YAUSH设计报告

YAUSH，即Yet Another Unix Shell，使用C语言实现自定义命令行解释器，也就是常说的Shell。下面将对其基本功能、代码架构以及测试结果进行介绍。

# 基本功能

Shell在Linux操作系统上为用户提供一个命令行交互窗口，用户输入命令，shell会对命令进行解析并执行。本作业中对基本的shell功能进行实现，可执行一般的shell命令，并支持输入输出重定向、管道、后台执行等功能。具体功能如下：

1. 执行基本命令，如**ls**，**echo "Hello World !"**，**rm a.txt**等，由一条命令和若干参数组成命令。
2. 将输入、输出重定向到文件，如**ls >> files.txt**，**cat < files.txt**等，将命令的输入保存在文件中，或从文件中读取内容作为命令输入。
3. 命令间可通过管道实现输入输出重定向，如**ls | grep main**，**cat < files.txt | grep main**，将前一个命令的输出作为下一个命令的输入。
4. 支持在后台执行程序，通过在命令最后添加**‘&’**字符，如**find / -name "mysql.sh" &**，可将命令放在后台执行，用户可在命令执行过程中继续输入并执行其他命令，而不需要等待后台命令执行完成。
5. 可查看历史命令，通过输入**history**命令，可查看最近输入的20条命令。
6. 支持文件名tab补全。在输入文件名时，可输入前几个字符并按下**tab**键，实现文件名自动补全，如输入**cat fi**，按下**tab**，命令自动补全为**cat files.txt**。

# 代码架构

代码整体流程可分为三部分，分别是命令输入、命令解析和命令执行。代码文件包括main.c文件和myshell.h文件，其中命令输入部分直接在main.c中实现，命令解析和命令执行通过定义在myshell.h中的函数实现。

代码流程图如下图所示。

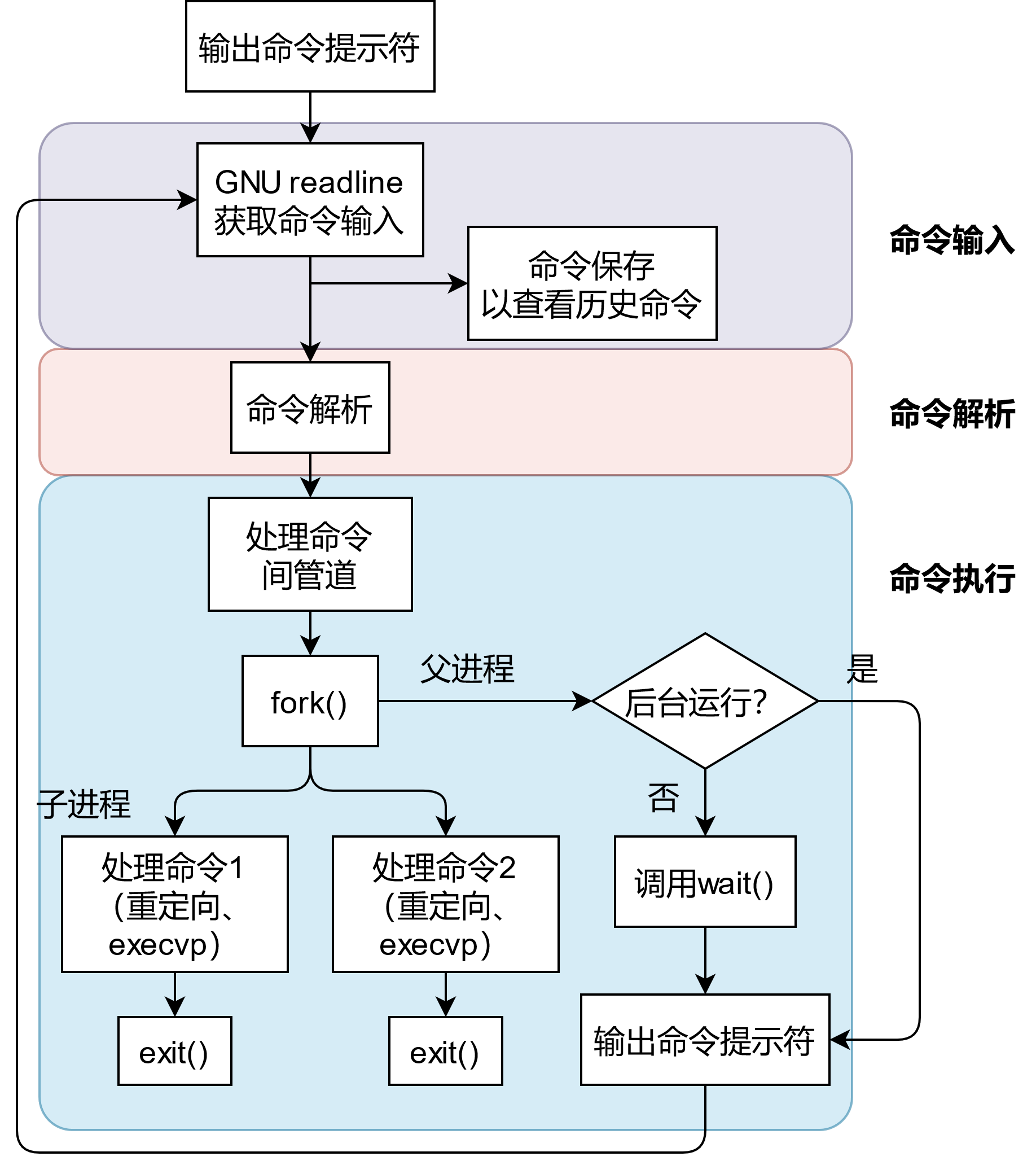


图 1 代码流程图

在命令输入部分，采用GNU readline第三方库进行命令的读入。通过该方式可实现在输入命令时对文件名进行tab键自动补全，并可在输入时使用左右箭头进行光标移动。在读入命令后，将命令保存在字符数组中，以实现**history**命令返回历史命令。

在命令解析部分，首先对输入命令字符串按照空格进行切分，获得字符串列表。然后在列表中查找有无**“|”**字符串，若有，则说明存在管道，需要定义匿名管道，并将相应的文件描述符发送给对应的命令进行重定向，若无，则说明不存在管道，直接执行命令即可。

在命令执行部分，首先根据命令解析结果，在存在管道的情况下定义匿名管道。然后使用fork()函数创建子进程，执行一个或多个命令。而父进程则根据命令是否为后台运行，即判断命令结尾是否有**‘&’**字符，来执行wait()函数等待子进程结束，或直接输出命令提示符开始等待下一条命令输入。在子进程执行命令过程中，如果命令的字符串列表中包含**“>>”、“>”**或**“<”**字符串，则进行相应的重定向操作，将命令的输入或输出重定向到指定的文件中。在子进程执行命令前，还需对命令进行查找，判断该命令是否存在。若命令存在，则调用execvp函数执行命令；若命令不存在，则输出错误信息。最后子进程调用exit()函数结束进程。

若命令为后台执行，则父进程不必等待子进程结束，直接输出命令提示符开始等待新的命令输入并执行；若命令不是后台执行，则父进程等待子进程全部结束后，重新输出命令提示符开始等待新的命令。

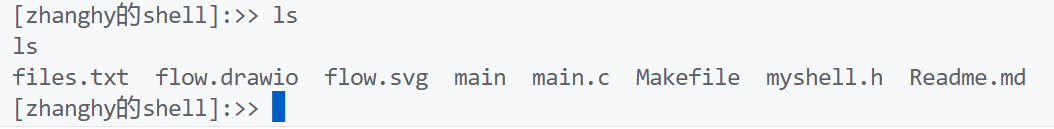
# 测试结果

通过makefile对上述代码进行编译，**make all**，得到可执行文件main。

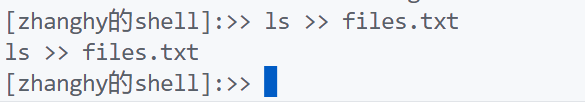
在Terminal运行程序，**./main**，得到输出如下图所示。



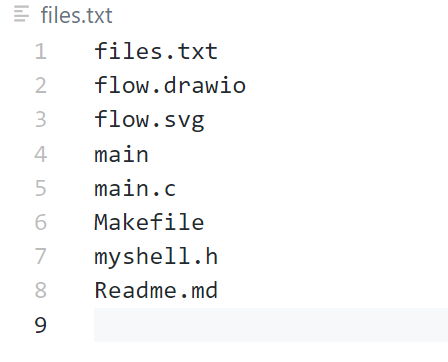
可在命令提示符后输出命令。当输入简单命令**ls**后，输出如下图所示。



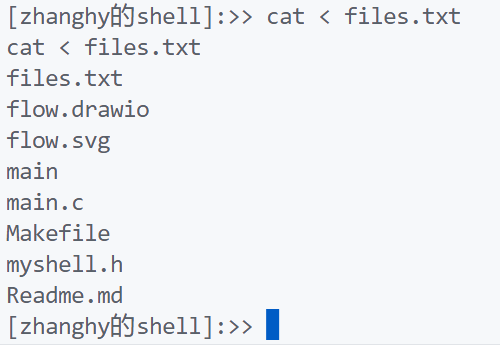
当输入重定向输出命令**ls >> files.txt后，输出如下图所示。**



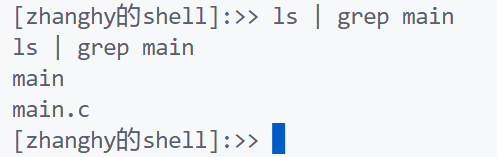
在当前目录下生成文件file.txt，其内容即为**ls**命令的输出，如下图所示。



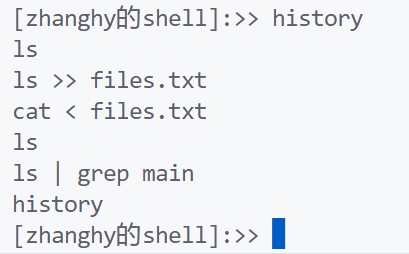
当输入重定向输出命令**cat < files.txt后，**输出结果如下图所示。



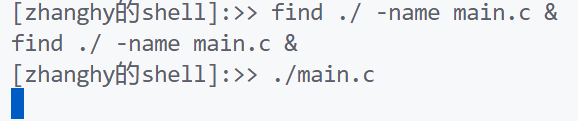
当输入管道命令**ls | grep main后，输出结果如下图所示。**



当输入**history**命令后，输出结果如下图所示。



当输入后台执行命令**find ./ -name main.c &**后，输出结果如下图所示。可以看到该命令的输出结果在输出下一次命令提示符后输出，说明父进程未等待子进程命令执行结束即开始等待接受新命令。



经上述测试，可以看到本作业实现的YAUSH成功完成相应的功能。