

## 1 实验一

### 1.1 实验内容

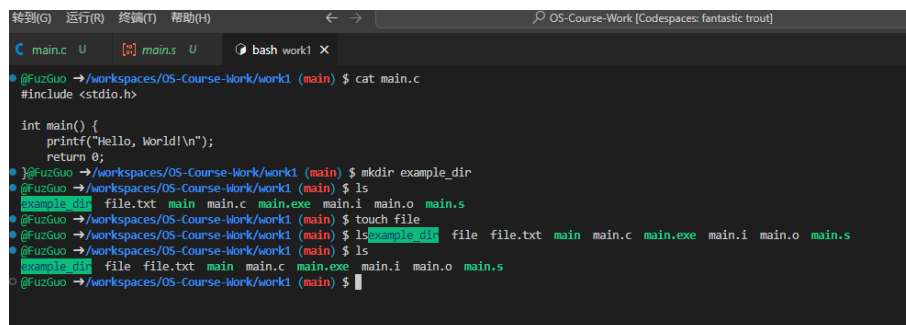
- 1、熟悉 Linux 常用命令。
- 2、熟悉 vi 编辑器的使用。
- 3、熟悉 gedit 编辑器的使用。
- 4、学会使用 gcc 编译链接一个简单的 c 语言程序。

### 1.2 实验步骤

#### 1.2.1 Linux 常用命令

命令	描述	示例用法
cat	显示文件内容	cat file.txt 显示文件内容
chmod	改变文件权限	chmod 755 script.sh 设置文件权限为可执行
chown	改变文件所有者	chown user:group file.txt 更改所有者和组
cp	复制文件或目录	cp source.txt dest.txt 复制文件
mkdir	创建新目录	mkdir new_folder 创建新目录
mv	移动或重命名文件或目录	mv oldname.txt newname.txt 重命名文件
rm	删除文件或目录	rm -r dir 删除目录及其内容
rmdir	删除空目录	rmdir empty_dir 删除空目录
touch	创建空文件或更新时间戳	touch file.txt 创建空文件

表 1: 文件管理命令



```
转到(G) 运行(R) 终端(T) 帮助(H) OS-Course-Work [Codespaces: fantastic trout]
main.c U [main.s U] bash work1 X
@FuzGuo → /workspaces/OS-Course-Work/world1 (main) $ cat main.c
#include <stdio.h>

int main() {
    printf("Hello, World!\n");
    return 0;
}
@FuzGuo → /workspaces/OS-Course-Work/world1 (main) $ mkdir example_dir
@FuzGuo → /workspaces/OS-Course-Work/world1 (main) $ ls
example_dir  file.txt  main  main.c  main.exe  main.i  main.o  main.s
@FuzGuo → /workspaces/OS-Course-Work/world1 (main) $ touch file
@FuzGuo → /workspaces/OS-Course-Work/world1 (main) $ ls
example_dir  file  file.txt  main  main.c  main.exe  main.i  main.o  main.s
@FuzGuo → /workspaces/OS-Course-Work/world1 (main) $
```

图 1: 执行结果

命令	描述	示例用法
<code>cd</code>	切换当前工作目录	<code>cd /home/user</code> 切换到指定目录
<code>ls</code>	列出目录内容	<code>ls -l</code> 以长格式列出文件
<code>pwd</code>	显示当前工作目录路径	<code>pwd</code> 输出 <code>/home/user</code>
<code>tree</code>	以树状结构显示目录内容（需安装）	<code>tree dir</code> 显示目录树

表 2: 文件和目录管理命令

```

转到(G) 运行(R) 终端(T) 帮助(H)
main.c U [main.s] U bash workspaces X
@FuzGuo →/workspaces/OS-Course-Work/work1 (main) $ ls -l
total 76
drwxrwxrwx+ 2 codespace codespace 4096 Feb 28 14:49 example_dir
-rw-rw-rw- 1 codespace codespace 0 Feb 28 14:49 file
-rw-rw-rw- 1 codespace codespace 35 Feb 28 14:44 file.txt
-rwxrwxrwx 1 codespace codespace 16696 Feb 28 14:18 main
-rw-rw-rw- 1 codespace codespace 79 Feb 28 13:54 main.c
-rwxrwxrwx 1 codespace codespace 16696 Feb 28 14:09 main.exe
-rw-rw-rw- 1 codespace codespace 16331 Feb 28 13:57 main.i
-rw-rw-rw- 1 codespace codespace 1680 Feb 28 14:09 main.o
-rwxrwxrwx 1 codespace codespace 655 Feb 28 14:03 main.s
@FuzGuo →/workspaces/OS-Course-Work/work1 (main) $ pwd
/workspaces/OS-Course-Work/work1
@FuzGuo →/workspaces/OS-Course-Work/work1 (main) $ cd ..
@FuzGuo →/workspaces/OS-Course-Work (main) $ cd ..
@FuzGuo →/workspaces $ tree OS-Course-Work/
OS-Course-Work/
├── README.md
└── work1
    ├── example_dir
    ├── file
    ├── file.txt
    ├── main
    ├── main.c
    ├── main.exe
    ├── main.i
    └── main.o
  
```

图 2: 执行结果

命令	描述	示例用法
<code>df</code>	显示磁盘空间使用情况	<code>df -h</code> 以人类可读格式显示
<code>du</code>	显示目录或文件磁盘使用量	<code>du -sh dir</code> 显示目录总大小
<code>mount</code>	挂载文件系统	<code>mount /dev/sda1 /mnt</code> 挂载设备
<code>umount</code>	卸载文件系统	<code>umount /mnt</code> 卸载挂载点
<code>fdisk</code>	磁盘分区工具	<code>fdisk /dev/sda</code> 管理磁盘分区
<code>mkfs</code>	格式化文件系统	<code>mkfs.ext4 /dev/sda1</code> 格式化为 ext4

表 3: 磁盘管理命令

```

● @FuzGuo → /workspaces/OS-Course-Work/work1 (main) $ df
Filesystem      1K-blocks    Used Available Use% Mounted on
overlay          32847680 11558224  19595360  38% /
tmpfs             65536         0     65536    0% /dev
shm              65536         8     65528    1% /dev/shm
/dev/root        30298176 13296540  16985252  44% /vscode
/dev/sda1        46127956 18602008  25150372  43% /tmp
/dev/loop4       32847680 11558224  19595360  38% /workspaces
● @FuzGuo → /workspaces/OS-Course-Work/work1 (main) $ du
4    ./example_dir
80   .

```

图 3: 磁盘管理

命令	描述	示例用法
vi	强大的文本编辑器	vi file.txt 打开文件编辑
nano	简单易用的文本编辑器	nano file.txt 编辑文本文件
ed	行编辑器	ed file.txt 启动行编辑器
sed	流编辑器，用于文本处理	sed 's/old/new/' file.txt 替换文本
awk	文本处理和数据提取工具	awk '{print \$1}' file.txt 打印第一列

表 4: 文档编辑命令

命令	描述	示例用法
lftp	命令行 FTP 客户端	lftp ftp://user@host 连接 FTP 服务器
scp	通过 SSH 安全复制文件	scp file.txt user@host:/path 传输文件
rsync	高效文件同步工具	rsync -av source/ dest/ 同步目录
wget	从网络下载文件	wget http://example.com/file 下载文件
curl	数据传输工具，支持多种协议	curl -O http://example.com/file 下载文件

表 5: 文件传输命令

命令	描述	示例用法
adduser	添加新用户	adduser newuser 创建新用户
userdel	删除用户	userdel olduser 删除用户
passwd	修改用户密码	passwd user 修改用户密码
sudo	以超级用户权限执行命令	sudo apt update 以 root 权限更新包
shutdown	关闭或重启系统	shutdown -h now 立即关机
reboot	重启系统	reboot 重启计算机
top	显示系统进程和资源使用	top 查看实时进程
ps	显示当前进程状态	ps aux 列出所有进程
kill	终止进程	kill -9 1234 强制终止进程 ID 1234

表 6: 系统管理命令

```
[1]+ Stopped top
● @FuzGuo → /workspaces/OS-Course-Work/work1 (main) $ sudo adduser user2
Adding user `user2' ...
Adding new group `user2' (1001) ...
Adding new user `user2' (1001) with group `user2' ...
The home directory `/home/user2' already exists. Not copying from `/etc/skel'.
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for user2
Enter the new value, or press ENTER for the default
  Full Name []: 123
  Room Number []: 123
  Work Phone []: 123
  Home Phone []: 123
  Other []: 123
Is the information correct? [Y/n] Y
● @FuzGuo → /workspaces/OS-Course-Work/work1 (main) $ sudo userdel user2
○ @FuzGuo → /workspaces/OS-Course-Work/work1 (main) $
```

图 4: 运行结果

命令	描述	示例用法
tar	打包和解包文件	tar -cvf archive.tar dir/ 打包目录
zip	压缩文件到 ZIP 格式	zip archive.zip file.txt 压缩文件
unzip	解压 ZIP 文件	unzip archive.zip 解压 ZIP 文件
gzip	压缩文件到 GZ 格式	gzip file.txt 压缩为 file.txt.gz
gunzip	解压 GZ 文件	gunzip file.txt.gz 解压文件
bzip2	压缩文件到 BZ2 格式	bzip2 file.txt 压缩为 file.txt.bz2
bunzip2	解压 BZ2 文件	bunzip2 file.txt.bz2 解压文件

表 7: 备份压缩命令

### 1.2.2 vi 编辑器的使用

Vi 编辑器是 Linux 系统中一个功能强大且广泛使用的文本编辑工具，学会它的基本操作可以显著提高文件处理效率。

要使用 Vi 编辑器，首先需要掌握如何调用它，只需在终端输入“vi”后跟文件名，例如“vi test.txt”，即可打开指定文件，如果文件不存在，Vi 会创建一个新文件。

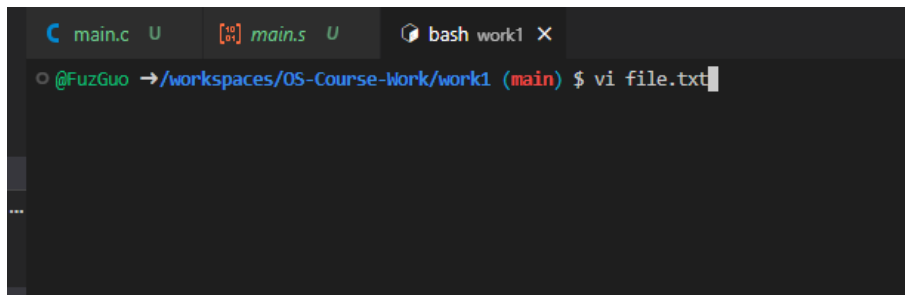


图 5: 使用 vi 新建并打开 file.txt

Vi 拥有三种主要运行模式：命令模式、插入模式和底线命令模式，理解这三种模式是熟练使用的基础。初次进入 Vi 时，默认处于命令模式，在此模式下可以执行移动光标、删除文本等操作；通过按下“i”键进入插入模式，用户可以输入和编辑文本；按下“Esc”键则返回命令模式，而输入“:”则切换到底线命令模式，用于保存或退出等操作。

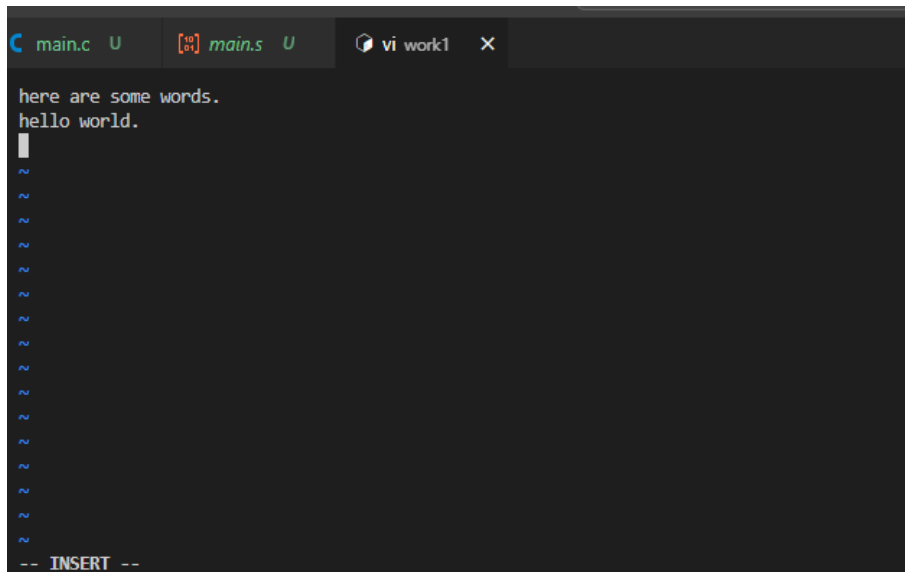


图 6: 输入一些示例内容

保存和退出文件是 Vi 中最常用的功能之一。在底线命令模式下，输入 “:w” 并按回车可以将当前修改保存到文件中，若想直接退出而不保存，可以输入 “:q”，如果文件已修改但不想保存，则需强制退出，使用 “:q!”。若要保持并退出，只需输入 “:wq” 即可。

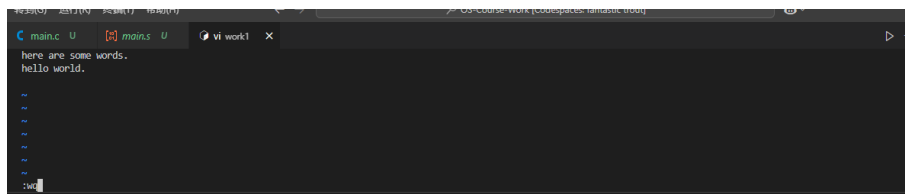


图 7: 保存退出

光标移动是 Vi 的核心操作，在命令模式下，可以使用 “h”、“j”、“k”、“l” 分别向左、下、上、右移动光标，也可以用 “0” 跳转到行首，“\$” 跳转到行尾，或者输入 “gg” 跳到文件开头，“G” 跳到文件末尾。

进入插入模式后，用户可以通过按 “i” 在光标前插入、“a” 在光标后插入、“o” 在当前行下新建一行插入等方式编辑文本。

文本内容的删除在命令模式下非常方便，按 “x” 可以删除光标所在字符，“dd” 删除整行，若误删了内容，可以立即按 “u” 撤销上一次修改，恢复到之前的状态。如果需要恢复被删除的内容，可以使用 “p” 将最后删除的文本粘贴到光标后。复制和粘贴操作同样简单，在命令模式下，“yy” 可以复制当前行，“p” 则粘贴到光标下方，通过组合数字，例如 “3yy”，可以一次性复制三行。关于行号，Vi 提供了方便的跳转功能，在底线命令模式输入 “: 行号”（如 “:5”）可以快速跳转到指定行，或者在命令模式下输入 “行号 G”（如 “5G”）实现相同效果。

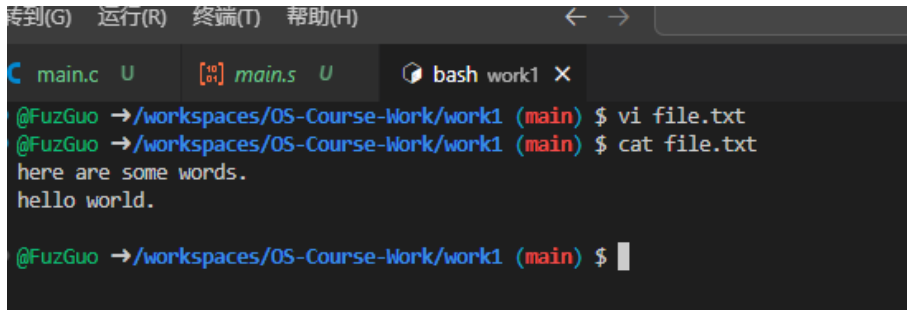


图 8: 查看保存结果

查找和替换是 Vi 的高级功能，在命令模式下，输入“/关键字”并按回车可以向下查找，按“?”则向上查找，使用“n”跳转到下一个匹配项。若要替换文本，可以在底线命令模式输入“:s/旧词/新词”替换当前行的第一个匹配项，若需替换整行所有匹配项，则用“:s/旧词/新词/g”，而全局替换则输入“:%s/旧词/新词/g”。

### 1.2.3 gedit 编辑器的使用

gedit 是 Linux 系统中一款轻量级且易于使用的文本编辑器，是 GNOME 桌面环境的默认编辑工具，它以简洁的界面和实用的功能受到广泛欢迎，尤其适合初学者和日常文本编辑需求。

要启动 gedit，只需在终端输入“gedit”后跟文件名，例如“gedit notes.txt”，即可打开指定文件进行编辑，如果文件不存在，gedit 会自动创建一个新文件，也可以通过图形界面的应用程序菜单找到并点击打开。与 Vi 编辑器不同，gedit 提供了一个直观的图形界面，用户无需记忆复杂的命令，所有的操作都可以通过菜单或快捷键轻松完成，例如保存文件只需点击“文件”菜单中的“保存”选项或按下“Ctrl+S”，关闭编辑器则可以通过“文件”菜单的“退出”或直接点击窗口的关闭按钮。

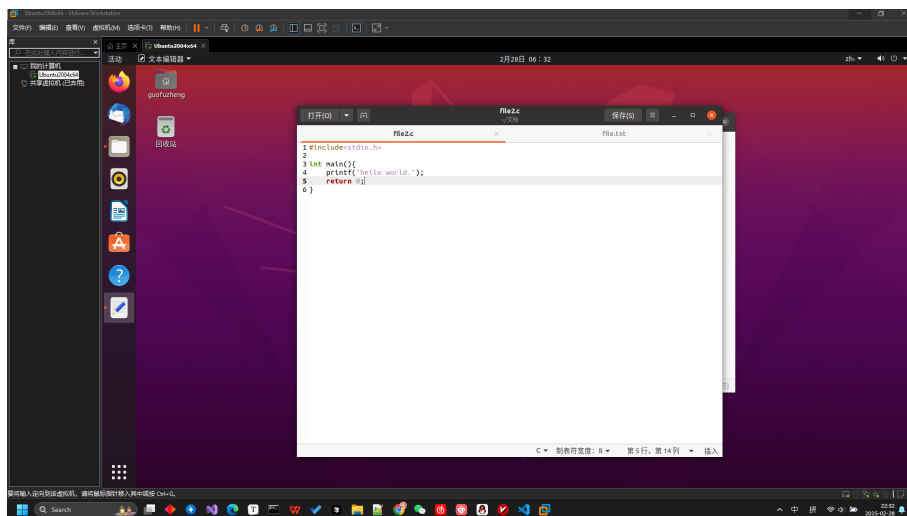


图 9: gedit 界面

在 gedit 中编辑文本非常简单，打开文件后即可直接输入或修改内容，光标可以通过鼠标点击或方向键移动到任意位置，支持常见的复制（Ctrl+C）、粘贴（Ctrl+V）和剪切（Ctrl+X）操作，这些功能与大多数图形化编辑器一致。如果需要撤销误操作，可以使用“编辑”菜单中的“撤销”或按“Ctrl+Z”，而恢复被撤销的内容则通过“重做”或“Ctrl+Y”实现。gedit 还支持文本的查找和替换功能，通过“搜索”菜单选择“查找”或按“Ctrl+F”打开查找栏，输入关键字后点击回车即可定位匹配项，若要替换文本，可以选择“替换”选项或按“Ctrl+H”，输入新旧内容后逐一替换或全部替换，非常适合快速调整文件内容。

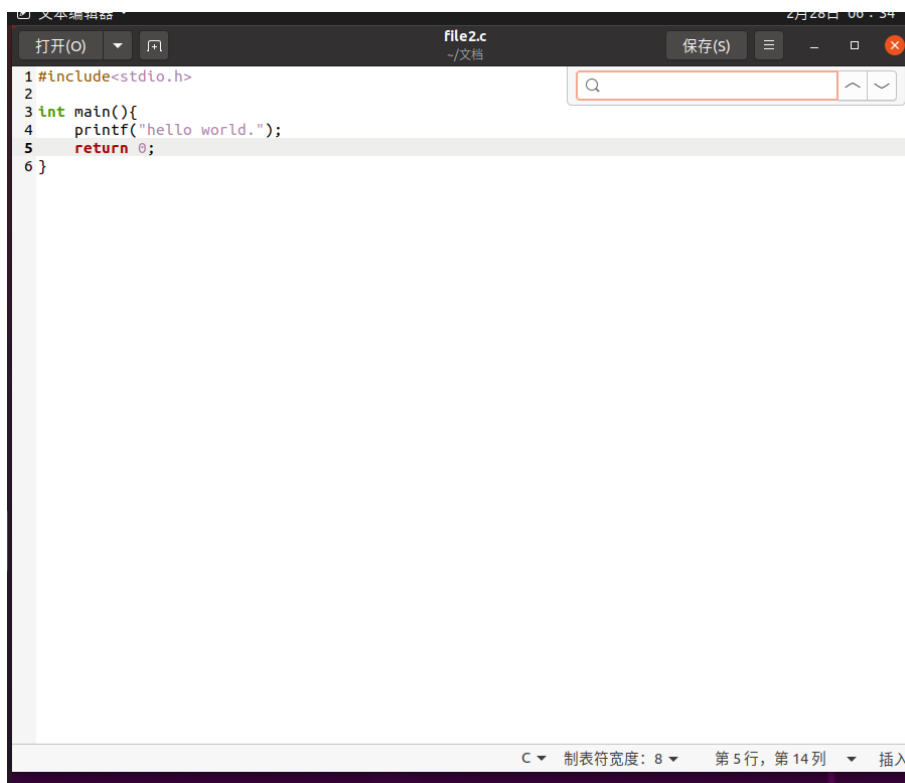


图 10: 查找功能

对于需要处理多文件的用户，gedit 提供了标签页功能，可以通过“文件”菜单的“打开”选项或直接拖拽文件到窗口中同时编辑多个文件，每个文件以独立的标签页显示，切换方便。保存文件时，gedit 默认使用当前文件名，若需另存为其他名称，可选择“文件”菜单中的“另存为”，指定新路径和文件名后保存。gedit 还支持基本的文本格式调整，例如通过“查看”菜单启用行号显示，便于定位特定行，或者调整字体和颜色方案以提升阅读体验。

#### 1.2.4 熟悉 gcc 编辑器

gcc 是 Linux 系统中广泛使用的 C 语言编译器，全称 GNU 编译器套件，它功能强大且灵活，能够将 C 程序代码转化为可执行文件。



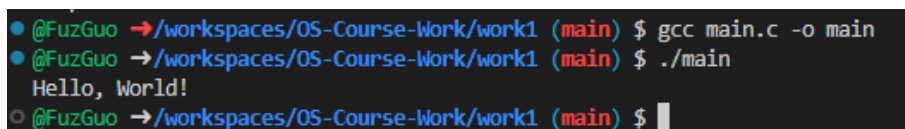
要使用 gcc 编译一个 C 程序，首先需要编写一个简单的代码文件，例如可以用编辑器创建一个名为 “hello.c” 的文件，内容包含标准 C 程序结构：引入头文件 `stdio.h`，定义 `main` 函数，并在其中用 `printf` 输出 “Hello, World!”。



```
main.c U x main.s U
work1 > C main.c > main()
1  #include <stdio.h>
2
3  int main() {
4      printf("Hello, World!\n");
5      return 0;
6  }
```

图 11: 示例程序

gcc 的编译过程通常包括预处理、编译、汇编和链接四个阶段，但通过一条命令就能完成所有步骤，在终端输入 “`gcc hello.c -o hello`”，这条命令会将 “hello.c” 编译并链接成一个名为 “hello” 的可执行文件，之后输入 “`./hello`” 即可运行程序，看到输出结果。如果编译过程中出现语法错误，gcc 会显示错误信息，帮助用户定位问题，例如缺少分号或未定义变量。

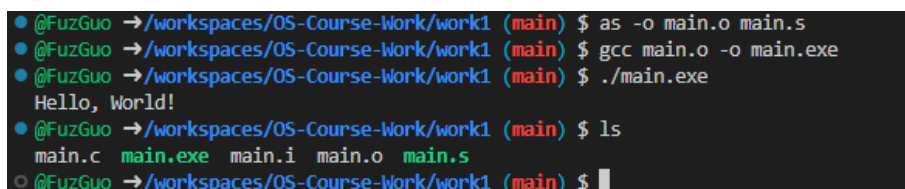


```
@FuzGuo →/workspaces/OS-Course-Work/work1 (main) $ gcc main.c -o main
@FuzGuo →/workspaces/OS-Course-Work/work1 (main) $ ./main
Hello, World!
@FuzGuo →/workspaces/OS-Course-Work/work1 (main) $
```

图 12: 示例指令

此外，还可以分步编译：

- 1、预处理：`gcc -E m.c -o m.i`
- 2、编译：`gcc -S m.i -o m.s`
- 3、汇编：`gcc -c m.s -o m.o`
- 4、连接：`gcc m.o -o m.exe`



```
@FuzGuo →/workspaces/OS-Course-Work/work1 (main) $ as -o main.o main.s
@FuzGuo →/workspaces/OS-Course-Work/work1 (main) $ gcc main.o -o main.exe
@FuzGuo →/workspaces/OS-Course-Work/work1 (main) $ ./main.exe
Hello, World!
@FuzGuo →/workspaces/OS-Course-Work/work1 (main) $ ls
main.c  main.exe  main.i  main.o  main.s
@FuzGuo →/workspaces/OS-Course-Work/work1 (main) $
```

图 13: 示例指令

### 1.3 实验体会

通过该实验，我完成了 Linux 常用命令、vi 编辑器、gedit 编辑器以及 gcc 编译器的学习与实践，分析了它们在操作系统中的基本功能和使用场景。在实验过程中，遇到了一些问题，比如初次使用 vi 编辑器时因不熟悉模式切换导致无法正常输入文本，最终通过查阅资料和反复练习掌握了命令模式与插入模式的切换方法；在 gcc 编译 C 程序时，因遗漏头文件引发错误，通过仔细检查代码并添加正确头文件解决了问题。这些经历让我认识到细心和实践的重要性，也加深了我对 Linux 环境下工具使用的理解。