**同济大学计算机系**

**操作系统课程实验报告**

****

**学 号 2251557**

**姓 名 代文波**

**专 业 计算机科学与技术**

**授课老师 方钰**

**实验七：UNIX V6++文件系统**

**一 实验目的**

结合课程所学知识，通过在UNIX V6++实验环境中编写使用文件管理相关的系统调用或库函数的应用 程序，进一步了解UNIX文件管理的工作过程。

**二 实验设备及工具**

已配置好UNIX V6++运行和调试环境的PC机一台。

**三 预备知识**

（1）在UNIX V6++的/lib/file.c文件中了解UNIX V6++支持的所有和文件管理有关的库函数。

（2）复习利用fork，wait和exit如何进行多进程编程及父子进程间的同步。

（3）熟悉UNIX文件系统的内存打开结构和父子进程对文件打开结构的共享。

**四 实验内容**

**4.1. 熟悉UNIX文件系统的接口**

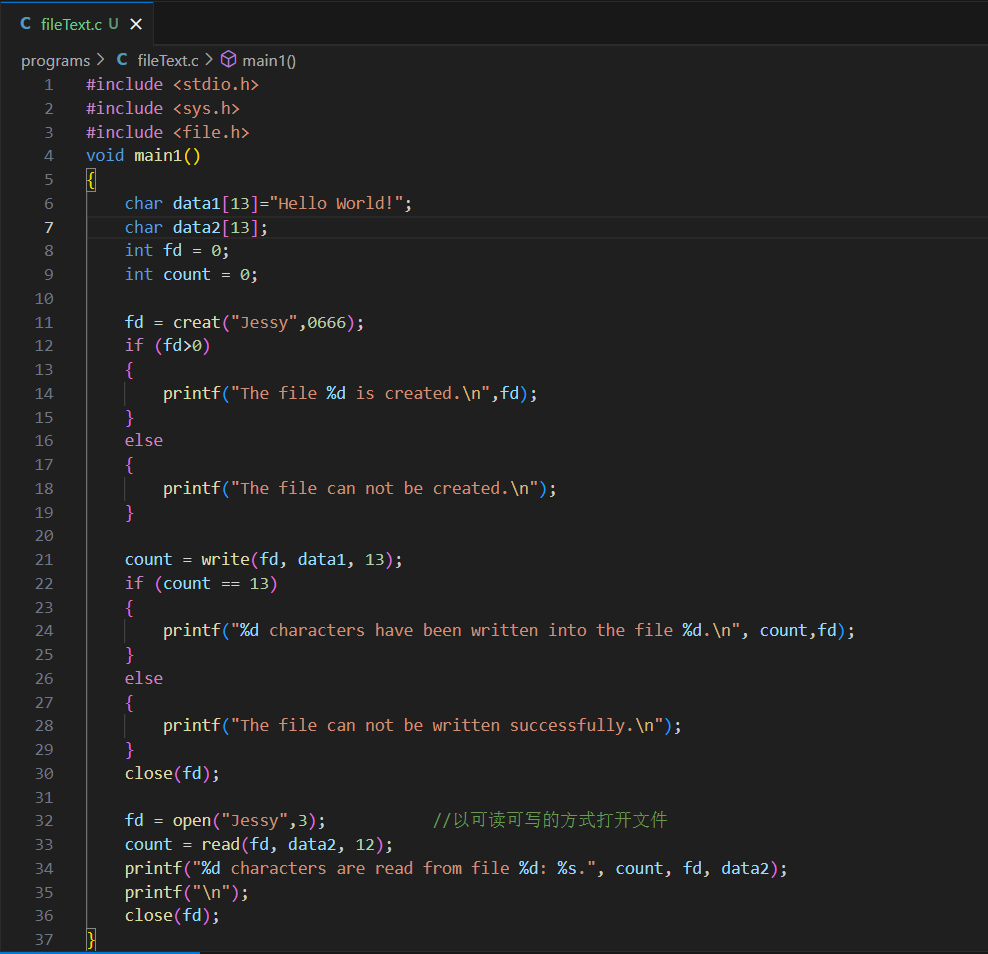
编写可执行程序fileText，实现以下功能：

（1）进程在根目录下创建文件“/Jessy”，创建时设置三类用户对该文件都有读写~~和可执行~~的权限；（**补充：这里我都是按照可读可写权限来创建文件了**）

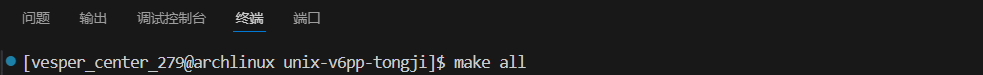
（2）向其中写入字符串“Hello World！”

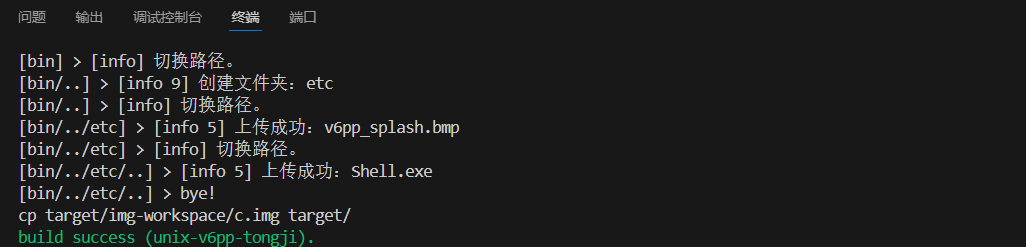
（3）进程将“/Jessy”文件的内容读出，屏幕打印，以判断写入的是否正确。

**4.1.1 在 program 文件夹中加入一个名为fileTest.c的文件**

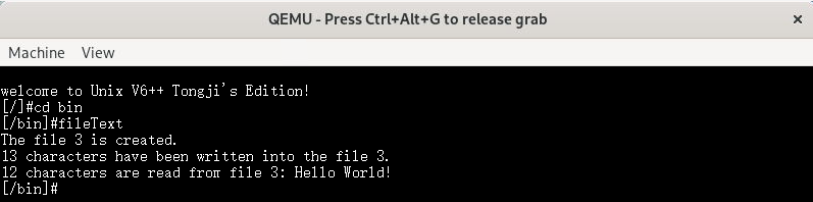


**4.1.2 重新编译运行UNIX V6++代码**





**4.1.3 程序运行结果**



**4.1.4 问题一**

文件创建成功之后，为什么没有直接完成读写操作，而是写过之后，先关闭，再重新打开？

（提示：读者可以尝试在代码1中将写操作完成之后的关闭文件和打开文件两句代码注释掉，将得到如下图所示的错误输出。



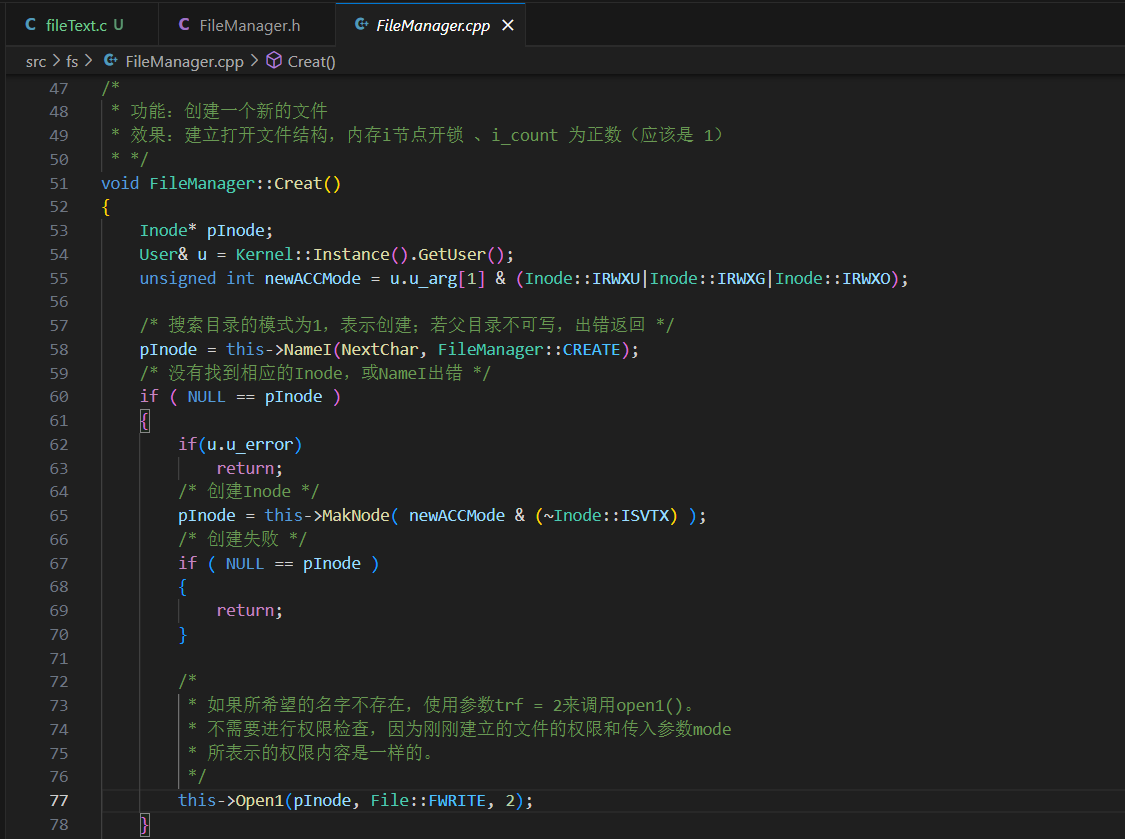
尝试解释其中的原因，并从UNIX V6++的代码中找到依据。）

解答：

文件创建成功后，会自动以可写的方式打开文件，这时可以对文件进行写操作，但是不可以对文件进行读操作。而后面我们要先把文件内容读到data2，然后把data2再写到文件里面，所以只能先关闭文件，再用可读可写模式打开文件。

UNIX V6++代码解释：

从下面代码的第77行可以看到，在文件成功创建后，会以File::FWRITE（可写权限）打开文件，等待写入信息。

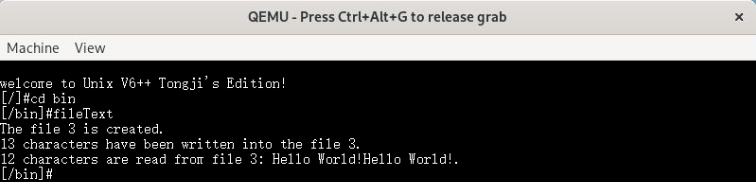


**4.1.5 问题二**

在代码1中，将两个字符串数组的长度都改为12，如下所示：



程序运行将获得如下图所示的输出，请解释出现这样的输出的原因。



解答：

如果这样修改，字符串data1将会没有终止符（尾0），之后从文件读出来写到data2后，也会导致data2没有终止符（尾0）.此外，在函数main1的栈帧中，数据data2、data1连续存储，这就导致在最后打印data2内容的时候，会将data2、data1的内容全部打出来。

**4.2 父子进程共享文件的读写权限和读写指针**

将fileTest 程序的代码修改成如代码2所示。代码主要流程如下：

（1）父进程首先创建了“/Jessy”文件，创建时，给三类用户分别设置了读写~~和可执行~~的权限；（**补充：这里我都是按照可读可写权限来创建文件了**）

（2）创建成功后，父进程将该文件关闭；

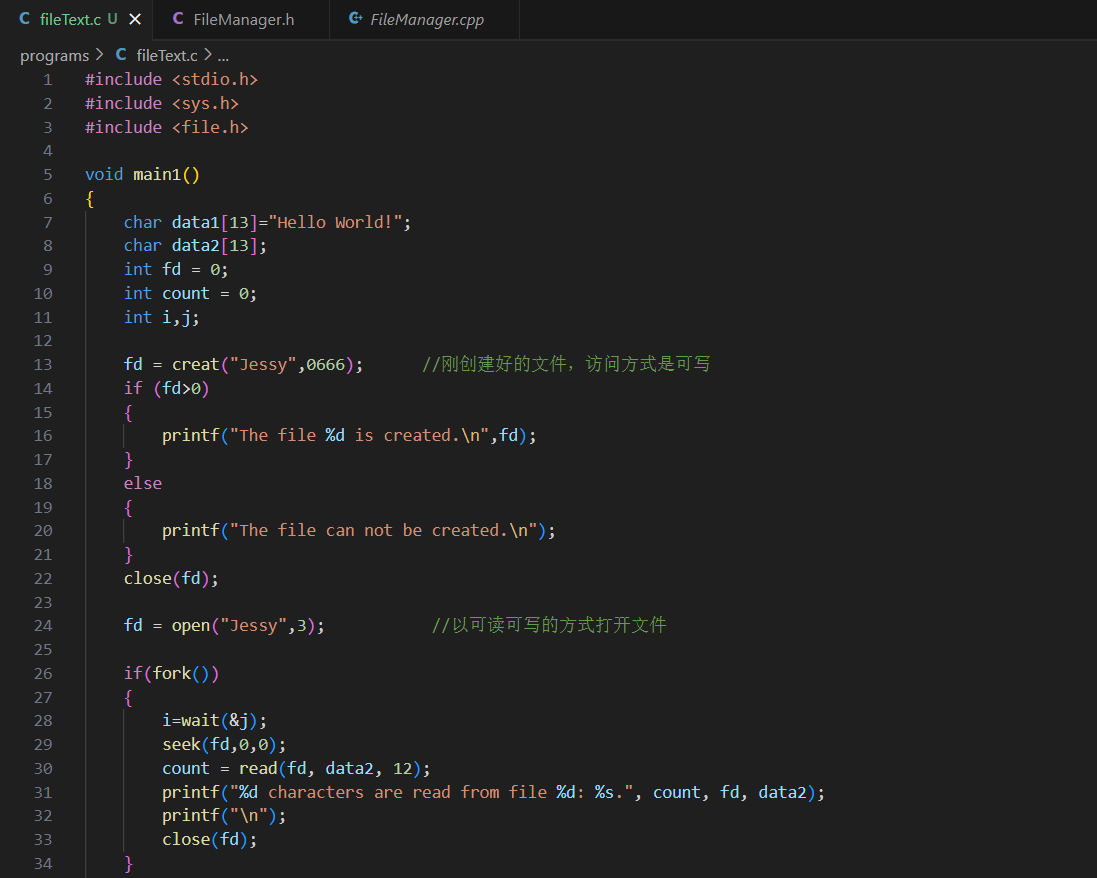
（3）父进程以可读可写的权限重新打开该文件，此时，建立了该文件的内存打开结构；

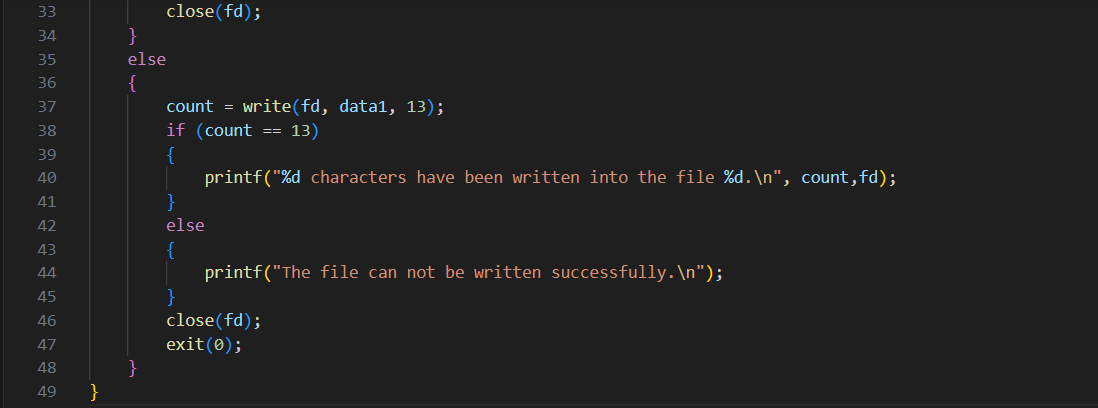
（4）成功创建子进程后，父进程睡眠等待子进程结束；

（5）子进程上台后，通过共享的文件打开结构，向“/Jessy”中写入“Hello World！”，子进程结束， 唤醒父进程；

（6）父进程上台后，从该文件中读出“Hello World！”，并在屏幕打印。

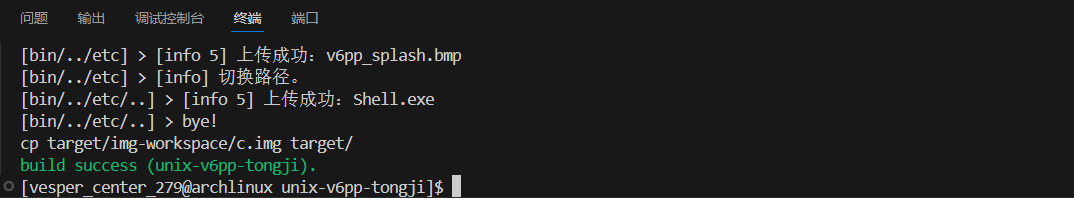
**4.2.1 在 program 文件夹中修改名为fileTest.c的文件**





**4.2.2 重新编译运行UNIX V6++代码**





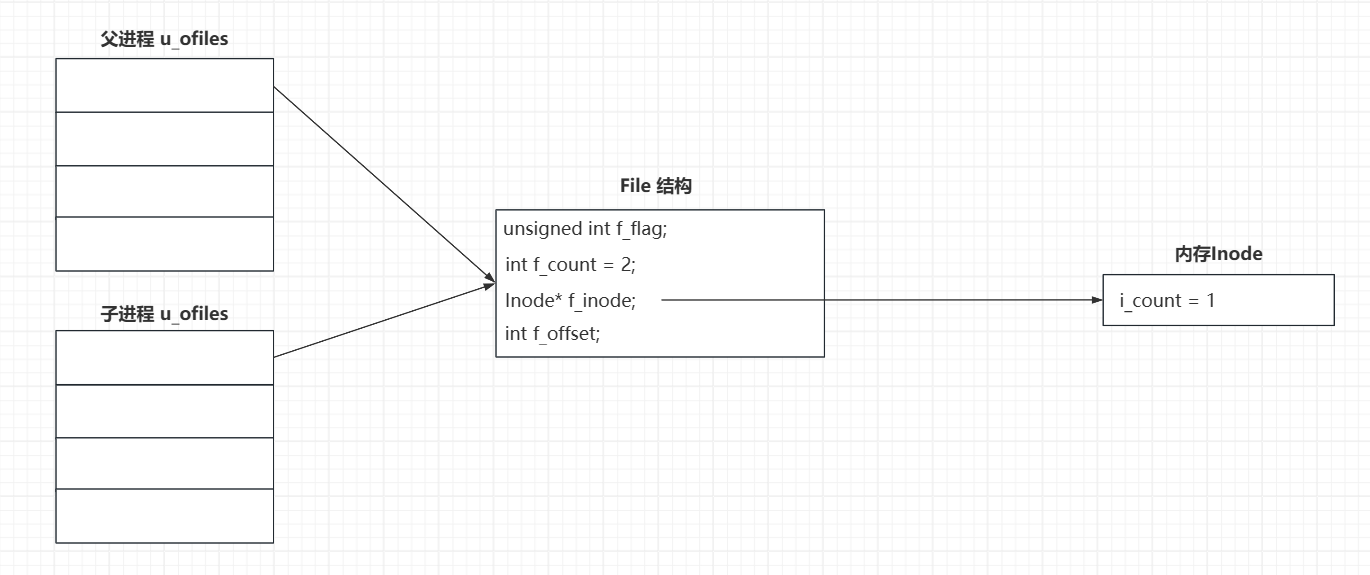
**4.2.3 程序运行结果**

****

**4.2.4 问题一**

以文字或绘制的方式说明在代码2中，父子进程如何实现对文件的读写权限和读写指针的共享；

解答：代码2中父子进程共享了同一个File结构以及同一个内存Inode节点，因此具有文件读写权限和文件读写指针。

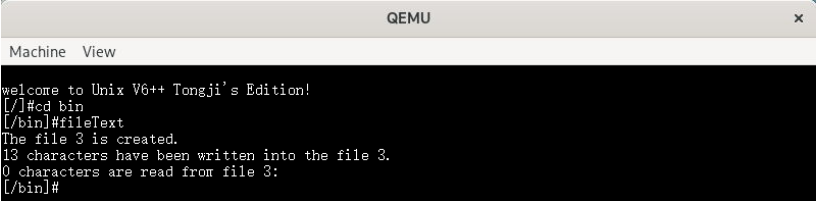


**4.2.5 问题二**

父进程被唤醒重新上台后，为什么要执行seek语句，如果没有这条语句，程序最后的输出是什么样的？为什么？

解答：

如果没有这条语句，最后输出如下图所示：



原因：

父子进程共享文件读写指针，子进程写操作完成后，文件读写指针停留在文件末尾。如果不执行seek语句将其放置回文件开头，那么父进程将从文件末尾进行读操作，因此会读不到任何东西。

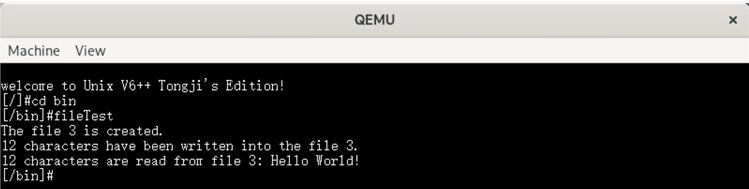
**4.2.6 问题三**

代码2中，父子进程执行的close操作有何不同？

解答：子进程先进行close操作，后先释放打开文件描述符fd，然后让对应File结构中f\_count-1,发现f\_count没有递减到0，操作结束。之后，父进程执行close操作，首先释放打开文件描述符fd，然后让对应File结构中f\_count--，发现f\_count=0，则释放该File结构，然后让对应内存Inode节点中i\_count--，发现i\_count=0，则释放该内存Inode节点。

**4.3 父子进程以不同的读写权限打开文件**

在本节实验中，要求读者按以下要求编写程序，并得到和下图完全一样的输出：



要求：

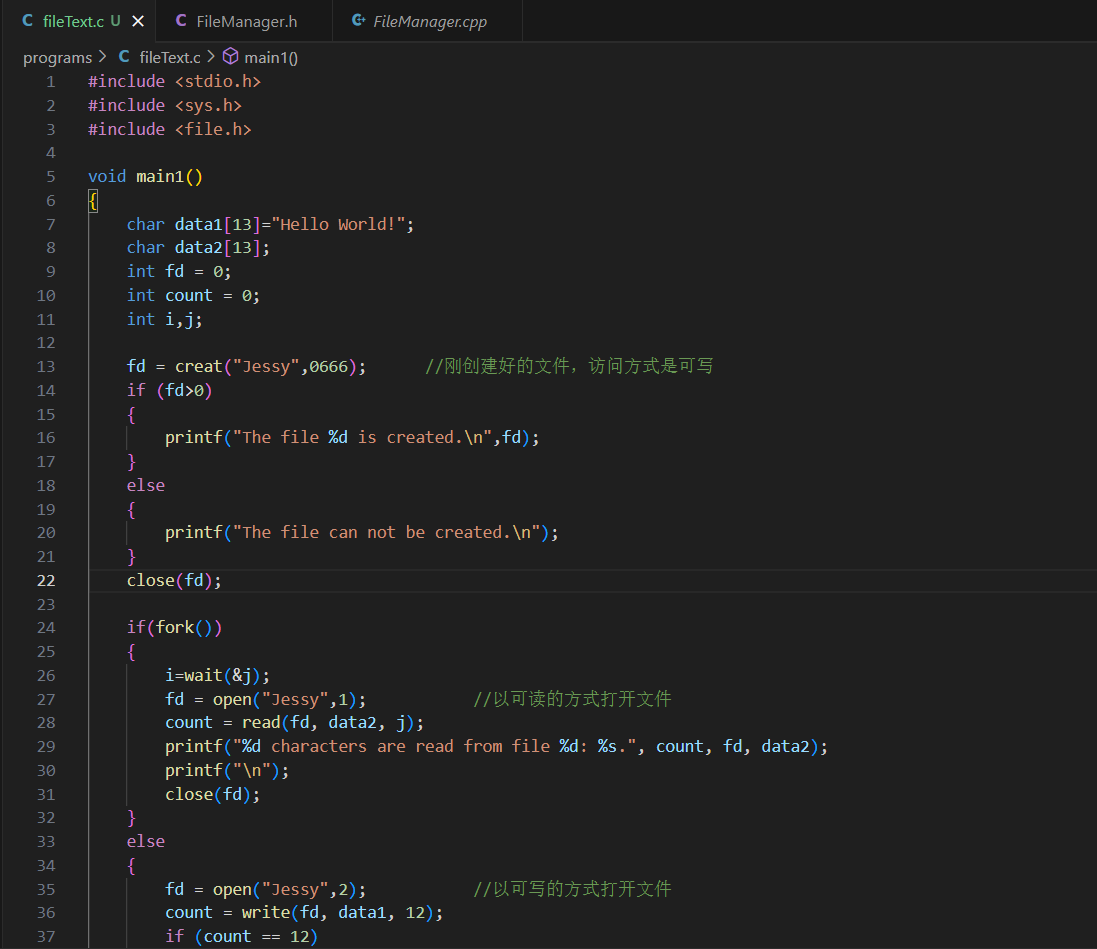
（1）由父进程创建磁盘文件“/Jessy”，创建时为三类用户分别设置可读可写~~和可执行~~的权限；（**补充：这里我都是按照可读可写权限来创建文件了**）

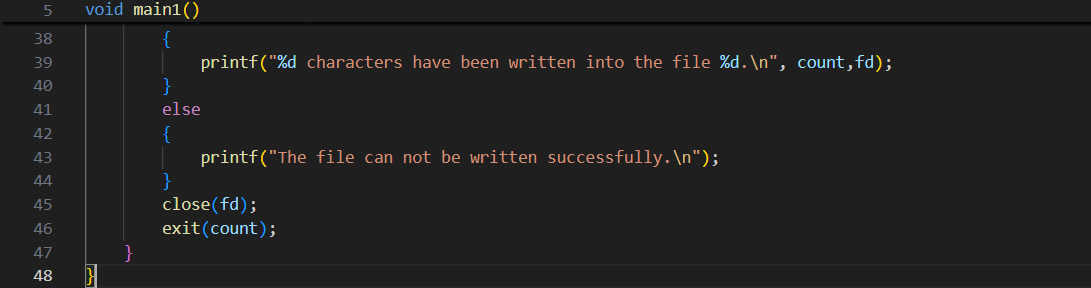
（2）父进程创建子进程，并睡眠等待子进程结束；

（3）子进程上台后，以可写的方式打开该文件，并向其中写入字符串“Hello World！”，关闭文件，进程终止，并将写入的字符个数以终止码的方式传递给父进程；

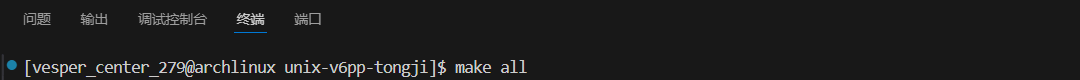
（4）父进程被唤醒重新上台后，以只读的方式打开该文件，按照子进程终止码的数量，从该文件中 读取字符，并在屏幕打印，关闭文件。

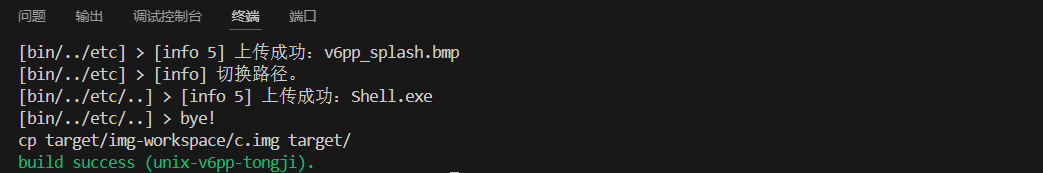
**4.3.1 在 program 文件夹中修改名为fileTest.c的文件**

****

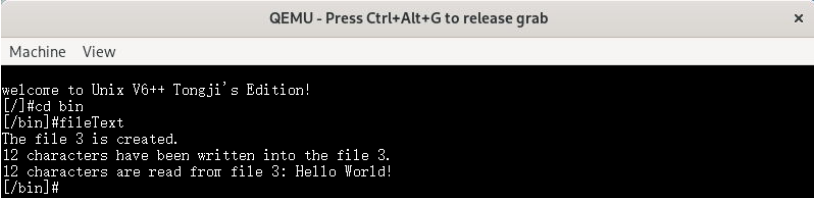
****

**4.3.2 重新编译运行UNIX V6++代码**

****

****

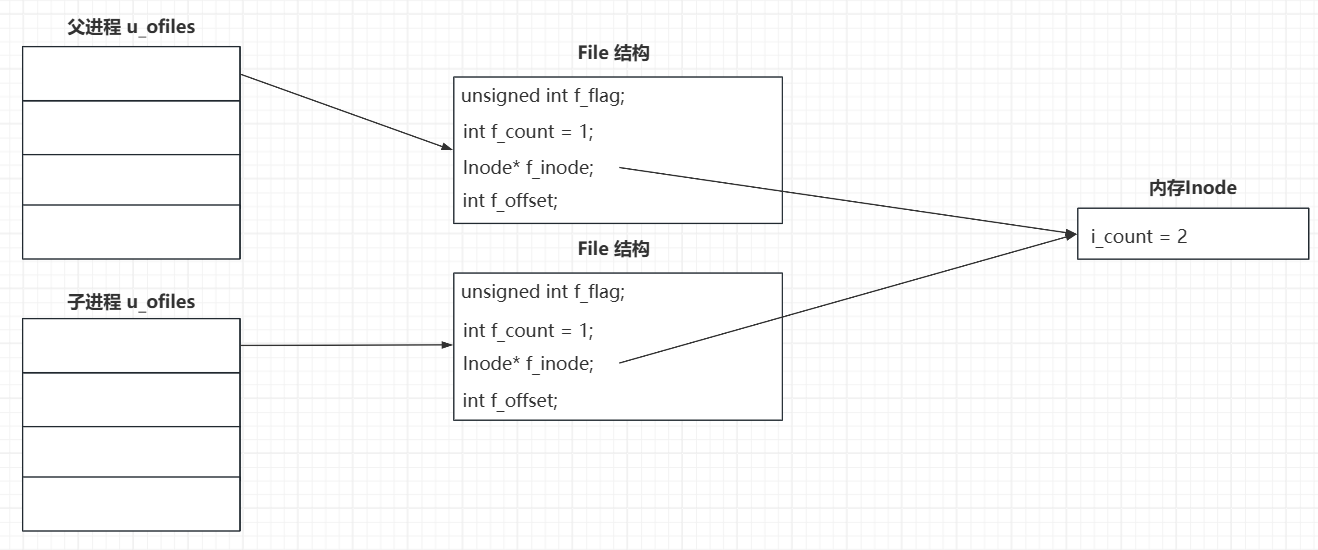
**4.3.3 程序运行结果**



**4.3.4 问题一**

以文字或绘制的方式说明在你的代码中，父子进程对Jessy文件的共享方式；

解答：父子进程分别使用一个File结构来共享一个内存Inode节点进而以不同的权限使用Jessy文件



**4.3.5 问题二**

在这样的共享方式中，父进程被唤醒重新上台后，是否还需要执行seek函数，为什么？

解答：不需要，因为父子进程有各自的File结构，所以有不同的文件读写指针，即子进程文件读写操作不会影响父进程的读写操作。

**4.3.6 问题三**

此处父子进程关闭文件的操作有何不同？

解答：

子进程先进行关闭文件，即执行close操作，首先释放打开文件描述符fd，然后让对应的File结构中的f\_count-1,发现f\_count=0,则释放该File结构，然后让对应的内存Inode结构中的i\_count-1,发现i\_count没有减到0，则关闭文件操作结束。

之后，父进程进行关闭文件操作，即执行close操作，首先释放文件打开描述符fd，然后让对应的File结构中的f\_count-1,发现f\_count=0,则释放该File结构，然后让对应的内存Inode结构中的i\_count-1,发现i\_count=0，则释放该内存Inode节点。