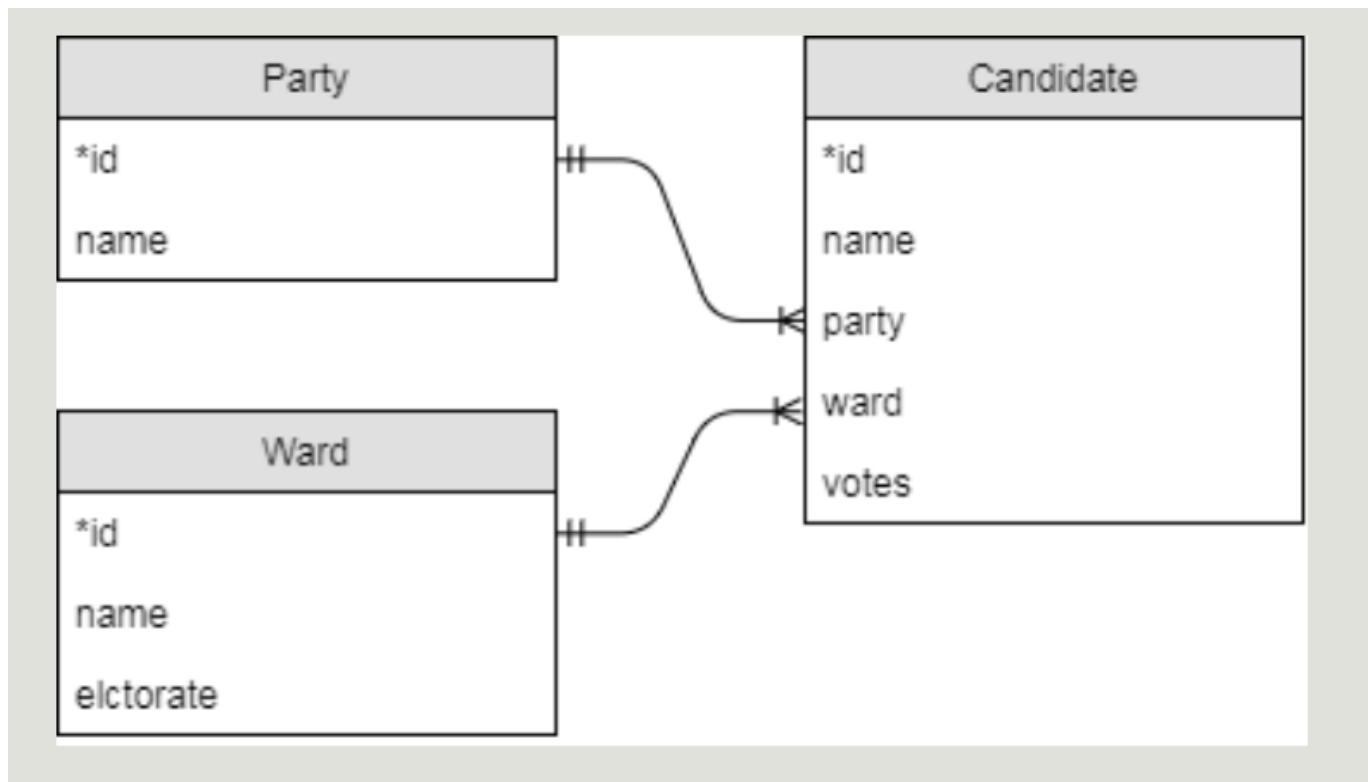
还没扔进去看

```
SELECT Candidate.name, Party.name,Candidate.votes AS name,party,votes
FROM Candidate
JOIN Ward ON Ward.id=Candidate.ward
JOIN Party ON Party.id=Candidate.party
WHERE Ward.name='Southville'
ORDER BY Candidate.votes DESC;
```

3. List the name, party and number of votes obtained for the winner only in the Knowle ward. (Hint: apart from changing the ward name, you only need one small modification to the statement from the last question. You may assume no ties.) 还没扔进去看x

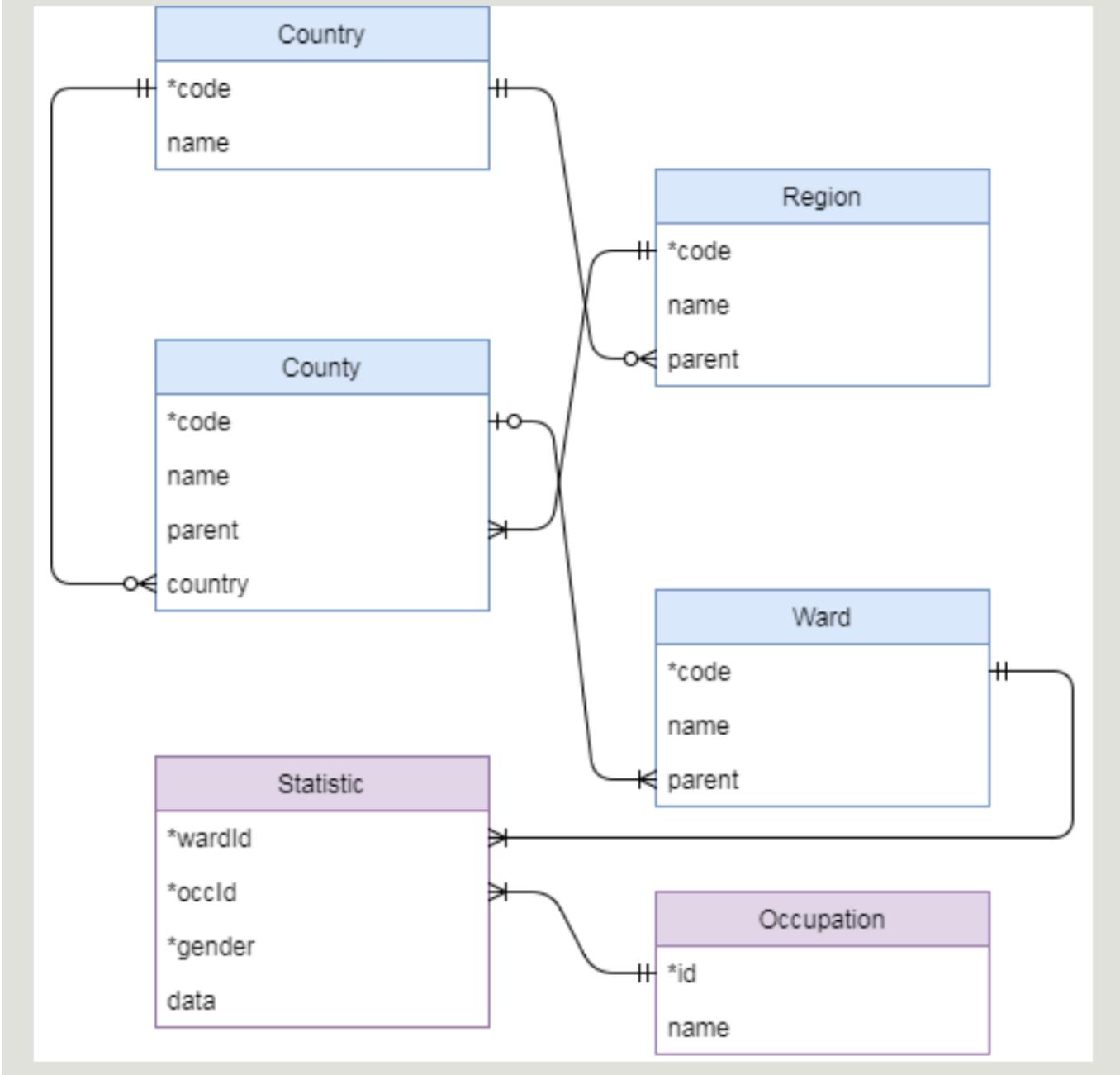
```
SELECT Candidate.name, Party.name,Candidate.votes AS name,party,votes
FROM Candidate
```

```
JOIN Ward ON Ward.id=Candidate.ward
JOIN Party ON Party.id=Candidate.party
WHERE Ward.name='Knowle'
ORDER BY Candidate.votes DESC
LIMIT 1;
```



The 2011 UK Census

The `census` database has the following schema.



1. The university of Bristol is situated in the Cabot ward (ward names are not always distinct, but this one is). Find the names and codes of the CLU, region and country containing the Cabot ward (CLU = county level unit = "row in County table").
2. If you used multiple SQL queries for the last question, do it in one single query now. (In other words, find a join strategy for the tables you need.)
3. Find the number of women in occupation class 1 (managers etc.) in the Cabot ward. You may use ward code for Cabot that you found in the first query and the occupation id 1 directly - you do not need any JOINs for this query.
4. For the Stoke Bishop ward (E05002003), list the 9 occupation class names and the number of men in each occupation. Your table should have two columns called name and number. You can use the provided ward code, you do not need to join on the ward name.

## Intermediate SQL

### Election

1. How many votes were cast in all of Bristol in the 2014 elections?
2. How many votes were cast in the 'Windmill Hill' ward and what percentage of the electorate in this ward does this represent? Your statement should produce a table with one row and two columns called 'votes' and 'percentage'.
3. List the names, parties and percentage of votes obtained for all candidates in the Southville ward. Order the candidates by percentage of votes obtained descending.
4. How successful (in % of votes cast) was the Conservative party in each ward?
5. Which rank did Labour end up in the 'Whitchurch Park' ward? Your statement should produce a table with a single row and column containing the answer as a number. You can assume no ties.
6. What is the total number of votes that each party got in the elections? Your result should be a table with two columns party, votes.
7. Find all wards where the Green party beat Labour and create a table with two columns ward, difference where the difference column is the number of Green votes minus the number of Labour votes. Your table should be ordered by difference, with the highest one first. **Census**
8. How many women work in sales and customer service occupations and live in the Cabot ward of Bristol (E05001979)?
9. How many people work in sales and customer service occupations and live in the Cabot ward of Bristol (E05001979)?
10. How many people work in caring, leisure and other service occupations (occupation class 6) in all of the City of Bristol CLU (E06000023)?
11. In the Cabot ward (E05001979), produce a table listing the names of the 9 occupation classes and the number of people in each of the classes in this ward.
12. Find the working population, ward name and CLU name for the smallest ward (by working population) in the 2011 census.
13. The same as the last question, but now produce a table with two rows, one for the smallest and one for the largest ward. There's no quicker way than repeating the last query twice, the question is how to stick the two "copies" together.
14. Find the average size of a ward's working population in the London (E12000007) region. The same as the last question but now for every region - your query should produce a table with one row per region. The intention here is not to repeat the above query 9 times.
15. Produce a table that lists, for each of the 9 regions of England, the percentage of people in managerial (class 1) occupations who are women.
16. For all CLUs in the London (E12000007) region, produce a table with three columns called CLU, occupation and count such that: CLU is the CLU name. count is the number of people of the occupation class in question in the given CLU. occupation is the name of the occupation class. Only rows with count  $\geq 10000$  appear in the table. The table is sorted by count ascending.
17. Create a table with three columns occupation, women and men and one row per occupation class. The occupation column should list the occupation class names. The women and men columns in each row should list the total number of women resp. men in the row's occupation class in the whole dataset. The intention here is not to have to copy-paste a subquery 9 times.
18. The same as question 9, but now with a 10th row in the table listing the value for all of England. You can use the string 'England' for the region column.

## JDBC

# Mid-Term

---

1. shell
2. git
- 3.

## Undergrad paper 1

### Q3

```
$ git fetch
ssh: Could not resolve hostname github.com: no address associated with name
fatal: Could not read from remote repository.
Please make sure you have the correct access rights
and the repository exists.

What is the issue?
A. Their public key has not been correctly set up on Github.
B. The repository doesn't exist; they need to create it before they can pull;
C. There is a merge conflict
D. Their network is down or misconfigured.
```

The answer is D. Maybe because it has something to do with SSH A. fatal: repository 'repository\_url' not found  
B. Permission denied (publickey). C. CONFLICT (content): Merge conflict in filename

### Q5

When running a command into their favourite pager more (so they can scroll it on the terminal) Alice notices the following unusual output. What is going on, and what are the weird ESC[01;34m bits?

```
$ ls --color=always | more
ESC[0mESC[01;34mBackupsESC[0m
ESC[01;34mDesktopESC[0m
ESC[01;34mDocumentsESC[0m
ESC[01;36mDownloadsESC[0m
ESC[01;34mMailESC[0m
ESC[01;34mMusicESC[0m
ESC[01;36mNotesESC[0m
ESC[01;34mPicturesESC[0m
ESC[01;34mReposESC[0m
ESC[01;34mVideosESC[0m
mbox
- (END)
```

I don't think that is concluded in the class.

## Q6

Bob has the following id:

```
$ id
uid=1000(bob) gid=800(users) groups=800(users)
The passwd program has the following permissions:
$ ls -l `command -v passwd`
-r-sr-sr-x 1 root bin 21112 Feb 12 00:27 /usr/bin/passwd
```

What user and group will it run as?

- A. uid=root, gid=bin
- B. uid=root, gid=users
- C. uid=bob, gid=users
- D. uid=bob, gid=bin

A.

- 用户权限中的 Setuid (s) : 以所有者的权限执行。
- 组权限中的 Setgid (s) : 以组的权限执行。
- 其他权限中的粘滞位 (s) (目录) : 限制删除到文件所有者。

The permission string -r-sr-sr-x for the file /usr/bin/passwd indicates that the setuid (set user ID) bit is set for the owner (root) and the setgid (set group ID) bit is set for the group (bin). because the user=root, group=bin so the file will be run as root and bin

## Q9

Alice has the following Makefile. Since the last build, they have made some updates to

```
main.c. Which rules will be run when next typing make?
main: main.c interface.o analysis.o
docs.html: main.c interface.c analysis.c library.c
documentation-generator main.c interface.c analysis.c library.c -o $@
interface.o: interface.c config.c
analysis.o: analysis.c library.o
library.o: library.c
A. main
B. main interface.o analysis.o
C. main docs.html
D. main interface.o analysis.o docs.html
```

because on the main.c has been changed, so the things within main.c is run too. Makefile in C

## Q10

```
Alice has created a new program in Java using the Maven buildsystem.
$ mvn test
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.mycompany.app:my-app >-----
[INFO] Building my-app 1.0-SNAPSHOT
[INFO] -----[jar]-----
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:resources (default-resources) @ my-app ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /tmp/test/my-app/my-
app/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:compile (default-compile) @ my-app ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:testResources (default-testResources) @
my-app ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /tmp/test/my-app/my-
app/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:testCompile (default-testCompile) @ my-app

[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to /tmp/test/my-app/my-app/target/test-classes
[INFO] -----
Page 7 of 29
Turn Over/Qu. continues ...(cont.)
[ERROR] COMPILATION ERROR :
[INFO] -----
[ERROR] /tmp/test/my-app/my-app/src/test/java/com/mycompany/app/AppTest.java:
[3,24]
package org.junit does not exist
[ERROR] /tmp/test/my-app/my-app/src/test/java/com/mycompany/app/AppTest.java:[3,1]
static import only from classes and interfaces
[ERROR] /tmp/test/my-app/my-app/src/test/java/com/mycompany/app/AppTest.java:
[5,17]
package org.junit does not exist
[ERROR] /tmp/test/my-app/my-app/src/test/java/com/mycompany/app/AppTest.java:
[15,6]
cannot find symbol
symbol: class Test
location: class com.mycompany.app.AppTest
[ERROR] /tmp/test/my-app/my-app/src/test/java/com/mycompany/app/AppTest.java:
[18,9]
cannot find symbol
symbol: method assertTrue(boolean)
location: class com.mycompany.app.AppTest
[INFO] 5 errors
[INFO] -----
[INFO] -----
[INFO] BUILD FAILURE
```

```
[INFO] -----
[INFO] Total time: 0.917 s
[INFO] Finished at: 2023-02-14T15:42:26Z
[INFO] -----
[ERROR] Failed to execute goal
org.apache.maven.plugins:maven-compiler-plugin:3.8.0:testCompile
(default-testCompile) on project my-app: Compilation failure: Compilation
failure:
[ERROR] /tmp/test/my-app/my-app/src/test/java/com/mycompany/app/AppTest.java:
[3,24]
package org.junit does not exist
[ERROR] /tmp/test/my-app/my-app/src/test/java/com/mycompany/app/AppTest.java:[3,1]
static import only from classes and interfaces
[ERROR] /tmp/test/my-app/my-app/src/test/java/com/mycompany/app/AppTest.java:
[5,17]
package org.junit does not exist
[ERROR] /tmp/test/my-app/my-app/src/test/java/com/mycompany/app/AppTest.java:
[15,6]
cannot find symbol
[ERROR] symbol: class Test
[ERROR] location: class com.mycompany.app.AppTest
[ERROR] /tmp/test/my-app/my-app/src/test/java/com/mycompany/app/AppTest.java:
[18,9]
cannot find symbol
[ERROR] symbol: method assertTrue(boolean)
[ERROR] location: class com.mycompany.app.AppTest
[ERROR] -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e
switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read
the following articles:
[ERROR] [Help 1]
What has gone wrong?
A. The Java compiler version is incorrect
B. The pom.xml is syntactically incorrect
C. The syntax of the mvn command is wrong
D. A dependency is missing
```

- A. it did not said error about the version B. the errors are pointing to

because the error messages specifically point to missing packages (org.junit) and symbols related to JUnit.

- C. The error messages are related to compilation issues and missing dependencies, not an incorrect Maven command

```
Failed to execute goal
org.apache.maven.plugins:maven-compiler-plugin:3.8.0:testCompile
(default-testCompile) on project my-app: Compilation failure: Compilation
```

So it is D.the absence of the JUnit dependency(In the context of Maven and Java projects, a dependency is an external library or module that your project relies on. )

## Q11

Bob wants to store a table containing students answers in an SQL database. The table

should have 3 attributes:

- student (an 8 digit number which references the id attribute of the student table)
- question (a 2 digit number which references the id attribute of the question table)
- answer (text containing their answer)

What SQL command would create the most appropriate table?

A. CREATE TABLE answers(student VARCHAR(8), question INTEGER, answer TEXT, FOREIGN KEY student REFERENCES student(id), FOREIGN KEY question REFERENCES question(id), PRIMARY KEY (answer));

B. CREATE TABLE answers(student VARCHAR(8), question INTEGER, answer TEXT UNIQUE, FOREIGN KEY student REFERENCES student(id), FOREIGN KEY question REFERENCES question(id), PRIMARY KEY (student, question));

C. INSERT INTO answers(student, question, answer) VALUES ("0607970", 10, "C");

D. CREATE TABLE answers(student VARCHAR(8), question INTEGER, answer TEXT, FOREIGN KEY student REFERENCES student(id), FOREIGN KEY question REFERENCES question(id), PRIMARY KEY (student, question));

**primary key**:A primary key is a column or a set of columns in a table that uniquely identifies each row or record in that table. **foreign key** :A foreign key is a column or a set of columns in a table that refers to the primary key of another table. It establishes a link between the two tables. C is definitely wrong,so select among ABD BD's primary key is (student, question) because the pair of one student and one question is unique, kinds of like map? answer TEXT UNIQUE: wrong ,answer is randomly among ABCD?

GROUP BY

AVG()  
COUNT()  
SUM()

HAVING

## Q13

The FOREIGN KEY constraint implies which other constraints (if any)?

- A. NOT NULL, UNIQUE
- B. NOT NULL
- C. UNIQUE
- D. No other constraints

D. BTW, the PRIMARY KEY should be NOT NULL,UNIQUE

Q14

Bob is trying to run a command but it isn't working. To debug their issue they run their command with strace. What command are they trying to run?

```
execve("/usr/bin/cd", ["cd", "/home/jh18636"], 0x7fff0476de18 /* 28 vars */ = 0s

open("/usr/share/locale/en.utf8/LC_MESSAGES/bash.mo", O_RDONLY) = -1 ENOENT (No such
file or directory)
open("/usr/share/locale/en/LC_MESSAGES/bash.mo", O_RDONLY) = -1 ENOENT (No such
file
or directory)
write(2, "/usr/bin/cd: line 2: cd: /home/jh18636: No such file or directory
") = 66
read(255, "", 26)
= 0
exit_group(1)
= ?
+++ exited with 1 +++
```

- A. bash
- B. cd
- C. execve
- D. /etc/ld.so.preload

[1 mark]

Q15. Bob is trying to run a command but it still isn't working. Using the trace from before:

what type of program does strace try and run, in the first instance?

- A. A compiled binary command
- B. A shell script
- C. A shell builtin command
- D. Impossible to say

[1 mark]

Q16. Bob is trying to run a command but it still isn't working. Using the trace from before:

what's the problem?

- A. The command they've tried to run doesn't exist
- B. The permissions are set to prevent their command running

- C. The filesystem and it's inodes are corrupt.
- D. A folder they've tried to access doesn't exist

14. execve: checks for the command: so it is CD

15. using man strace because there is too many results that one command can't be done So it is shell script

16. "No such file or directory"

## Q18

Alice is trying to set up key-based login on a service she currently accesses via using ssh and entering her password. Here are listings of her .ssh directory on her local machine:

There's a problem evident with her current setup. Identify which file's presence, absence or

visible details indicates the problem.

- A. The problem lies with 'authorized\_keys' on the local machine.
- B. The problem lies with 'known\_hosts' on the server.
- C. The problem lies with 'id\_rsa' on the local machine.
- D. The problem lies with 'id\_rsa.pub' on the server.

C Because there should be a private key on the local machine

## Q19

Which of the below would redirect both standard output and standard error from myprogram to log.txt?

- A. ./myprogram > log.txt 1> log.txt
- B. ./myprogram 1> log.txt 2>&1
- C. ./myprogram 2>&1 > log.txt
- D. ./myprogram 2&1> log.txt

1> redirects standard output to the specified file (log.txt). 2>&1 redirects standard error to the same location as standard output, which is log.txt.

So, option B ensures that both standard output and standard error are redirected to the file log.txt

D. & here doesn't mean AND, so the correct version should be /myprogram 2>&1 > log.txt

## Q20

If command2 ignores standard input and will only work with a file passed as an argument,  
how can we redirect the output of command1 to be used by command2?

- A. command1 > command2
- B. command2 \$(command1)
- C. command1 | command2
- D. command2 <(command1)

D.<() is called process substitution in Bash. It allows the output of command1 to be treated as if it were a file, and this "file" is then passed as an argument to command2

A. command1 > command2 the output of command1 will be put into a FILE named command2 B. use the output of command1 to be the argument of command2 C. command2 's standards output is not ignored.

Q21

because the whole file is empty, so there is no file called '\*' and 'cp'...

## Undergrads paper 2

Q1

Alice wishes to rebase the bugfix bugfix branch so that instead of being based on the release-1.0 branch, it is based on the release-1.1 branch. Which commands should they run?

- A. git checkout bugfix; git rebase release-1.1
- B. git checkout bugfix; git merge release-1.1
- C. git checkout release-1.1; git rebase bugfix
- D. git checkout bugfix; git rebase release-1.1

这个考了rebase, 就是和merge不一样 merge是如果要把A merge到main上, 应该先跳转到main分支, 然后 merge A rebase是要看base是什么, 如果based是release-1.1, 那么就需要先去bugfix再rebase, 这样bugfix是在release-1.1的顶部了

Q12

Bob wants to find what their mean soup ranking is.

Which SQL query will not tell them what they want?

- A. `SELECT SUM(ranking)/COUNT(ranking) FROM soupranking;`
- B. `SELECT AVG(ranking) FROM soupranking;`
- C. `SELECT (1.0*SUM(ranking))/COUNT(ranking) FROM soupranking;`
- D. `SELECT SUM(ranking)/(1.0*COUNT(ranking)) FROM soupranking;`

这个在PPT上提到过, 实际上就是浮点数的处理问题 A.

Q14

Alice says Bob's code is dangerous. Bob says it isn't. The code doesn't seem to crash and

compiles. Who is right and why?

- ```
final String query = "SELECT password FROM users WHERE user LIKE "+username;
```
- A. Alice is probably right: it looks like it may be vulnerable to SQL injection.
 - B. Bob is probably right: it doesn't crash
 - C. Alice is probably right: there is a buffer overflow
 - D. Bob is probably right: they have used prepared statements

A.很容易被注入，为了防止注入，应用程序应该使用参数化查询或预处理语句

例如，考虑以下的 SQL 查询语句：

```
sql Copy code  
SELECT * FROM users WHERE username = '输入的用户名' AND password = '输入的密码';
```

如果应用程序简单地将用户输入直接插入到查询中，而不进行适当的处理，攻击者可以输入类似于 `'' OR '1'='1' --` 的内容，使查询变为：

```
sql Copy code  
SELECT * FROM users WHERE username = '' OR '1'='1' -- AND password = '输入的密码';
```

这将始终返回真，绕过了密码验证，因为 `1=1` 永远为真。攻击者可以利用这种方式来登录系统，甚至获取数据库中的所有用户信息。

Q18

Q18. Alice is trying to set up key-based login on a service she currently accesses via using ssh and entering her password. Here are listings of her .ssh directory on her local machine:

```
-rw----- 1 alice alice 389 Jan 19 10:56 authorized_keys
-rw----- 1 alice alice 395 Feb 21 13:03 id_rsa
-rw-r--r-- 1 alice alice 395 Feb 21 13:03 id_rsa.pub
-rw-r--r-- 1 alice alice 225 Feb 21 13:01 known_hosts
```

And on the server:

```
-rw-r--r-- 1 alice alice 395 Feb 21 13:21 id_rsa.pub
-rw-r--r-- 1 alice alice 225 Jan 19 11:02 known_hosts
```

There's a problem evident with her current setup. Identify which file's presence, absence or visible details indicates the problem.

- A. The problem lies with 'authorized_keys' on the local machine.
- B. The problem lies with 'authorized_keys' on the server.**
- C. The problem lies with 'known_hosts' on the local machine.
- D. The problem lies with 'id_rsa.pub' on the server.

Solution: The absence of an authorized_keys file on the server is a clear problem for key-based login. The permissions on the files are all correct. Moreover, 'authorized_keys' serves no purpose locally, and 'known_hosts' and 'id_rsa.pub' don't matter on the server. We went through this key-based setup process in the SSH labs at the start of Part 1.

这个还是考的本地和server SSH的问题 缺少authorized_keys

Q19

Q19. Which of the below would redirect both standard output and standard error from the command noisy to /dev/null?

- A. noisy > /dev/null 1> /dev/null
- B. noisy 2&1> /dev/null
- C. noisy 1> /dev/null 2> /dev/null
- D. noisy 2>&1 > /dev/null

需要redirect, B的语法错误, A是指数除了standard output D也是错的, 应该是先1, 再有2>&1 所以最后选C

If command2 needs to be given a string argument that comes from the output of command1,

how can we capture the output of command1 to be used by command2?

- A. command1 > command2
- B. command2 \$(command1)
- C. command1 | command2
- D. command2 <(command1)

string output->as arguments, 所以是B A是将command1的结果输出到一个名为command2的filezhong C.是将command1的输出作为command2的管道输入, 而不是argument? D.是重定向

past paper