# Linux安装JDK

## 下载

<https://www.oracle.com/cn/java/technologies/downloads/#java21>

## 解压

|  |
| --- |
| tar -zxvf /development/jdk-21\_linux-x64\_bin.tar.gz -C /development |

查找解压位置：

|  |
| --- |
| find / -name jdk-21.0.4 |

## 配置环境变量

|  |
| --- |
| vim /etc/profile |

|  |
| --- |
| export JAVA\_HOME=/development/jdk-21.0.4 export PATH=$PATH:$JAVA\_HOME/bin export CLASSPATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar |

刷新环境变量：

|  |
| --- |
| source /etc/profile |

# 编译运行java文件

## 编译文件

|  |
| --- |
| javac 源文件 [-d 编译文件输出路径] [-encoding utf-8]  # 编译文件中的多个java源文件  javac @sourcesList\_file [-d 输出路径] [-encoding utf-8] |

|  |
| --- |
| # 编译文件  javac Test.java  # 编译当前目录下src文件夹下的文件  javac src/\*.java |

|  |
| --- |
| javaList.txt 是列出要编译的java文件列表的文件，各个java源文件之间用回车符进行分割，如：  src\com\northcastle\Javac02A.java  src\com\northcastle\Javac02B.java  src\com\utile\JavaUtile.java  # 编译  javac @javaList.txt -d target -encoding utf-8 |

## 运行

|  |
| --- |
| # 执行class文件，class后面的参数是main函数的参数  java [-options] class [args...]  # 执行jar文件  java [-options] -jar jarfile [args...] |

|  |
| --- |
| # 使用java命令运行.class文件，注意不要带.class后缀  java Test  # 带vm参数执行  java -XX:+PrintCompilation Test  # 运行指定目录下的class文件  java -classpath C:\java\DemoClasses Test  # 运行指定目录下的jar包  java -classpath C:\java\myclasses.jar |

# 打jar包

## java命令方式

### 只含有一个类的jar包

用记事本写一个HelloWorld.java的文件

|  |
| --- |
| public class HelloWorld {  public static void main(String[] args) {  System.out.println("Hello, Native World!");  while(true){}  }  } |

编译这个文件

|  |
| --- |
| javac HelloWorld.java |

将编译后的HelloWorld.class文件打成jar包

|  |
| --- |
| jar -cvf HelloWorld.jar HelloWorld.class |

* c表示要创建一个新的jar包
* v表示创建的过程中在控制台输出创建过程的一些信息
* f表示给生成的jar包命名

运行jar包

|  |
| --- |
| java -jar HelloWorld.jar |

报错：HelloWorld.jar中没有主清单属性

添加Main-Class属性

用压缩软件解压HelloWorld.jar，会发现里面多了一个META-INF文件夹，里面有一个MENIFEST.MF的文件，用记事本打开

写入 Main-Class: HelloWorld（注意冒号后面有一个空格，最后一行空行）

|  |
| --- |
| Manifest-Version: 1.0  Created-By: 21.0.5 (Oracle Corporation)  Main-Class: HelloWorld |

再压缩成zip，拓展名修改为jar

运行jar包

|  |
| --- |
| java -jar HelloWorld.jar |

Hello, Native World!

### 含有两个类的jar包

|  |
| --- |
| public class Tom{  public static void speak(){  System.out.println("hello");  }  } |

|  |
| --- |
| public class HelloWorld {  public static void main(String[] args) {  Tom.speak();  while(true){}  }  } |

编译：

|  |
| --- |
| javac HelloWorld.java |

编译HelloWorld.java的同时，还会编译Tom.java，因为HelloWorld.java中调用了Tom.java

打jar包：

创建一个META-INF文件夹，META-INF文件夹里再创建MANIFEST.MF文件。

MANIFEST.MF文件写入下面内容（注意冒号后面有一个空格，最后一行空行）

|  |
| --- |
| Manifest-Version: 1.0  Created-By: 21.0.5 (Oracle Corporation)  Main-Class: HelloWorld |

打jar包：

|  |
| --- |
| jar -cvfm HelloWorld.jar META-INF\MANIFEST.MF HelloWorld.class Tom.class |

* c表示要创建一个新的jar包
* v表示创建的过程中在控制台输出创建过程的一些信息
* f表示给生成的jar包命名
* m表示要定义MENIFEST文件

### 有目录结构的jar包

目录结构：

|  |
| --- |
| META-INF  MANIFEST.MF  com  　Tom.class  HelloWorld.class |

|  |
| --- |
| package com;  public class Tom{  public static void speak(){  System.out.println("hello");  }  } |

|  |
| --- |
| import com.Tom;  public class HelloWorld {  public static void main(String[] args) {  Tom.speak();  while(true){}  }  } |

编译：

|  |
| --- |
| javac HelloWorld.java -d target |

会自动编译com/ Tom.java，因为HelloWorld.java引用了

把META-INF文件夹放到target 文件夹下

打jar包：

|  |
| --- |
| jar -cvfm HelloWorld.jar META-INF\MANIFEST.MF \* |

* \*：表示把当前目录下所有文件都打在jar包里

### 将引用了jar包的java文件打成jar包

目录：

|  |
| --- |
| META-INF  MANIFEST.MF  Tom.java  HelloWorld.java |

|  |
| --- |
| public class Tom{  public static void speak(){  System.out.println("hello");  }  } |

|  |
| --- |
| public class HelloWorld {  public static void main(String[] args) {  Tom.speak();  while(true){}  }  } |

编译：

|  |
| --- |
| javac Tom.java |

将打Tom成jar包

|  |
| --- |
| jar -cvf Tom.jar Tom.class |

编译：

|  |
| --- |
| javac -cp Tom.jar HelloWorld.java |

* -cp 表示 -classpath，指的是把Tom.jar加入classpath路径下

将HelloWorld.class打成jar包：

|  |
| --- |
| jar -cvfm HelloWorld.jar META-INF\MANIFEST.MF HelloWorld.class |

此时这个jar包还不能运行

MANIFEST.MF文件内容（注意冒号后的空格和最后的空行）：

|  |
| --- |
| Manifest-Version: 1.0  Created-By: 21.0.5 (Oracle Corporation)  Main-Class: HelloWorld  Class-Path: Tom.jar |

* Class-Path，路径指向你需要的所有jar包。多个jar包，中间用空格隔开

重新打成jar包，目录情况（忽略运行时无关目录）：

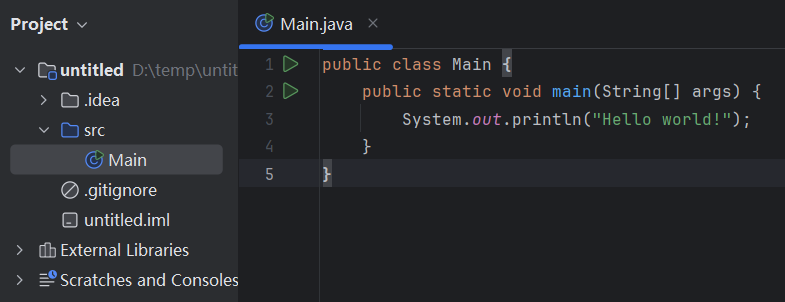
|  |
| --- |
| Tom.jar  HelloWorld.jar |

缺点：如果把Tom.jar移开，就不能运行java.lang.NoClassDefFoundError: Tom

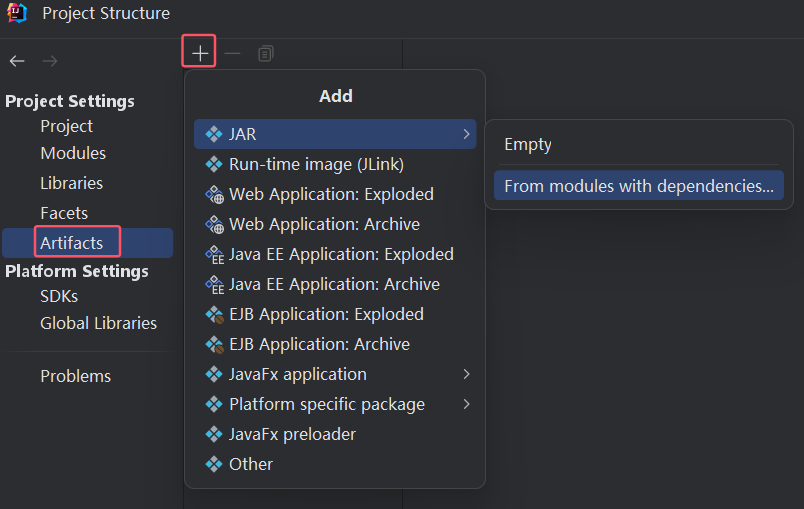
## maven方式

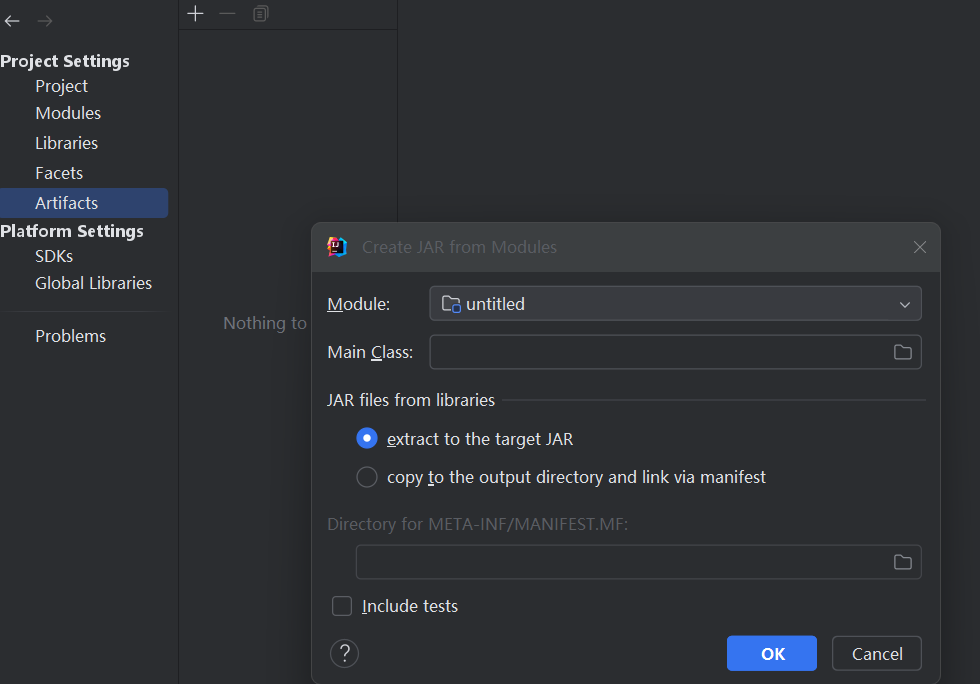
使用Maven的install命令即可

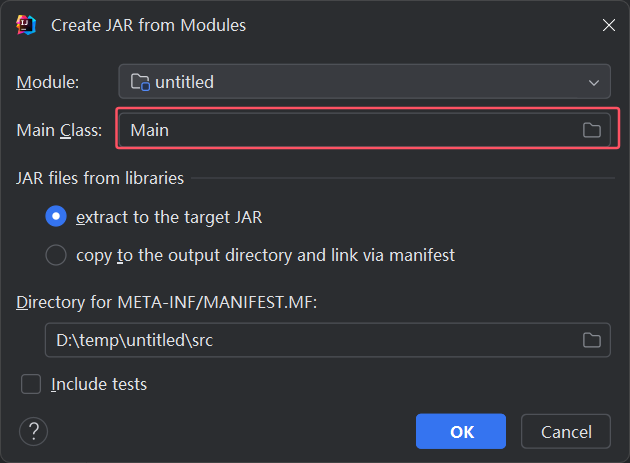
## 【IntelliJ IDEA】方式



随便编写一个主方法







选择主方法，点击 " OK " 后 , IntelliJ IDEA 会在 src 目录下生成 META-INF/MANIFEST.MF 文件

