第1章 引　言

1.1　仓颉语言的起源

计算机高级编程语言发展到今天，已经出现了大量的高级编程语言，如 C、C++、Java、Python、Go 等，但是这些语言无一例外地都起源于国外设计者。

仓颉程序设计语言是由我国华为技术有限公司开发的高级计算机编程语言，“仓颉”一词源于我国古代传说仓颉造字，具有明显的中国文化特色。

尽管仓颉造字的故事流传了几千年，但是华为技术有限公司开发的仓颉编程语言的历史却很短。2020 年 8 月，华为技术有限公司注册了“仓颉语言”商标。2021 年 10 月，PLLab 在 Gitee 上为仓颉创建了仓库，并首先发布了 v0.22.3 版本，随后开始对部分开发者开放。到 2022 年 4 月底，PLLab 在 Gitee 上先后又发布了多个 alpha 版本。目前，仓颉程序设计语言作为一门新生的计算机高级编程语言还在不断发展及完善中。

1.2　仓颉语言的特点

仓颉程序设计语言是一门计算机高级编程语言，它具有一般高级编程语言的特点。仓颉语言同时也是结合了现代编程语言技术的面向全场景应用开发的通用编程语言，仓颉语言有以下主要特点。

（1）类型安全：仓颉语言是静态强类型语言，语言要求程序中所有的量必须有确定的类型，这样在编译时能够尽早发现程序中的类型不匹配错误，避免运行时出现错误。

（2）类型自动推定：仓颉编译器提供了强大的类型自动推断能力，在不存在歧义的情况下，自动推断判定类型，这样可以减少开发者在编写程序时进行类型标注的工作量，提高编写程序

代码的灵活性。

（3）内存自动管理：仓颉语言采用了垃圾收集机制，支持自动内存管理，避免了内存泄漏等错误。程序在运行时还会进行数组下标越界、计算溢出检查等，可以确保程序运行期间内存访问安全。

（4）多范式编程：仓颉语言支持函数式、面向对象方式和命令式的多种范式编程。语言融合了代数数据类型、高阶函数、模式匹配等函数式语言的先进特性。仓颉语言具有继承、封装、多态等面向对象的基本特征，支持类、对象、接口、泛型等。同时，仓颉语言还有值类型、全局函数等简洁高效的命令式语言特性。

（5）多线程：仓颉语言支持多线程，可以进行多线程并发编程。仓颉语言提供了原生的用户态轻量化的多线程，支持高并发编程。

（6）跨语言：仓颉语言可以实现与其他多种语言的互通，如在仓颉语言编写的程序中可以调用 C、Python 等语言编写的程序。仓颉可以高效调用其他主流编程语言，实现对其他语言库的复用和兼容。

（7）助力 AI 开发：仓颉提供了原生自动微分支持，可有效助力 AI 应用开发。

当然，仓颉程序设计语言目前还是一门新生的计算机高级编程语言，还处于不断演进和完善过程中，仓颉语言存在诸多优点，当然也会存在一些不足，开发者需要以发展的眼光看待仓颉程序设计语言。

1.3 本书面向的读者

本书旨在介绍仓颉程序设计语言的语法结构、数据类型、程序开发方法和程序组织等。尽管本书可以定位为一本介绍仓颉程序设计语言的基础书，但是本书不是一本介绍程序设计语言的基础书，因此希望读者具有一定的程序设计基础，这样读者可以更容易地学习和理解仓颉程序设计语言。

本书适合具备计算机软件专业基础，并想了解仓颉程序设计语言的特点和基本语言用法的读者，适合将要选用仓颉程序语言作为语言工具进行程序设计的开发者。

本书适合具有一种或多种高级语言基础的读者，如学习过 C、C++、Java、Python、Go 等一种或多种高级编程语言，想快速认识及了解仓颉程序设计语言的读者。

本书并非专门面向没有学习过任何编程基础的读者，这些读者在阅读本书时可能会遇到一定的困难，本书内容不太适用于以仓颉程序设计语言作为入门编程语言的读者。

第2章

第1个仓颉程序

2.1 仓颉 Hello World 程序

在开始介绍一种编程语言时，一般会从 Hello World 程序开始。下面是使用仓颉语言编写的输出"Hello World!"的程序。

首先创建一个文本文件，命名为 hello.cj，并通过文本编辑器打开该文件，输入以下代码：

```
//ch02/proj0201/src/hello.cj
/* 文件名：hello.cj
   功能：输出 Hello World!
   说明：这是第 1 个仓颉语言程序
*/
main(){
    print("Hello World!\n") //输出 Hello World!
}
```

仓颉程序默认的扩展名是 cj，即 Cangjie 的缩写。

以上代码，位于/*和*/之间的内容为注释，注释是为了方便程序员理解程序，注释内容对程序运行没有影响。

第 1 和 7 行//后面的内容也为注释，在仓颉语言中可以通过/*和*/进行多行注释，通过//进行单行注释。

第 6 行中的 main 是程序入口，是程序的起点，main 在仓颉程序里是一个特殊的函数，也称为 main 函数或主函数，每个用仓颉语言所编写的程序有且只有一个主函数，主函数是仓颉程序执行的入口函数，其后有一对圆括号“()”，圆括号内可以有参数，参数可以传递给当前函数，也可以没有参数，这里的主函数没有参数。

位于左花括号“{”和右花括号“}”之间的内容为主函数的函数体，函数体是函数要执行的代码操作。

第 7 行调用了系统内置的 print()函数，该函数是标准输出函数，功能是向屏幕输出需要打印的内容，输出函数参数中双引号中间的内容为输出的字符串内容，其中\n 表示换行，相当于输出键盘上的 Enter 键。

以上代码是一个简单的输出"Hello World!"并换行的仓颉语言源代码程序，仓颉程序编译后，运行从主函数开始，主函数中可以有任意合法的仓颉代码，如定义变量、调用其他函数等，仓颉程序执行后最终会回到主函数结束。

备注：仓颉语言以 main()函数作为入口函数，这一点和 C/C++语言类似，主函数的定义上和 Go 语言基本相同，但和 Java 不同，Java 要求必须建立一个含有静态 main 方法的主类，因此可以说仓颉语言不是纯粹的面向对象语言。

2.2 编译和运行

编译（Compile）是将用程序语言编写的源程序生成目标程序的过程。编译过程一般由编译器完成，编译器其实也是一个程序，其主要功能是将源代码翻译成目标代码，如 C 语言常用的编译器 GCC、Java 语言编译器 Javac 等。

仓颉语言是编译型语言，通过仓颉语言书写的源程序需要进行编译后才能在操作系统上运行，仓颉语言的专用编译器为 cjc，其名字为 Cangjie Compile 的缩写。

目前，仓颉工具链已适配 Linux 和 Windows 平台，下面首先在 Linux Ubuntu18.04 环境下安装仓颉编译器，并运行第 1 个仓颉程序。

2.2.1 在 Ubuntu 系统下编译运行

首先，在仓颉官网下载相应安装包，这里以 CangJie_0.39.5-linux_x64.tar.gz 为例，当然随着仓颉语言的发展，其版本号会不断变化，由于当前的版本是 0.39.5，所以以此版本为例。这里下载的压缩包放置到了当前用户的主目录/home/ubuntu 下，在 Ubuntu 系统的终端下通过 ls 命令查看，结果如图 2-1 所示。

首先对下载的仓颉安装包进行解压，通过 tar 工具进行解压的具体命令如下：

```
tar -xvf CangJie_0.39.5-linux_x64.tar.gz
```

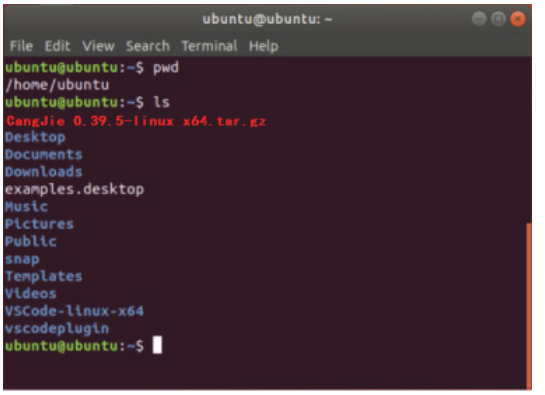

图 2-1　下载的仓颉安装包

解压后，会在当前目录下生成一个名称为 cangjie 的目录，仓颉语言的基本库及编译器等都默认存放在该目录下。为了能够使用仓颉语言提供的功能，还需要进行必要的环境配置。仓颉安装包中提供了一个配置环境的脚本文件 envsetup.sh，可以通过 source 命令快速地配置仓颉环境变量，具体命令如下：

```
source cangjie/envsetup.sh
```

当环境变量配置成功后，一般情况下仓颉编程环境即完成配置，为了进一步验证仓颉编程环境是否配置成功，可以使用编译器版本查看命令进行验证，具体命令如下：

```
cjc -v
```

执行以上命令后，如果出现如图 2-2 所示的版本信息，则说明仓颉编译器安装成功。

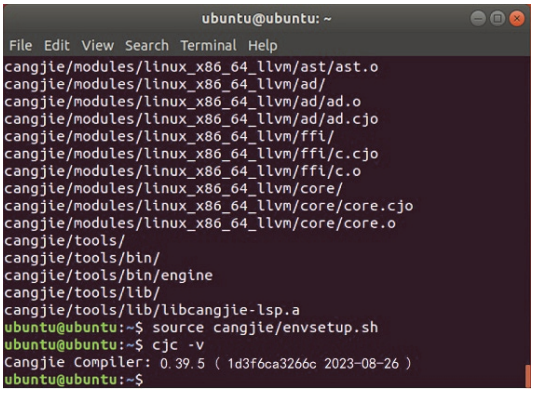

图 2-2　仓颉编译器安装成功

接下来，可以在当前目录下创建 hello.cj 文件，并录入 Hello World 程序代码，或复制前面创建的 hello.cj 文件到当前目录。仓颉源代码程序其实就是一个文本文件，其扩展名为仓颉两个字拼音的缩写 cj，创建好仓颉源程序 hello.cj 后，通过下面的命令编译该仓颉程序：

```
cjc hello.cj -o hello
```

这里的 cjc 用于调用仓颉语言的编译器对源程序进行编译，hello.cj 为要编译的仓颉源代码程序，-o 表示输出，hello 表示编译成功后生成的文件名。编译成功后，会在当前目录下生成可执行文件 hello，可以通过 ls 命令可以查看当前目录下的所有文件，接下来可以运行生成的可执行程序，运行方式如下：

```
./hello
```

该程序运行后会输出"Hello World!"，如图 2-3 所示。

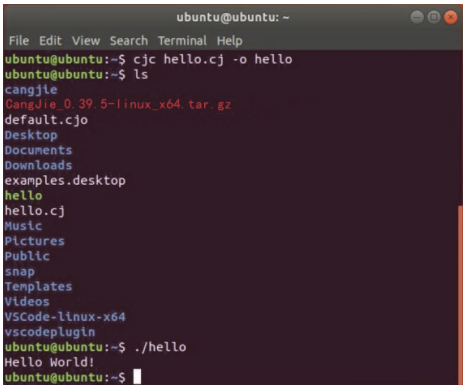

图 2-3　编译运行第 1 个仓颉程序

至此，在 Ubuntu 系统环境下的第 1 个仓颉程序的编写、编译、运行成功了。

备注：仓颉语言是一种编译型语言，程序运行之前需要完成编译，编译后生成可执行文件。这一点和 C/C++相同，但和 Java、Python 不同，Java 程序的运行需要 Java 虚拟机，而 Python 是解释型语言。编译型语言编译后生成的是可以直接执行的代码，所以执行效率相对较高。

2.2.2　在 Windows 10 系统下开发仓颉程序

当前，仓颉语言还不支持在 Windows 环境下直接编译运行，但是可以在 Windows 操作系统上通过一定的配置进行仓颉程序开发。

在 Windows 环境下开发仓颉程序需要虚拟环境的支持，一种方法是直接在 Windows 系统上安装虚拟机管理系统，如 VMware Workstation 等，并在虚拟机管理系统中创建安装 Linux 操作系统环境。如果开发者通过这种方式在 Windows 10 上安装了虚拟机管理系统，并创建安装了 Ubuntu 操作系统环境，则在 Windows 10 环境下开发仓颉程序也就完全变成了在 Ubuntu 环境中编写仓颉程序。

鉴于一些开发者更熟悉在 Windows 环境下编写程序，这里给出采用 Windows 10+WSL 的方式编写和编译仓颉语言程序的方法。

WSL 是 Windows Subsystem for Linux 的简称，是 Windows 为 Linux 提供的一个子系统，是一个在 Windows 10 上能够运行原生 Linux 二进制可执行文件的兼容层，也可以理解为一个轻量级的虚拟机环境。下面具体说明 Windows 10+WSL 环境的配置和开发仓颉程序的过程。

首先，在 Windows 10 环境的开始菜单选择设置进入设置界面，如图 2-4 所示。

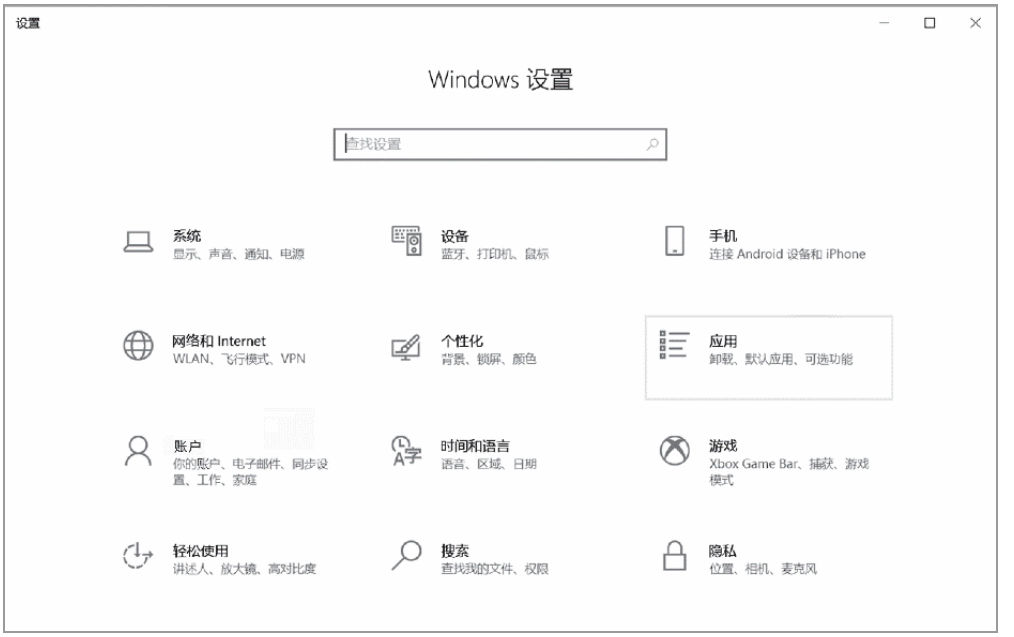

图 2-4　Windows 设置

选择“应用”进入，在“应用和功能”下，选择“可选功能”，在“可选功能”界面下方可以看到更多 Windows 功能，如图 2-5 和图 2-6 所示。

图 2-5　应用和功能

图 2-6　可选功能

进入 Windows 功能界面，如图 2-7 所示，找到“适用于 Linux 的 Windows 子系统”选项，勾选后单击“确定”按钮。

由于 Windows 10 默认情况下没有开启 WSL 功能，因此需要以上步骤进行开启。待 Windows 功能已经完成请求的更改后，立即重新启动计算机，如图 2-8 所示。

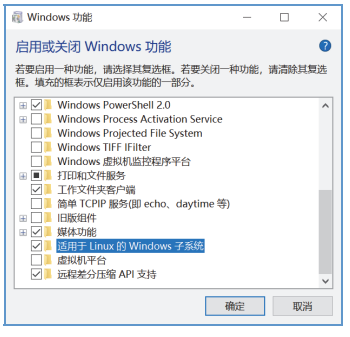

图 2-7　Windows 功能

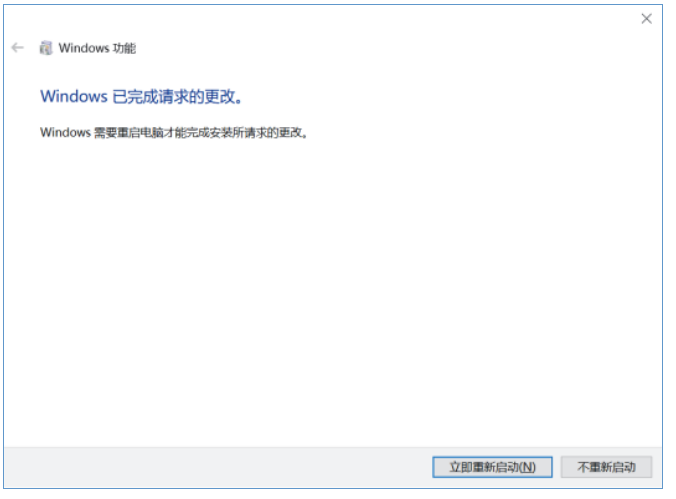

图 2-8　Windows 功能已完成请求的更改

重启计算机后便可进入 cmd 命令行界面，如图 2-9 所示，键入 wsl 命令后会提示尚未安装，可以继续键入 wsl --list --online 命令查看可安装的系统列表，如图 2-9 所示，其中包含了

Ubuntu-18.04 系统，通过命令 wsl　--install　-d　Ubuntu-18.04 下载并安装 Ubuntu 系统，此后会进入自动下载并安装过程，如图 2-9 和图 2-10 所示。

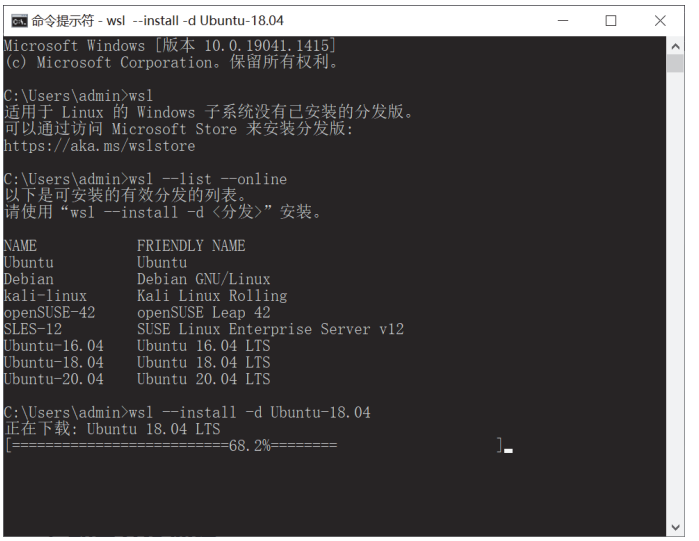

图 2-9　安装 Ubuntu

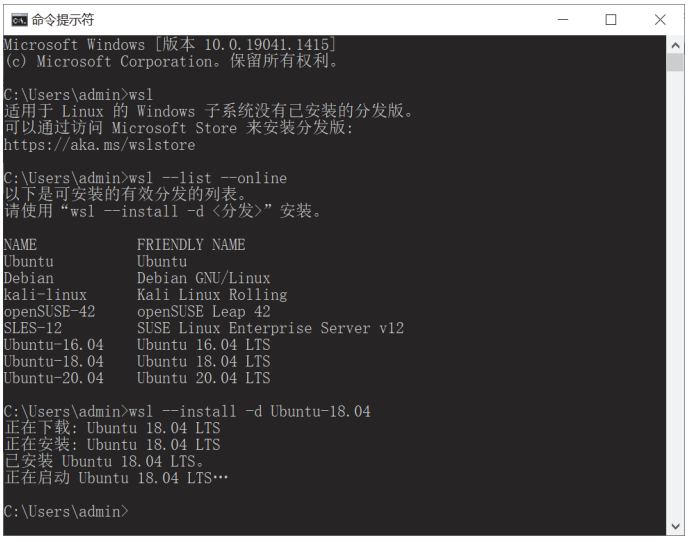

图 2-10　安装 Ubuntu 系统成功

在安装 Ubuntu 虚拟环境的过程中，需要创建一个用户名和密码，所创建的用户名不能是 root 账户，当输入的用户名为 root 时会提示用户名已经存在，因为 root 默认是系统的内置超级管理账户。这里重新键入用户名 laz，并重复输入两次密码后创建该用户。待出现命令行提示符时，说明 Ubuntu 系统已经安装成功，如图 2-11 所示。

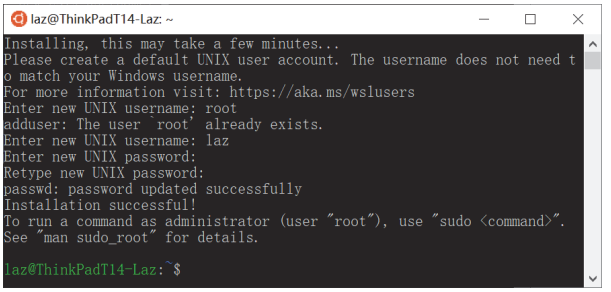

图 2-11　进入 Ubuntu 系统

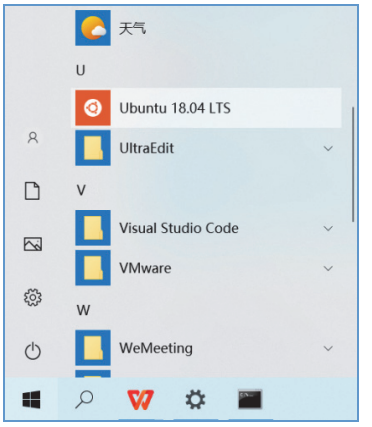

图 2-12　开始菜单中的 Ubuntu 系统启动项

这样，就在 Windows 系统环境下建立了一个 Ubuntu 虚拟环境系统，即适用于 Linux 的 Windows 子系统。该子系统再次使用时，可以在 Windows 的开始菜单中找到并进行启动，该子系统的默认名称为 Ubuntu18.04 LTS，如图 2-12 所示。

为了能够在所安装的 Ubuntu 环境运行仓颉语言程序并能够进行开发，还需要解决必要的相关依赖，可以在终端界面（见图 2-11）中依次运行下面的命令进行相关软件包的安装及必要的配置，执行命令后会从网上自动下载相关的包，并进行自动安装，所需时长也视网络情况而定，开发者需要等待并按照必要的提示确认安装即可。

```
sudo sed -i s@/archive.ubuntu.com/@/mirrors.aliyun.com/@g /etc/apt/sources.list
sudo  apt-get  update
sudo  apt  full-upgrade  -y
sudo  apt  autoremove
sudo  apt-get  install  build-essential
sudo  apt  install  libncurses5
sudo  apt-get  install  libgcc-7-dev
sudo  apt-get  install  binutils
sudo  ln  -s  /usr/lib/x86_64-Linux-gnu/libstdc++.so.6  /usr/lib/libstdc++.so
```

如果开发者需要在集成开发环境 Visual Studio Code 中进行开发，则可以通过下面的命令安装该集成开发环境：

```
sudo apt-add-repository -r ppa:Ubuntu-desktop/ubuntu-make
sudo apt update
sudo apt install ubuntu-make
sudo umake ide visual-studio-code
```

执行以上命令时，也可能会出现需要确认的情况，如提示[I Accept (a)/I don't accept (N)]，则输入 a 确认即可。

安装好 Ubuntu 环境后便可以非常方便地访问 Windows 下的文件系统，Windows 的盘符默

认挂载在/mnt 目录下，通过 ls 命令可以看到 Windows 的盘符，如图 2-13 所示。

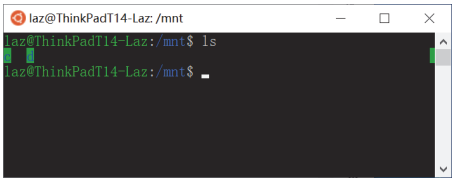

图 2-13　Ubuntu 环境下的 Windows 盘符

至此，WSL 环境已经完全配置成功，可以在 WSL 中很容易地访问 Windows 系统中的文件，接下来配置仓颉开发环境。

首先，从仓颉官网下载相应安装包并解压，然后放置到 Windows 系统的目录中，这里以 Cangjie_0.26.2 为例，解压后放到 Windows 系统的 D 盘 Cangjie_0.26.2 目录下，如图 2-14 所示。

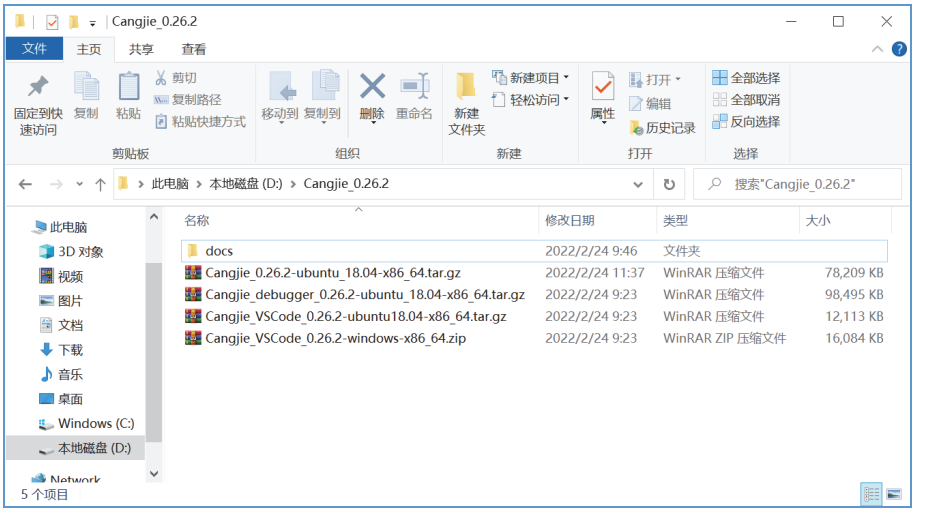

图 2-14　下载的仓颉安装包

在前面的 Ubuntu 环境下解压仓颉安装包，可以首先切换到对应的/mnt/d/Cangjie_0.26.2 目录下，然后执行解压安装命令，如图 2-15 所示。

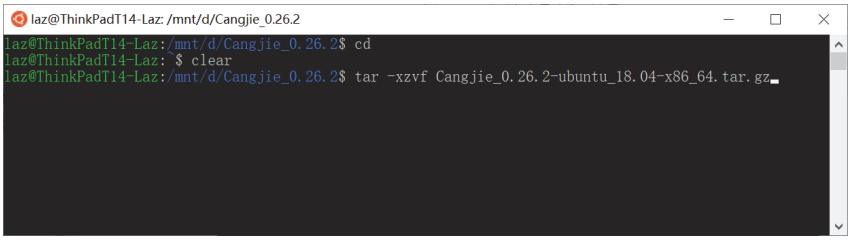

图 2-15　解压仓颉安装包

通过下面的命令进行系统环境变量设置和查看版本，通过查看安装版本命令验证仓颉编译

器是否安装成功。

```
source cangjie/envsetup.sh
cjc -v
```

执行以上命令后，如果出现如图 2-16 所示的版本信息，则说明仓颉编译环境配置成功。

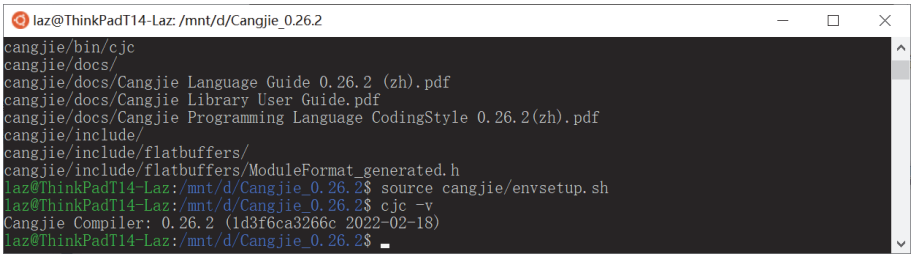

图 2-16　仓颉编译环境配置成功

在如图 2-16 所示的终端下编写仓颉程序和在 Ubuntu 环境下编写一样，前面已经介绍过，这里不再赘述。另外，由于所配置的 Ubuntu 子系统环境可以非常方便地访问 Windows 文件系统中的文件，因此可以在 Windows 下使用任意的文本编辑器编写仓颉程序并保存，通过图 2-16 所示的终端找到对应的源代码，并通过仓颉编译器 cjc 进行编译后即可运行仓颉程序。

为了提高开发效率，在 Windows 10 环境下，还可以通过 Visual Studio Code 进行仓颉程序开发。使用该集成开发环境可以提高开发效率和获得编辑代码提示，在 Windows 10 系统下配置 Visual Studio Code 进行仓颉程序开发的具体步骤如下。

第 1 步，下载并安装 Visual Studio Code，通过官方网址 https://code.visualstudio.com/Download 下载 Windows 系统的相应版本，下载完成后，双击安装文件便可安装。此过程和一般的在 Windows 环境下安装软件的方法相同，按向导安装即可。

第 2 步，解压 Cangjie_VSCode_0.26.2-Windows-x86_64.zip 得到同名文件夹，解压后的文件和文件夹主要如下：

（1）modules 文件夹

（2）LSPServer.exe

（3）Cangjie-lsp-0.26.2.vsix

（4）Cangjie-Format-0.26.2.vsix

（5）Cangjie VS Code Plugin User Guide.pdf（该文件是插件使用说明文档）

为了配置方便，建议把以上前 4 项全部移动到一个相对简单的路径目录中，这里假设为 D:\VSCode，后续以此路径配置。

第 3 步，打开 VS Code 以便安装本地插件，依次单击扩展按钮和更多，选择 Install from VSIX 以便安装插件，如图 2-17 所示。

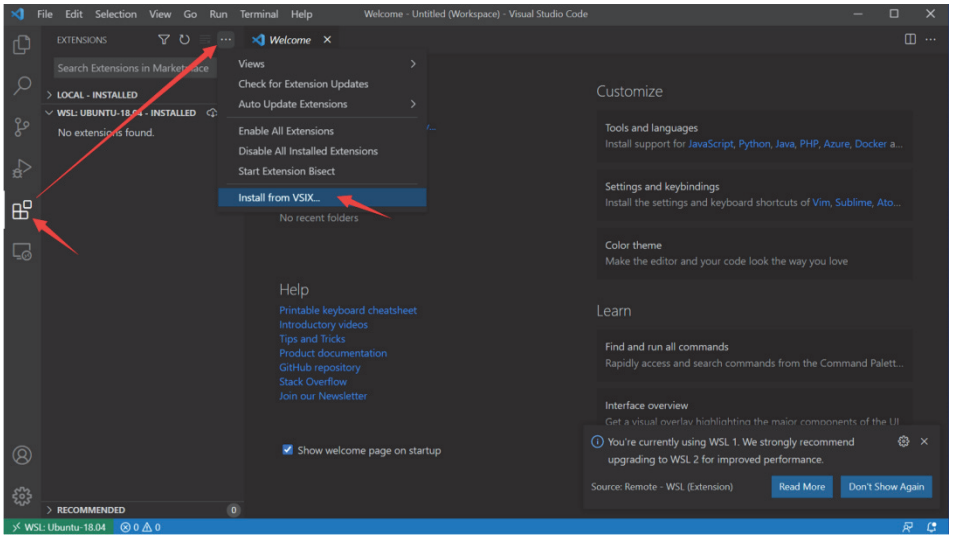

图 2-17　选择安装插件

选择浏览本地，定位到第 2 步解压的文件的位置，选择 Cangjie-lsp-0.26.2.vsix 文件以便安装仓颉插件，如图 2-18 所示，单击 Install 按钮进行安装，同样方法安装 Cangjie-Format-0.26.2.vsix 插件。

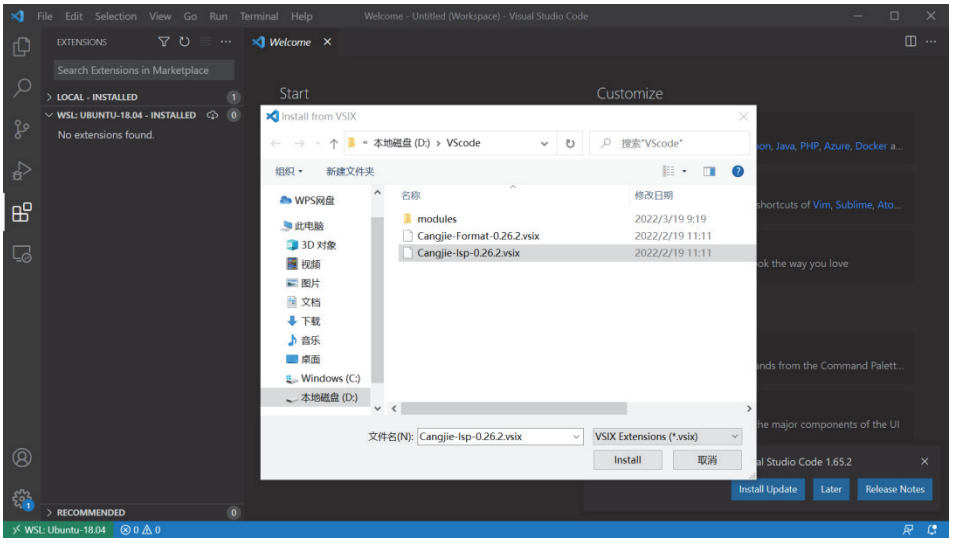

图 2-18　确认安装插件

第 4 步，配置服务器端路径，单击 Cangjie 插件右下角的齿轮，选择扩展设置（Extension Settings），在仓颉语言服务路径配置框中输入 LSPServer.exe 所在的绝对路径，这里是 D:\VSCode\LSPServer.exe，如图 2-19 所示。

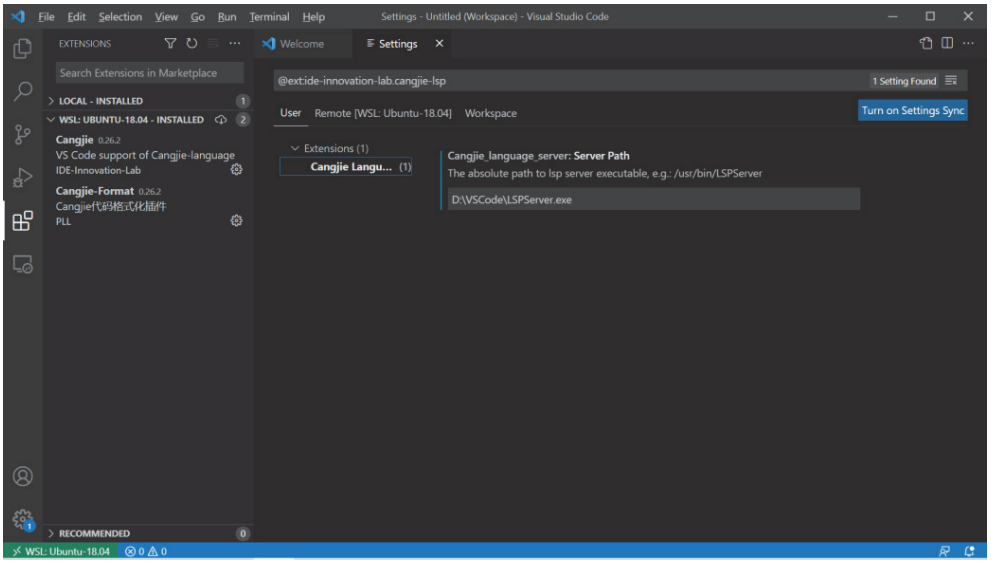

图 2-19　设置仓颉语言服务路径

第 5 步，重启 Visual Studio Code 以便使配置生效。

第 6 步，在 Windows 环境下创建存放仓颉代码的文件夹，这里为 D:\VSCode\mycode，在该文件夹下创建 project 文件夹，并在 project 文件夹下创建 src 文件夹，在 src 文件夹下创建仓颉源代码文件，这里命名为 hello.cj，其中，project 和 src 目录是当前采用 Visual Studio Code 开发仓颉项目的固定目录结构。

第 7 步，通过 Visual Studio Code 打开项目文件夹，如图 2-20 所示。

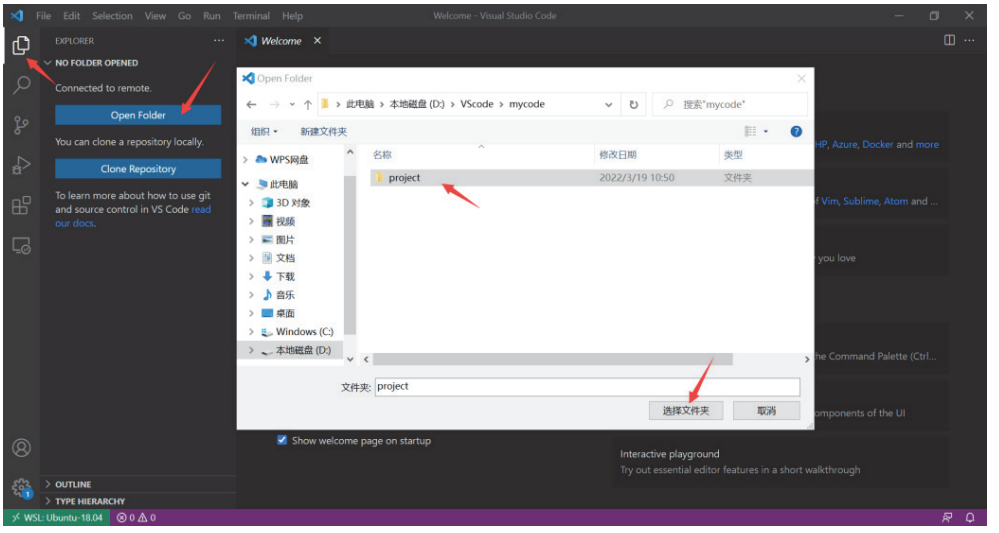

图 2-20　选择项目文件夹

第 8 步，通过 Visual Studio Code 打开仓颉源文件，进行源代码编写。理论上，在 src 目录下，开发者可以创建任意多的包或源文件进行开发，这里仅对 hello.cj 文件进行编辑，实现了仓颉 Hello World 程序，如图 2-21 所示。Visual Studio Code 集成开发环境为开发者提供了代码着色、补全、查找等多种功能，是很好的集成开发环境，可以大大提高开发效率，当然，针对仓颉语言的支持还在不断完善中。

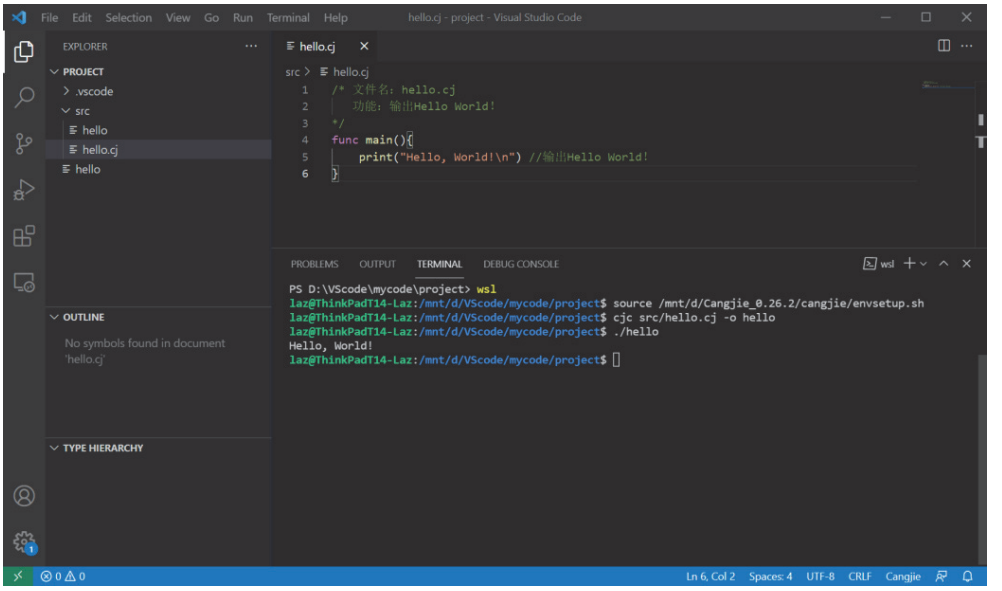

图 2-21　在 Visual Studio Code 中开发仓颉程序

第 9 步，编译运行。通过 Visual Studio Code 编写的仓颉代码需要编译后才能运行，编译方法可以在如图 2-16 所示的终端下通过编译器 cjc 进行编译，也可以通过 Visual Studio Code 集成的终端（Terminal）进行编译，如图 2-21 所示，通过 wsl 命令转换到 Ubuntu 环境中，通过 source 命令配置环境变量，通过 cjc 编译仓颉源程序后便可以在终端中运行编译后的代码。如果终端没有打开，则可以通过 VS Code 的菜单 Terminal 打开终端窗口。

通过以上步骤，可以在 Windows 10 环境下，通过 Visual Studio Code 进行仓颉程序的开发。由于目前仓颉语言编译环境依赖于 Linux，所以不管采用什么方法编写仓颉源代码，最终都需要在 Linux 环境中进行编译和运行。

备注：用仓颉语言编写的程序在本书撰写过程中，还仅支持在 Linux 环境下运行。尽管这里介绍了在 Windows 10 下搭建和开发仓颉程序的方法，但是实际上也是通过配置了 Linux 虚拟环境才可以在 Windows 10 环境下对仓颉程序进行编译和运行。相信仓颉语言未来会支持在更多的操作系统上直接编译和运行。

2.3 仓颉程序包含的元素

为了说明仓颉程序包含的元素，首先给出一个仓颉源程序文件 test.cj，代码如下：

```
//ch02/proj0202/src/test.cj
/* 文件名：test.cj
    功能：求和
*/
main(){
    let  a:Int32 = 10
    var  b:Int32 = 20
    var  c = a + b ; b = 30
    print("c = ${c}\n") //输出 c
}
```

一个仓颉程序主要包含以下元素。

1. 主函数

主函数可以称为 main 函数，一个仓颉程序有且仅有一个 main 函数，它是程序运行的入口函数，也是程序运行的结束函数。在主函数中可以调用其他代码，其他代码还可以继续调用别的代码，但是不管调用有多深，最终还是要返回主函数结束运行的程序。

2. 常量

常量是程序中不变量的量，也可以称为字面量，如上例中的 10、20 都是常量，这些数值常量的默认形式为十进制。

3. 变量

变量是由程序定义的值可以改变的量，如上例中的 a、b、c。变量有时也会被关键字限制改变其值，如上例中的 a 采用了 let 关键字修饰，a 虽然是一个变量，但是其不能被再次赋值。

4. 运算符

运算符是程序中进行运算功能的符号，如上例中的加号“+”、等号“=”，+表示把两个量相加，=表示把右边的值赋给左边。仓颉提供了大量的运算符，可以完成常见的算术、关系、逻辑等基本运算。

5. 关键字

关键字是仓颉语言固定了特定含义的一些单词或单词缩写，如上例中关键字 func 表示定义函数，关键字 var 表示定义变量。

6. 语句

语句是程序中执行的一个操作或表达式，仓颉语言中一般一行表示一个语句。当一行表示一个语句时，其后面可以有分号，也可以省略分号。如果在同一行想书写多个语句，则可以通

过“;”符号分隔。

7. 分隔符

分隔符主要是为了分隔不同的单元或部分，如上例中的左右花括号，即{和}，它们界定了主函数的函数体边界。

8. 函数

函数是一个功能模块，在程序中有具体的名称，可以方便和其他模块区分及被调用，如上例中的 main 函数，另外上例中还调用了系统函数 print()。

9. 注释

注释主要为了使程序更容易被理解，而在程序中书写的说明部分，注释对程序本身的运行没有作用。仓颉程序中位于/*和*/之间的内容为注释，一行中位于//后面的内容也为注释。

当然，仓颉语言代码中包含的元素不限于上面提到的内容，更多语言元素会随着学习的深入不断展开介绍，这里只是说明了基本的几种元素。

备注：语句上，仓颉和 Python 类似，一般一行为一个语句，也可以通过分号把多个语句放到一行中，而 C/C++/Java 语言要求每个语句后面都必须有分号。仓颉允许左花括号单独成行，这一点和 Go 语言不同。注释上，仓颉和 C/C++/Java 语言相同，但和 Python 不同，Python 使用#号进行单行注释，对于多行注释 Python 使用 3 个连续的单引号或者 3 个连续的双引号。

第3章 基本类型和运算符

3.1 常量和变量

3.1.1 常量

常量是程序中表示固定值的量，在仓颉程序中常量包括数值常量、布尔常量、字符常量、字符串常量，数值常量又包括整型常量和浮点型常量。

1. 数值常量

1）整型数值常量

整型数值常量可以直接使用数字表示，如 10、-20 等。整型数值常量一般使用十进制表示，默认情况下即为十进制。仓颉支持整数类型常量采用二进制、八进制、十进制和十六进制 4 种进制表示形式。

二进制以 0b 或 0B 为前缀，如 0b101 或 0B101 都表示二进制的数值常量，对应的十进制数值是 5。

八进制以 0o 或 0O 为前缀，如 0o207 或 0O207 都表示八进制的数值常量，对应的十进制数值是 135。

十进制没有前缀，这一点和日常表示数值比较一致，如 128 就表示十进制数一百二十八。

十六进制以 0x 或 0X 为前缀，如 0xFF 或 0XFF 都表示十六进制的数值常量，对应的十进制数值是 255。

在整型常量各种进制的表示中，可以使用下画线 _ 充当分隔符，以方便识别数值的位数，如二进制表示 0b0001_1000 等价于 0b00011000，前者看上去更清晰，特别是数值位数较多时，

使用下画线分隔符非常有益。分隔符可以隔开任意位数，如 600_123 和 60_0123 都表示十进制数 600123。但是分隔符不能位于数字的最前方，如用_123 表示十进制常数 123 是错误的。

2）浮点型数值常量

程序中，实数常用浮点型表示。在仓颉程序中，浮点型常量在进制上支持两种，即十进制和十六进制，在表示形式上支持小数表示和指数表示。

在十进制小数形式表示中，一个浮点型常量要包含小数部分，采用小数形式表示时，浮点型常量中的小数点不能省略，示例代码如下：

```
3.14          //十进制小数
0.314
.314          //等价于 0.314，小数点前面的 0 可以省略
314.          //小数点不能省略，如果省略，则认为是整数 314
-0.314        //负数
-.314         //等价于-0.314
-314.0        //负数，后面的.0 不能省略，否则会认为是整型数值
```

在十进制指数形式中，浮点型数值常量包含两部分，即小数部分和指数部分，前者为一个十进制浮点数，此时的小数点可以省略，指数部分以 e 或 E 为前缀表示底数为 10，以一个整数为后缀表示指数，示例代码如下：

```
2e3           //十进制指数表示法，值为 2 乘以 10 的 3 次方
2.6e-1        //十进制指数表示法，值为 2.6 乘以 10 的-1 次方
-.3e-2        //十进制指数表示法，值为-0.3 乘以 10 的-2 次方
-.3E-2        //十进制指数表示法，值为-0.3 乘以 10 的-2 次方
1e2           //表示 1 乘以 10 的 2 次方
e2            //错误，小数部分不能省略
1e1           //表示 1 乘以 10 的 1 次方
1e            //错误，指数部分不能省略
1e1.0         //错误，指数部分必须为整数
```

在十六进制情况下，浮点型常量必须以指数形式表示，表示仍然包含小数部分和指数部分。小数部分以 0x 或 0X 为前缀，表示是十六进制。指数部分不能省略，并且指数部分以 p 或 P 做前缀表示底数为 2，p 或 P 后面的整数表示幂。十六进制表示的浮点型常量中，仅小数部分为十六进制，指数部分仍然是十进制，示例代码如下：

```
0x1.1p0       //十六进制 1.1 乘以十进制数 2 的 0 次方，值为十进制 1.0625
0xap2         //十六进制 a 乘以十进制 2 的 2 次方
0x.2P4        //十六进制 0.2 乘以十进制 2 的 4 次方
0x1p2         //十六进制 1 乘以十进制 2 的 2 次方
0xp2          //错误，小数部分不能省略
0x2p0x2       //错误，指数不能是十六进制
```

2. 布尔常量

布尔类型的常量只有两个，一个是 true，另一个是 false。true 表示真，false 表示假，布尔

类型一般用于逻辑判断。

3. 字符常量

字符常量是程序中的一个字符，仓颉字符常量有 3 种形式，分别是单个字符、转义字符和通用字符。字符常量的表示方法是单引号引起来的一个字符。

单个普通字符通过单引号引起来表示一个字符常量，如'a'和'n'等。

对于一些特殊含义的字符，则需要用转义字符进行说明，如“\n”表示换行字符，“\t”表示 Tab 字符，“\'”表示一个单引号字符。

通用字符是以\u 开头说明的字符，在其后面为一对花括号，花括号中间为 1～8 位十六进制数，该数的值对应字符的 Unicode 编码，Unicode 编码是一种全球统一的字符编码，各国各种语言的每个字符都对应一个唯一编码。例如“\u{9889}”表示汉字“颉”。

4. 字符串常量

字符串常量最普遍的形式是用双引号引起来的多个字符，如"Hello World!"。在仓颉语言中，字符串常量分为单行字符串、多行字符串、多行原始字符串。

单行字符串常量的内容在一对双引号之内，双引号中的内容可以是任意个字符，在程序中，单行字符串常量只能写在同一行，不能跨越多行，如果字符串中有特殊字符，则可以使用转义字符说明，示例代码如下：

```
"Hello Cangjie Lang"
"\"Hello Cangjie Lang\""
"Hello Cangjie Lang\n"
```

多行字符串常量以 3 个双引号开头，并以 3 个双引号结尾。开头的 3 个双引号单独占用一行，所以需要换行，否则会出现编译错误。字符串的内容从开头的 3 个双引号的下一行的第 1 个字符开始，到结尾的三个双引号之前的一个字符为止，之间的内容可以是任意数量的字符，如果字符串中包含特殊字符，也可以使用转义符说明，和单行字符串不同的是多行字符串可以跨越多行，示例代码如下：

```
"""
Hello
 Cangjie Language"""

"""
\"Hello\n Cangjie Language\""""
```

多行原始字符串可以使字符串保持在代码中的原始样子，它以一个或多个井号（#）加上一个双引号开始，以一个双引号加上和开始位置相同个数的井号（#）结束。在输出多行原始字符串时，它的内容会保持原样，转义字符在多行原始字符串中不起作用，同样保持原样，示例代码如下：

```
#"
\\Please select 1 or 2,then press Enter\\
 1. input
 2. output"#
```

上面示例多行字符串中的\\在输出时会原样输出，输出的是两个\\，而不是一个。

3.1.2　变量

变量顾名思义是可变的量，变量在程序中一般用来存放可变的数据。变量具有变量名、变量类型、变量值。变量名是程序定义的可以赋值的标识符，变量类型限定了变量中存储的值的类型，变量值则是变量中存储的当前内容。

在仓颉语言中定义变量的一般形式如下：

```
修饰词　变量名:类型 = 初始值
```

仓颉语言是强类型语言，要求定义变量时必须确定变量的类型，但是在仓颉语言中允许类型自动推定，如果可以确定给定的初始值类型，则定义变量时可以省略类型，此时定义变量的基本格式如下：

```
修饰词　变量名 = 初始值
```

变量定义时一般需要初始化，即赋初始值，但是也允许在变量定义时不赋初始值，此时定义变量的基本格式如下：

```
修饰词　变量名:类型
```

在变量定义中可以使用修饰词 let 或 var，采用 let 定义的变量初始化后，其值不能再改写，因此也可以称为不可变变量。通过 var 修饰的变量，可以在程序中随时改变变量的值。

备注：let 可以理解成“假设”“让”，let 修饰的变量初始化后不能再改变。实际上仓颉语言中的变量不必在定义变量时进行初始化，作为局部 let 变量，第 1 次赋值即为初始化。var 是英文单词 variable 的缩写，即变量，采用 var 修饰的变量可以随时改变值。

变量名必须是仓颉语言的合法标识符，标识符在仓颉语言中又分为普通标识符和原始标识符。普通标识符是除了仓颉关键字以外的，由字母开头，后接任意长度的字母、数字或下画线组合而成的连续字符。普通标识符也可以用若干下画线开头，后接 1 个以字母，再接任意多个字母、数字或下画线。原始标识符是在普通标识符的外面加上一对反引号（`），反引号内的内容允许和仓颉关键字重复。原始标识符主要用在需要将仓颉关键字作为标识符使用的场景，一般情况下不使用原始标识符，示例代码如下：

```
name              //合法
Name              //合法，大小写敏感，name 和 Name 是两个不同的标识符
s_name            //合法
s_name_1          //合法
```

```
    `for`               //合法，原始标识符

    6age                //不合法，不能以数字开头
    _6age               //不合法，下画线后必须跟字母
```

初始值是在定义变量时变量所获得的一个值，采用 let 定义变量时，一般在定义时直接进行变量初始化。

下面是几个定义变量的例子：

```
//ch03/proj0301/src/test.cj
main() {
    let x: Int64 = 6                    //定义不可变变量 x，64 位整型，赋值为 6
    var y: Float64 = 3.6                //定义变量 y，64 位浮点类型，赋值为 3.6
    let b: Bool                         //定义布尔变量 b
    b = true                            //变量 b 的初始化值为 true，第一次赋值即用初始化值

    var first_name: String              //定义字符串变量 first_name
    first_name = "zhang"                //将变量 first_name 赋值为 zhang
    var age = 18                        //自动将 age 推定为 Int64 类型
    var `for` = "for 是循环关键字"      //原始标识符

    var your name = "zhang"             //错误，标识符名中间不能有空格
    var _name = "zhang"                 //错误，标识符名中的下画线不能是第 1 个字符
    var 6name = "zhang"                 //错误，标识符名中的数字不能是第 1 个字符
}
```

另外，变量定义中的类型可以是仓颉语言的内置类型，也可以是用户自定义类型，自定义类型将在后续内容中具体说明。

3.2 数据类型

数据类型可以帮助程序员在给定范围内更好地使用数据，仓颉语言支持的基本数据类型有整数类型、浮点类型、布尔类型、字符类型、字符串类型、元组类型、区间类型、Unit 类型、Nothing 类型。

1. 整数类型

整数类型所能表示的数值范围是数学中整数的子集，仓颉语言中的整数类型又分为有符号（Signed）整数类型和无符号（Unsigned）整数类型。

有符号整数类型包括的类型名称有 Int8、Int16、Int32、Int64 和 IntNative，这些类型名称中的数字表示对应类型量所占的内存位数，如 Int32 表示占用 32 位的整数类型，这里的位数是指二进制位，每位是 1bit。IntNative 表示和平台相关的有符号整数值的类型。

无符号整数类型包括 UInt8、UInt16、UInt32、UInt64 和 UIntNative。无符号整数类型主要用于表示正整数。

由于不同的类型所占内存空间的大小不尽相同，因此它们能表示的整数的范围也不同。表 3-1 给出了不同的整数类型表示的整数范围。

表 3-1　整数类型表示的整数范围

类　　型	表示的整数范围
Int8	−128～127，即-2^7～2^7-1
Int16	−32 768～32 767，即-2^{15}～$2^{15}-1$
Int32	−2 147 483 648～2 147 483 647，即-2^{31}～$2^{31}-1$
Int64	−9 223 372 036 854 775 808～9 223 372 036 854 775 807，即-2^{63}～$2^{63}-1$
IntNative	与平台相关
UInt8	0～255，即 0～2^8-1
UInt16	0～65 535，即 0～$2^{16}-1$
UInt32	0～4 294 967 295，即 0～$2^{32}-1$
UInt64	0～18 446 744 073 709 551 615，即 0～$2^{64}-1$
UIntNative	与平台相关

在开发仓颉程序过程中，选择什么数据类型一般取决于解决问题需要的数据范围。仓颉语言在没有类型显式说明的情况下，默认将整数推断为 Int64 类型，Int64 类型是所有整数类型中占用空间最大的类型，其表示范围也是最大的。使用 Int64 整数类型可以避免不必要的类型转换，同时提供更大的运算空间。

备注：Int 是 Integer 的缩写，即整型。仓颉提供的整数类型中可以通过类型名看出数据存储的空间大小，如 Int64 表示整型数，采用 64 位存储，即采用 8 字节存储。但仓颉的整数类型不是一种类型，如 Int32 和 Int64 是两种不同的类型，如果它们之间进行运算，则仍然需要转换。

2. 浮点类型

浮点类型所能表示的数值是数学中实数的子集，仓颉语言中的浮点类型包括 Float16、Float32 和 Float64。Float16 用 16 位表示一个浮点数，Float32 用 32 位表示一个浮点数，Float64 用 64 位表示一个浮点数。Float32 和 Float64 分别对应 IEEE 754 标准中的单精度和双精度格式。浮点类型表示的数值范围如表 3-2 所示。

表 3-2　浮点类型表示的数值范围

类　　型	表示的数值范围
Float32	-3.40×10^{38}～$+3.40\times10^{38}$，注意，并不能精确表示范围内的所有实数
Float64	-1.79×10^{308}～$+1.79\times10^{308}$，注意，并不能精确表示范围内的所有实数

Float32 表示的数字精度对应十进制一般为小数点后 6 位，Float64 的精度约为小数点后 15

位。尽管浮点类型具有表示范围，但其并不能完全精确地表示其范围内的所有实数，实际上绝大部分数学中的实数不能被浮点数精确地表示。开发中可以根据精度和范围的需要，选择合适的浮点类型进行数据表示，但也要注意计算过程中可能会出现误差叠加。为了更加精确，开发中首选精度和范围都更高的 Float64 类型，但和 Float32 类型相比会消耗更多存储空间。

3. 布尔类型

布尔类型使用 Bool 表示，用来表示逻辑中的真和假，真即 true，假即 false。

4. 字符类型

字符用于表示单个字符，字符类型使用 Char 表示。仓颉语言中字符采用的是 Unicode 编码，一个字符占用两字节，Char 类型可以表示 Unicode 字符集中的所有字符。

5. 字符串类型

字符串是由若干字符构成的一串字符，字符串中字符的个数可以是 0 个或多个，当为 0 个时表示为空字符串。字符串类型使用 String 表示，字符串常用来表示文本数据，如姓名等。

6. 元组类型

元组（Tuple）是将多种类型组合在一起而构成的一种新类型，元组类似于向量，元组中各个分量的类型可以一样，也可以不一样，元组类型使用（Type1, Type2,..., TypeN）表示，其中 Type1 到 TypeN 可以是任意类型。元组至少包含两个元素，每个元素可以对应一种类型，具有两个元素的元组称为二元元组，如（Int32, Float64）表示一个二元元组类型。具有 3 个元素的元组称为三元元组，如（String, Int32, Float64）表示一个三元元组类型。具有 3 个以上元素的元组则可以统一称为多元元组。

元组类型是不可变类型，即一旦定义了一个元组，它的内容只能被读取，不能被改写。元组常量采用小括号括起来的多个分量表示，分量之间用逗号分隔。访问元组的分量采用中括号加下标的形式。

通过元组定义的元组变量是引用类型，它可以引用不同的元组常量，但是不能通过它修改所引用的元组元素内容。

示例代码如下：

```
//ch03/proj0302/src/test.cj
main() {
    var X: (String, Int32) = ("张三", 18)      //定义元组变量 X 并初始化
    var f:Bool = false                          //定义 f
    var Y = (f, true)                           //省略了类型说明，可以推定为（Bool,Bool）
    println(X[0])                               //输出张三
    f = true
    println(Y[0])                               //输出 false，f 的值变了，但元组分量没变
    X[1] = 19                                   //错误,元组元素不可写
    X = ("李四", 19)                            //X 可以引用新的元组
```

```
    println(X[0])                                    //输出李四
    println(Y[2])                                    //错误，下标 2 越界
}
```

7. 区间类型

区间表示一个范围，仓颉的区间类型用于表示指定范围内拥有固定步长的整数序列。区间类型本质上是泛型类 Range<T>，当 T 被具体化为不同的整数类型时，会得到不同的区间类型，常用 Range<Int64>表示整数区间。

每个区间类型的实例都会包含 start、end 和 step 3 个值，其中，start 和 end 分别表示区间序列的起始值和结束值，step 表示序列中相邻两个元素之间的差值，即步长。start 和 end 的类型要求相同，即 T 被具体化的类型，step 的类型是 Int64 类型。

区间有两种形式，分别是左闭右开和左闭右闭。左闭右开区间的表示格式如下：

```
start..end:step
```

表示一个从 start 开始，以 step 为步长，到 end（不包含 end）为止的所有的整数序列。左闭右闭区间的表示格式如下：

```
start..=end:step
```

表示一个从 start 开始，以 step 为步长，到 end（包含 end）为止的所有的整数序列，示例代码如下：

```
let n = 10
let r1 = 0..10:1        //r1 为左闭右开区间，包括 0、1、2、3、4、5、6、7、8、9
let r2 = 0..=n:1        //r2 为左闭右闭区间，包括 0、1、2、3、4、5、6、7、8、9、10
let r3 = n..0:-2        //r3 包括 10、8、6、4、2
let r4 = n..=0:-2       //r4 包括 10、8、6、4、2、0
let r5 = n..0:-3        //r5 包括 10、7、4、1
let r6 = n..=0:-3       //r6 包括 10、7、4、1，尽管有=号，但是根据步长到不了 0
```

需要注意，step 的值不能为 0，当 step 为 1 时，可以省略。另外，区间也可以是空区间，即不包含任何元素的空序列，示例代码如下：

```
let r7 = 0..6           //默认步长为 1，包含 0、1、2、3、4、5
let r8 = 0..10:0        //错误，步长不能是 0
let r9 = 10..0:1        //空区间
let r10 = 0..=10:-1     //空区间
let r11 = 0..=10:15     //非空区间，r11 只包含 0
let r12 = 0..=0:-1      //非空区间，r12 只包含 0
```

8. Unit 类型

Unit 是仓颉语言中定义的一种特殊类型，可以翻译成单元类型，一般直接说成 Unit 类型。Unit 类型主要应用于只关心表达式的运算作用，而不关心值的表达式。例如，print 函数、赋值表达式、复合赋值表达式、自增和自减表达式、循环表达式等，一般不需要使用它们的返回结

果，而只是关心它们的执行过程，因此它们的类型都是 Unit 类型。

Unit 类型只有一个值，其值规定为( )。Unit 类型量支持赋值、判断相等和判断不等，除此之外，Unit 类型不支持其他操作，示例代码如下：

```
let u1 = ()                 //定义 u1，并赋值
var u2:Unit                 //定义 u2
u2 = u1                     //将 u2 赋值为 u1
u2 = println("good")        //u2 值为 ()
```

9. Nothing 类型

Nothing 是仓颉语言定义的一种特殊的类型，可以翻译成空类型，一般直接说成 Nothing 类型。Nothing 类型不包含任何值，Nothing 类型是所有类型的子类型。break、continue、return 和 throw 表达式的类型是 Nothing，程序执行到这些表达式时，它们之后的代码将不会被执行。

需要注意的是，目前仓颉编译器还不允许在使用类型时显式地使用 Nothing 类型。

备注：仓颉是一种强类型语言，要求所有的数据都要有确定的类型，并且类型确定后不能改变，但类型可以转换。C/C++、Java、Go 都是强类型语言，也需要在定义变量时确定类型。Python 语言则不同，变量标识符可以被赋值为不同的类型。仓颉语言采用了类型推定原则，在编译器可以推定类型时，可以省略显式的类型说明。

3.3 运算符

运算符是仓颉语言定义的用于执行特定的数学或逻辑等操作的符号。仓颉语言内置了丰富的运算符，这些运算符按类别可分为算术运算符、关系运算符、逻辑运算符等。

3.3.1 算术运算符

算术运算符一般用于算术运算，仓颉语言中算术操作符包括负号（-）、加法（+）、减法（-）、乘法（*）、除法（/）、取模（%）、幂运算（**），其中负号为一元运算符，即只需一个操作数，其他均为二元运算符，需要两个操作数。二元算术运算符在不进行重载的情况下，要求左右操作数具有相同的数值类型，示例代码如下：

```
var  a:Int32 = 100
var  b:Int32 = 2
var  c:Int64 = 600
var  r:Int32
var  f:Float32
r = a+b             //r 的值为 102
r = -a-b            //r 的值为-102
r = a/b             //r 的值为 50
r = b/a             //r 的值为 0，整数相除表示整除
```

```
f = 3.6/3.0          //f 的值为 1.2
r = a*b              //r 的值为 200
r = 100%3            //r 的值为 1，100 除以 3 的余数为 1
r = a**b;            //r 的值为 10000

r = a+c              //错误，a 和 c 类型不一致
f = 3.6/3            //错误，3.6 和 3 类型不一致
r = b**a             //错误，结果溢出
```

备注：仓颉语言提供的数值类型较多，不同的数值类型在程序中被认为是不同的类型，如 Int32 和 Int64 是不同的类型，所以不能直接进行算术运算。当然，开发者可以对算术运算符进行重载，这样可以使算术运算符进行更多的运算。

3.3.2　关系运算符

关系运算符主要用于比较量的大小，仓颉语言中关系运算符包括小于（<）、大于（>）、小于或等于（<=）、大于或等于（>=）、相等（==）、不等（!=）。关系运算符要求左右操作数是相同的数据类型，关系运算的结果是 Bool 类型，如果被比较的两个量关系成立，则关系表达式的结果为 true，否则结果为 false。

对于数值类型，可以进行比较大小、等或不等，对于一些其他类型有的只能进行相等或不等比较，示例代码如下：

```
//ch03/proj0303/src/test.cj
main() {
    var a: Int32 = 100
    var b: Int32 = 2
    var t1: Bool
    var t2: Bool
    t1 = a >= b                  //t1 的值为 true
    t2 = false                   //t2 的值为 false
    if (t1 == t2) {              //判断 t1 和 t2 是否相等，t1==t2 的值为 false
        println("t1 和 t2 相等")
    } else {
        println("t1 和 t2 不相等")
    }
    let r1 = 'b' > 'a'           //字符可以比较大小，即比较 Unicode 编码值
    let r2 = '你' > '我'          //字符可以比较大小，即比较 Unicode 编码值
    let r3 = "ab" > "aa"         //字符串可以比较大小，比较从左起第 1 个不同的字符的大小
    let r4 = t1 == t2            //r4 的值为 false

    let r5 = t1 > t2             //错误，Bool 类型不能比较大小
    let r6 = 100< 206.6          //错误，不同类型不能比较大小
}
```

3.3.3 逻辑运算符

逻辑运算符可以进行真或假逻辑运算，常用于判断复合条件的真假。逻辑运算符包括逻辑非（!）、逻辑与（&&）、逻辑或（||），逻辑运算符的运算对象是 Bool 类型的逻辑值。逻辑非是单目运算符，逻辑与和逻辑或是双目运算符。

逻辑非表示对逻辑值进行取反运算，即真取反后为假，假取反后为真。

逻辑与是当参与运算的对象都为真时，运算结果为真，否则运算结果为假。

逻辑或是当参与运算的对象有一个为真时，运算结果为真，否则运算结果为假。

示例代码如下：

```
//ch03/proj0304/src/test.cj
main() {
    var a: Int32 = 70
    var b: Int32 = 2
    var t1: Bool
    var t2: Bool
    var t3: Bool
    t1 = a > b                          //t1 为 true
    t2 = !t1                            //t2 为 false
    t3 = t1 && t2                       //t3 为 false
    t3 = t1 || t2                       //t3 为 true
    if (a % 5 == 0 && a % 7 == 0) {     //判断 5 和 7 的公倍数
        println("a 是 5 和 7 的公倍数")
    }
}
```

3.3.4 其他运算符

1. 自增自减运算符

自增运算符为++，其功能是对运算对象自增 1；自减运算符为--，其功能是对运算对象自减 1。在仓颉语言中自增自减运算符只能用在运算对象的后面，即后增或后减，没有前增和前减运算符。自增和自减的运算结果是 Unit 类型，因此一般自增和自减都是独立的表达式，而不用于其他表达式中，示例代码如下：

```
var x:Int32 = 1
var y:Int32
x++                     //x 自加 1
--x                     //错误，仓颉语言中不存在前置自减运算符
y = x++                 //错误，Unit 类不能为 Int32 类型赋值
```

2. 位运算符

位运算是按照二进制位对运算对象进行运算的，仓颉语言支持的位运算符有按位左移（<<）、

按位右移（>>）、按位与（&）、按位异或（^）、按位或（|）。这些位运算符均为双目运算符，示例代码如下：

```
//ch03/proj0305/src/test.cj
main() {
    var x: Int32 = 5   //5 对应二进制 0101，左侧高位省略 28 个 0
    var y: Int32 = 15  //15 对应二进制 1111，左侧高位省略 28 个 0
    var z: Int32
    x = x << 2         //左移两位，右侧补 0，高位舍弃，得到 010100，对应十进制 20
    y = y >> 1         //右移 1 位，左侧补 0，右侧舍弃，得到 0111，对应十进制 7
    z = x & y          //z 二进制为 00100，对应十进制为 4
    z = x ^ y          //z 二进制为 10011，对应十进制为 19
    z = x | y          //z 二进制为 10111，对应十进制为 23
    println(z)         //输出 z 为 23
    z >> 2             //表达式的值是 5，位运算符不改变运算对象的值
    println(z)         //输出 z 仍然为 23
}
```

3. 赋值运算符

赋值运算的功能是将运算符右侧的值赋给左侧，赋值运算符用=表示，基本用法如下：

```
变量 = 表达式
```

赋值运算符可以和算术运算符、关系运算符、位运算符等双目运算符组合成复合赋值运算符，复合赋值运算符包括**=、*=、/=、%=、+=、-=、<<=、>>=、&=、^=、|=、&&=、||=。它们的用法均为如下格式：

```
变量 复合赋值运算符 表达式
```

等价于

```
变量 = 变量 复合赋值运算符 （表达式）
```

示例代码如下：

```
var x:Int32 = 6
var y:Int32 = 9
var z:Int32
z = x                  //赋值运算符的基本用法
z **= 2                //等价于 z=z**2,即 z 的 2 次方
z *= y                 //等价于 z=z*y
z /= y                 //等价于 z=z/y
z += x*y               //等价于 z=z+x*y
z *= x-y               //等价于 z=z*(x-y)
```

4. 类型判断运算符

在开发过程中，经常会进行类型判断，仓颉语言的类型判断运算符为 is，其基本格式如下：

```
表达式 is 类型
```

类型判断运算表达式的返回结果为 Bool 类型，即结果为 true 或 false。实践中常根据类型做出相应的处理，示例代码如下：

```
var x = 6           //默认整数是 Int64 类型
if(x is Int32)      //is 表达式返回的是 Bool 类型，这里返回的值是 false
{
    //进行相应处理
}
```

5. 更多运算符

除了前面介绍的运算符外，仓颉语言还支持一些别的运算符，如成员运算符（.）、问号（?）运算符等。运算符是伴随着数据和数据类型进行运算的，开发者可以在后续的学习中掌握更多的运算符。

另外，仓颉语言支持运算符重载，通过重载可以扩展现有运算符的运算能力，使运算符支持更多数据类型和运算功能。