



**CSE/C510Z**

**计算机组成与系统**

# 第一讲

Introduction, Linux and C

- 薛浩
- [stickmind.com](http://stickmind.com)

“

# 今日话题

- Introduction
- Linux
- C

推荐阅读：

- CSAPP, ch1
- K&R, ch1

”

“

## 今日话题

- Introduction
- Linux
- C

”



**如 何 / 为 何 ?**

# 关于课程

前置课程 CS101 目的是教会你如何通过编程解决[解决问题](#)，本课程在此基础上更深入一个层次，理解[如何](#) How 和[为何](#) Why 两个问题。

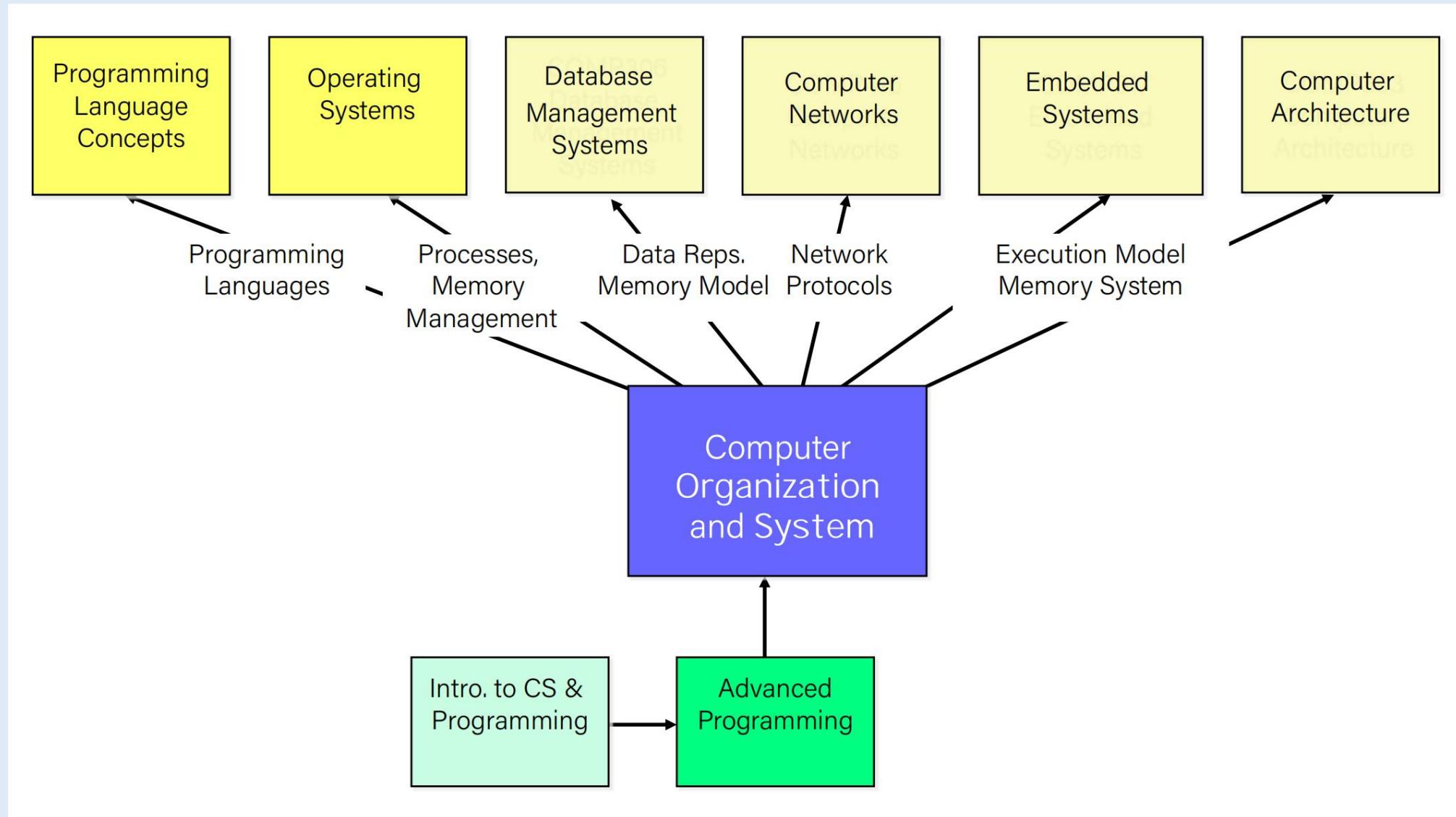
- 程序中的数据是[如何](#)真实地表示的
- 堆内存是[如何](#)工作的
- 计算机是[如何](#)知道按照何种方式执行程序的
- 程序是[如何](#)映射到计算机各个组件上的
- 程序[为何](#)没有按照预期执行

在这个层级上理解计算 Computing 有两个好处。

- 启发我们了解那些看似复杂的系统是如何工作的
- 提示我们的能力，更好地处理未来更大型的项目

本课程会进一步提升你的编程技能，让你在处理编程问题时，更能游刃有余。此外，本课程重度依赖调试技术，以此深入底层，一探究竟。在作业中，本课程强调如何更好地调试程序、如何编写更健壮的代码、如何进一步提升你的软件开发技能。

# 关于课程



# 课程目标

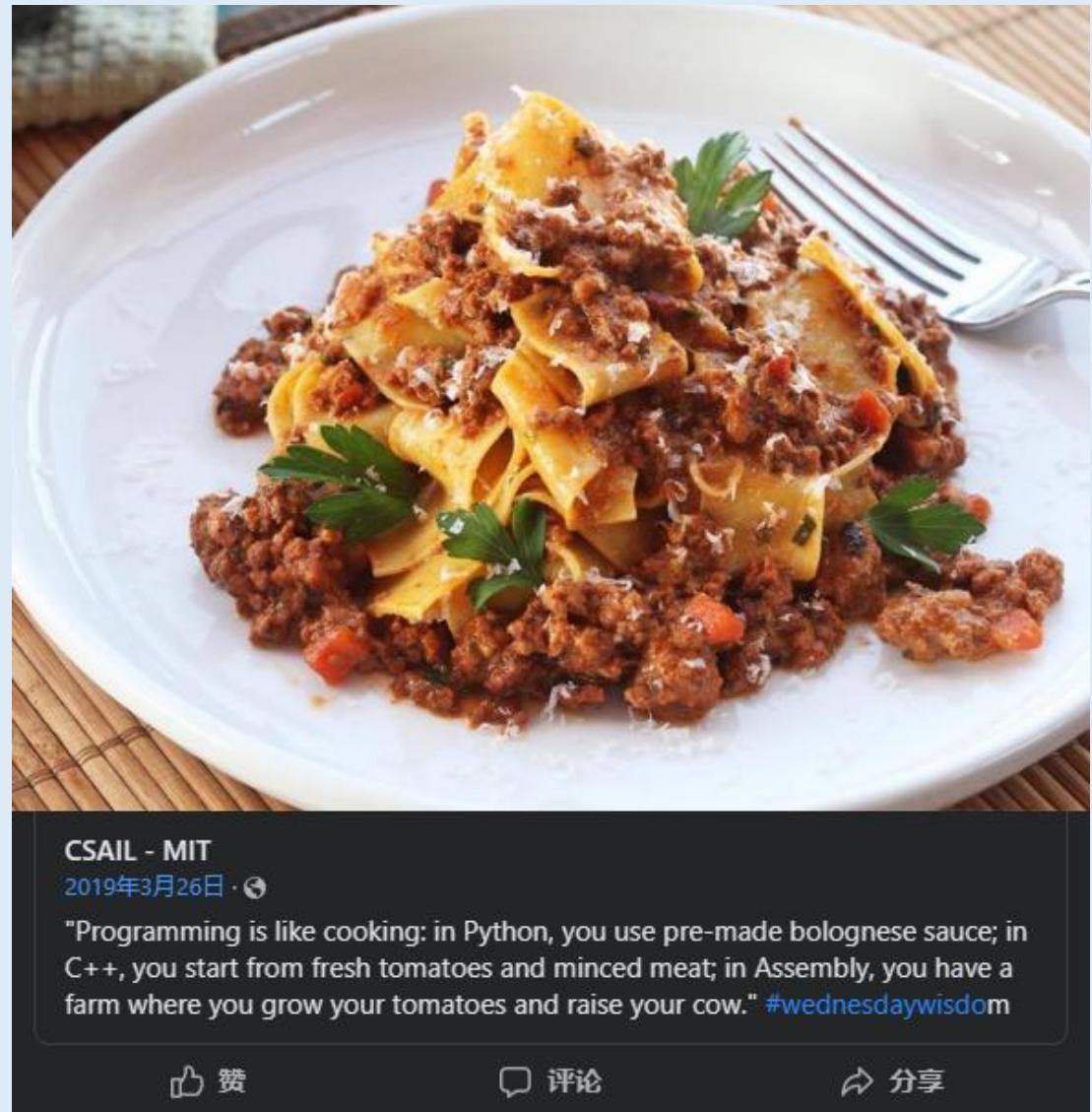
本课程的目标是让学生掌握

- 以复杂的内存和指针方式编写 C 程序
- 对地址空间以及 C 程序的编译/运行时行为，能够建立一个准确的模型

获得以下能力

- 把 C 语言转换到 x86-64 汇编语言
- 编写适配硬件算术局限性的程序
- 识别程序瓶颈并提高运行时性能
- 在 Linux 环境中开发程序

贯穿整个课程，最终会带你揭开计算机基础架构的奥秘。



# 课程大纲

- 话题 1：位（Bit）和字节（Byte） - 计算机是如何**表示数值**的？
- 话题 2：字符（Char）和字符串（C-String） - 计算机是如何**表示并处理文本**这样更复杂的数据的？
- 话题 3：指针（Pointer）、栈（Stack）和堆（Heap） - 在编程中如何有效地**管理不同数据类型的内存**？
- 话题 4：泛型（Generics） - 根据数据表示和内存相关的知识，如何**编写通用函数**能够处理不同类型的数据？
- 话题 5：汇编（Assembly） - 计算机如何**解释并执行 C 程序**？
- 话题 6：堆分配器（Heap Allocator） - 像 malloc 和 free 这些核心的**内存分配**操作是如何工作的？

除此之外，我们还会学习使用 Linux 开发环境、命令行工具与 C 语言，能够编写、测试、调试与优化我们的程序。

# 课程资源

课程官网

<https://cs102.stickmind.com/>

# 教学团队



薛浩

StickMind

xuehao0618@outlook.com



# 课程结构

课程主要包括如下三个部分

- 课堂：介绍一些概念，并作一些演示
- 实验：学习使用相关工具，并研究一些真实的项目案例，以便更好的完成作业
- 作业：提升编程技术，巩固课堂和实验中的内容

课程基本上按照  $2 + 1 + 1$  结构进行，即 2 次课安排一次实验，并发布一次作业。

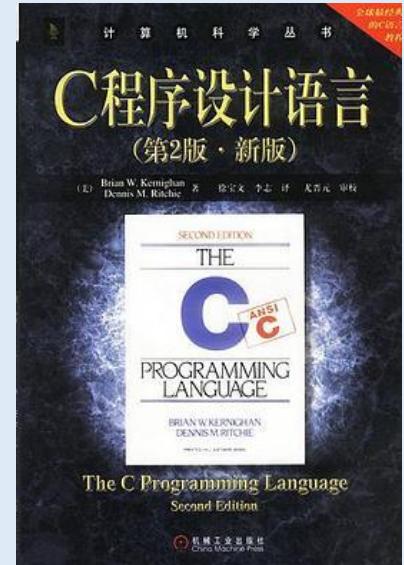
## 推荐教材

- 深入理解计算机系统（原书第3版）

- 作者: Randal E.Bryant / David O'Hallaron
- 译者: 龚奕利 / 贺莲
- 出版年: 2016-11
- ISBN: 9787111544937

- C 程序设计语言（第2版 · 新版）

- 作者: [美] Brian W. Kernighan / [美] Dennis M. Ritchie
- 译者: 徐宝文 / 李志译 / 尤晋元审校
- 出版年: 2004-1
- ISBN: 9787111128069



## 实验练习

- 实验课是探索的过程，会围绕最近的两次课的内容作进一步的研究，利用这段时间进一步练习你的技能
- 实验课会选取一些业界经典的案例作为研究对象，通过研究业界最佳实践，提供反思和批评的机会
- 每个实验还会为接下来的作业做一些铺垫，对实验中出现的问题和概念理解的越透彻，越能更好地完成作业

## 作业说明

本课程共 7 个作业，共 100 分，每个作业分值占比如下：

assign0	assign1	assign2	assign3	assign4	assign5	final project
5	15	15	15	15	15	20

作业提交后通过单元测试进行评分，全部通过即为满分，部分通过按比例计算分值。

## 作业说明

本课程共 7 个作业，共 100 分，每个作业分值占比如下：

assign0	assign1	assign2	assign3	assign4	assign5	final project
5	15	15	15	15	15	20

作业提交后通过单元测试进行评分，全部通过即为满分，部分通过按比例计算分值。

### 奖学金计划

前 6 个 Assignment (总分 80) 全部完成，并且总分  $\geq 65$  分即可获得奖学金 500 元。

# 作业说明

本课程共 7 个作业，共 100 分，每个作业分值占比如下：

assign0	assign1	assign2	assign3	assign4	assign5	final project
5	15	15	15	15	15	20

作业提交后通过**单元测试**进行评分，全部通过即为满分，部分通过按比例计算分值。

## Bonus

单元测试仅能够进行功能性 (functionality) 评估，最终评分会根据代码风格 (style) 是否优雅 (elegance) 提供 0~1 分的奖励。

6 个 Assignment 最多可以得到 5 分的 Bonus 奖励并计入总分。



问题 ?

“

## 今日话题

- Introduction
- Linux
- C

”



本课程开发环境选用远程 Linux 云端服务器和命令行工具，搭配本地终端和 VS Code 可以减少一些操作上的不习惯。



## FAQ 为什么使用远程服务器进行开发？

如果回到上个世纪 80 年代，即使电脑就在你的面前，你也必须使用终端才能对计算机进行操作。

现如今，随着个人计算机的发展，虽然图形界面早已成熟，但在真实的开发场景中，使用终端开发仍然占据主导地位。大量的前后端框架、开源项目、开发工具，也严重依赖终端操作。

本课程尽量还原类似的真实开发场景。当你以后有机会去软件公司实习时，你会发现，在这里掌握的技能，将会给你带来极大的回报。

# Linux 介绍

Linux 是 Unix 的一个变种，常见的 macOS 也是基于 Unix 开发。

Unix 是一个由一系列标准（standards）和软件开发工具（tools）组成的集合。

推荐阅读：

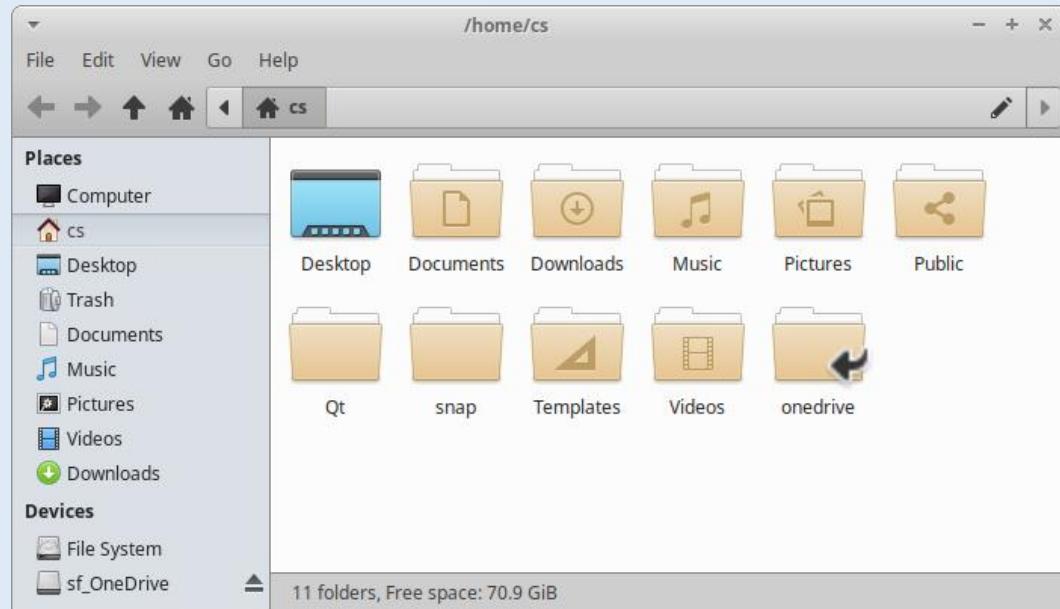
*The Strange Birth and Long Life of Unix,  
Warren Toomey, IEEE Spectrum, 28 Nov 2011*

访问 Linux 最方便的工具是命令行（command line），通过终端（terminal）来操作。

本课程介绍的常见命令，都可以在这些系统中无差别使用。

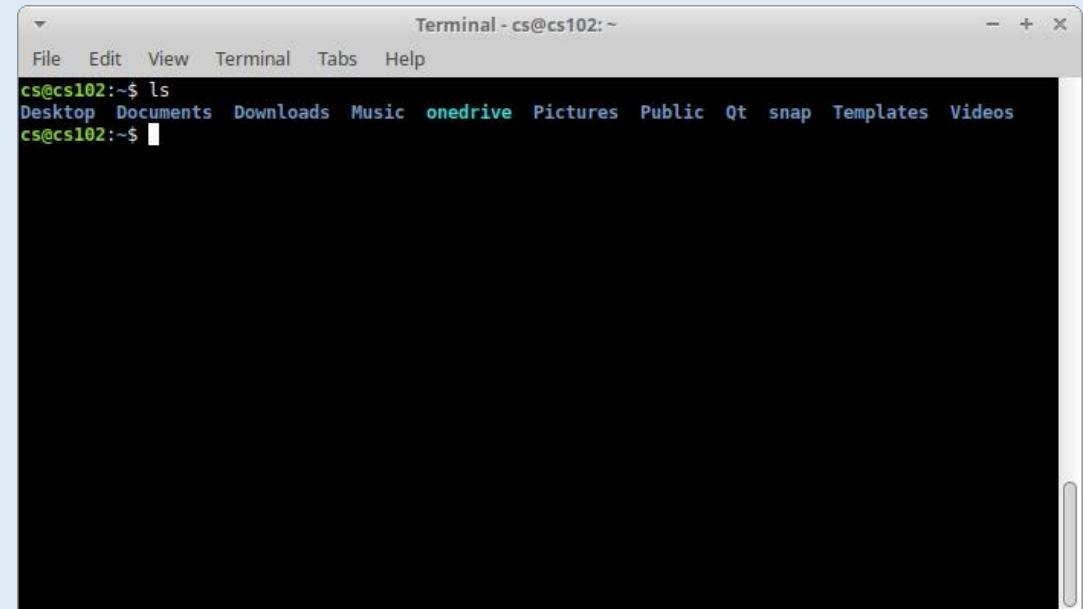


# 命令行介绍



现代 GUI 用户界面使用基于鼠标的操作方式。

查看文件夹下的文件，GUI 用户界面可以通过文件浏览器直接查看。



命令行是基于文本命令的操作方式。

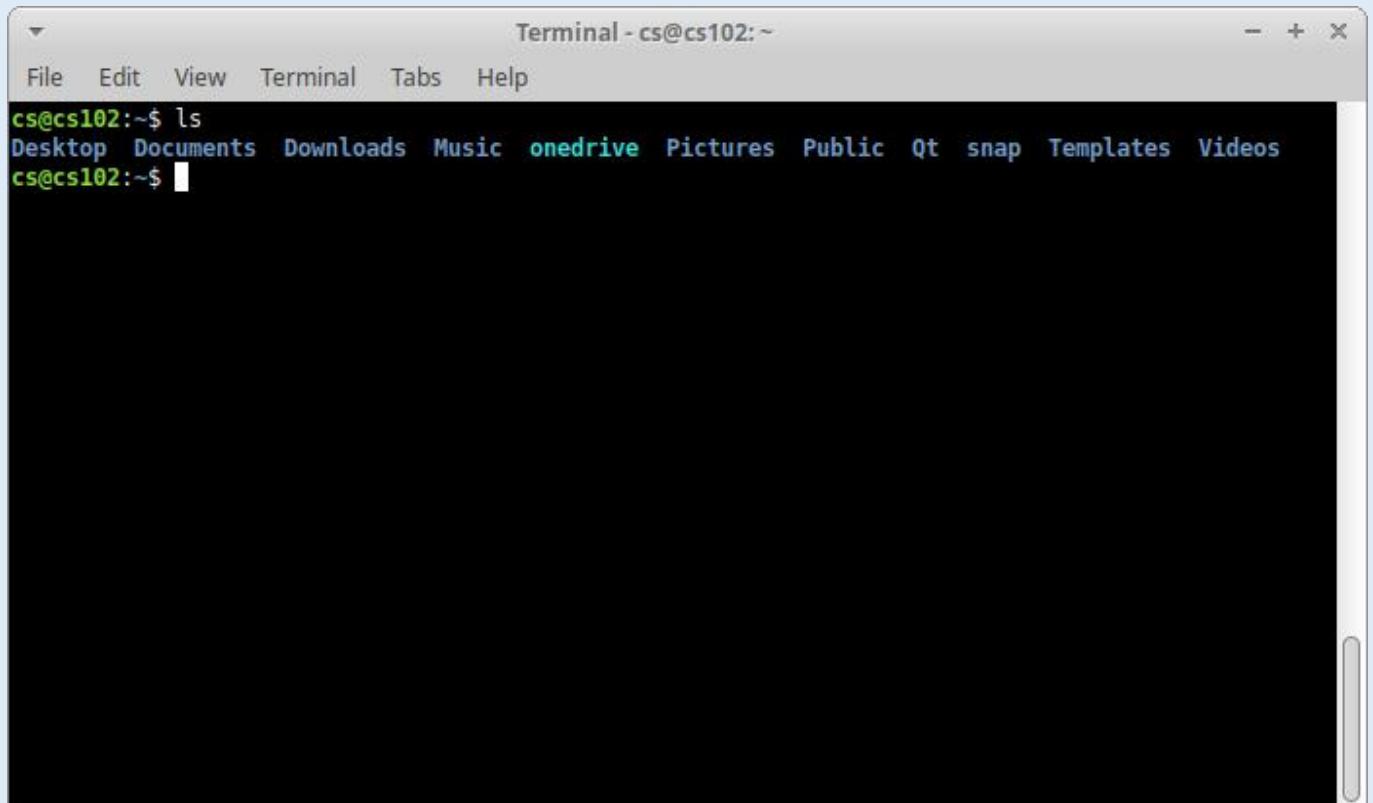
命令行方式需要输入 ls 命令，才会在屏幕上打印出当前目录下的文件列表。

“

## 对比练习

GUI 用户界面支持的操作，在命令行下都有对应的替代方式：

- 显示当前所在路径
- 访问或退出某个文件夹
- 创建文件、编辑文件
- 执行程序
- .....



A screenshot of a terminal window titled "Terminal - cs@cs102: ~". The window has a standard OS X-style title bar with icons for minimize, maximize, and close. The menu bar includes "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main terminal area shows the command "ls" being run at the prompt "cs@cs102:~\$". The output of the command is a list of directories: Desktop, Documents, Downloads, Music, onedrive, Pictures, Public, Qt, snap, Templates, and Videos. The terminal window is set against a light blue background.

```
Terminal - cs@cs102: ~
File Edit View Terminal Tabs Help
cs@cs102:~$ ls
Desktop Documents Downloads Music onedrive Pictures Public Qt snap Templates Videos
cs@cs102:~$
```

”

# 常用命令行工具

- cd
- cp
- ls
- mkdir
- Emacs / Vim
- rm
- man



## FAQ 如何学习使用 Linux 和命令行工具？

在 Linux 开发环境中开发程序，一开始会有很多不习惯。习惯了图形界面开发工具，面对黑乎乎的窗口，你甚至不清楚要输入什么字符，这些都是正常的。

学习在 Linux 环境中开发软件，只能通过不断地练习，慢慢熟悉这种操作方式。

最重要的是，在你遇到问题时，有人可以及时帮你答疑。所以，利用好课堂和飞书群，互相分享好用的技巧，一定会事半功倍。

在没有人帮助的情况下，要不断培养信息检索能力，通过搜索引擎解决自己的问题。



问题 ?

“

## 今日话题

- Introduction
- Linux
- C

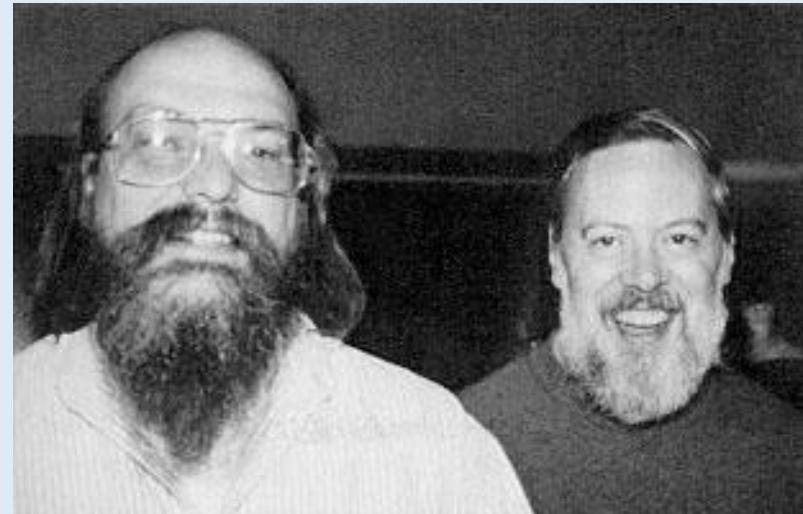
”

# C 语言

C 语言是贝尔实验室的 Dennis Ritchie 在 1970 年左右创建的。作为 Unix 系统的程序语言，伴随着 Unix 的发明而出现。

C 语言的成功与其设计哲学密切相关：

- 移植性：和 Unix 的密切关系，方便程序移植到新的设备上
- 简洁性：仅对硬件做了简单的抽象
- 实践性：为实现 Unix 而生，是系统编程的首选



C 语言一直位居 TIOBE 指数的前两名，TIOBE 指数是衡量编程语言受欢迎程度的指标。

## 语言异同

大部分现代编程语言 C++/Python/Java 都是基于 C 语言发展而来，基本的语法规则大同小异：

- 语法
- 基本数据类型
- 算术、关系、逻辑运算符

和 C++ 相比，C 语言的局限性主要有以下几点：

- 没有高级特性，比如运算符重载、默认参数、引用传参、类和对象、抽象数据类型等
- 没有大量的库，比如图形库、网络库等
- 编译器几乎不提供运行时检查，可能造成严重的安全问题

正因为如此，C 语言是一门不太复杂的语言，只需要一两百页的书就能够讲清楚。这里借用 K&R C 前言中的一句话：“C 是一门越用越顺手的语言”。



## FAQ 为什么还要学习 C 语言？

- C 语言仍然是非常流行的编程语言，常年占据 TIOBE 排行榜前三
- 大量的工具，甚至是其他编程语言（Python、Lua）都是由 C 语言编写
- C 语言是开发快速、高效软件的有力竞争者
- 在系统编程（操作系统、网络等）中，C 语言更为流行
- C 语言允许你在更低的抽象层操作数据，有利于理解系统的工作原理

# 第一个 C 程序

```
/* File: hello.c
 *
 * -----
 * This program prints a welcome message to the user.
 */

#include <stdio.h> // for printf

int main(int argc, char *argv[]) {
    printf("Hello, World!\n");
    return 0;
}
```

# 编写、编译、调试、运行

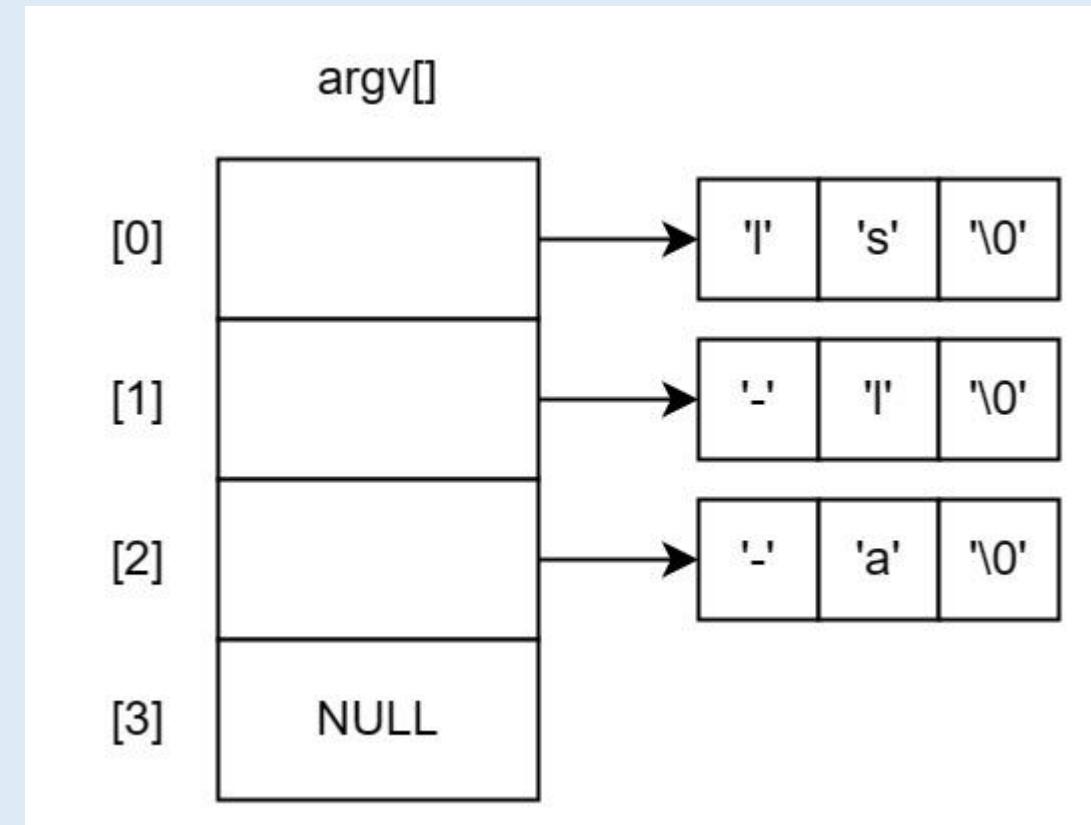
利用第一个 C 程序，演示在 Linux 环境中的工作流程：

- **ssh** - 连接远程服务器
- **vim** - 虽然现代编辑器非常优秀，但是在本课程中，总要学会使用一款命令行编辑器，Vim/Emacs 任选其一。
- **gcc** - 本课程不再使用集成开发环境，而是使用 GCC 编译器，这是使用最广泛的编译器之一。
- **gdb** - 本课程最棘手的地方是没有便利的图形化调试工具，所有作业都需要使用纯粹的命令调试方式。不过在实验环节，我们会提供大量的 gdb 调试训练。
- **./hello** - 运行程序

## 关于 `argc` 和 `argv[]`

在 Linux 中，执行命令行程序时，可以传递一些额外的参数。

例如，在调用 `ls -l -a` 时，它会执行程序 `ls` 并将字符串 `-l -a` 作为参数传递。



# 关于 argc 和 argv[]

```
/* args.c
* -----
* This program prints out information about its received
* command-line arguments.
*/
#include <stdio.h>

int main(int argc, char* argv[]) {
    printf("This program received %d argument(s)\n", argc);
    for (int i = 0; i < argc; i++) {
        printf("Argument %d: %s\n", i, argv[i]);
    }
    return 0;
}
```

# 关于 `printf`

- `printf` 用于向控制台输出（console output）字符串，可以通过格式化占位符（placeholder）插入变量或表达式的值。

```
printf(format_string, arg1, arg2, ...);
```

- 常见的占位符有：

- %s - string
- %d - integer
- %f - double
- %p - pointer

## 关于 **bool**

- 早期的 C 语言没有用来表示真和假的值，所以 C 把 0 当作假处理，非 0 当作真处理。
- C99 标准发布后，通过引入头文件 stdbool.h 可以使用 bool 类型，允许程序中使用 true 和 false 关键字，但编译器最终还是会转换成 1 和 0 两个值来处理。

```
#include <stdbool.h>

bool flag = true;
flag = false;
```



问题 ?

“

# 今日话题

- Introduction
- Linux
- C

推荐阅读：

- CSAPP, ch1
- K&R, ch1

第一次课 Q&A



打开飞书“扫一扫”

”