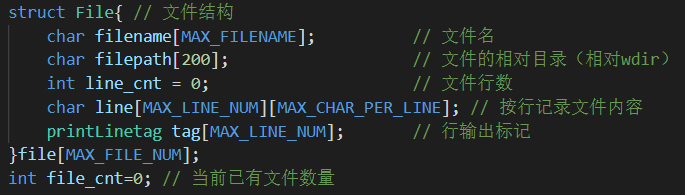
**程序设计基础Tinyshell说明文档**

**——第二部分grep的实现**

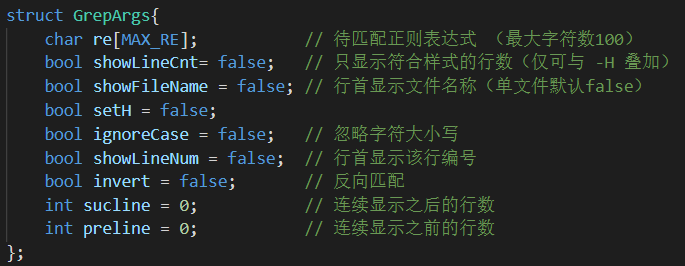
计05 聂子坤 2020010963

1. **数据定义**
2. **File结构**



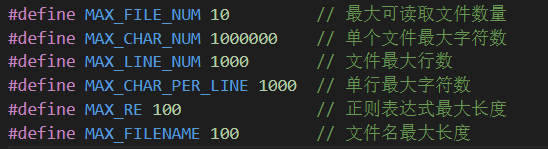
若参数为文件，每次按行读入文件的全部内容，保存在File结构数组中（标准输入stdin也作为一个名为“（standard input）”的文件记录在内）。File除记录文件内容外，还记录了控制行输出的一些标记信息。

1. **GrepArgs结构**



解析参数时记录下的参数信息，包括是否显示文件名、是否显示行号、是否忽略大小写、是否只显示行数、是否反向匹配等。doGrep函数会产生GrepArgs的一个实例，参数解析完成后将其作为实参传递给matchLine函数。

1. **处理能力约定**



1. **基本逻辑架构**
2. 

**doGrep函数**的主体部分负责解析参数列表，完成对File和GrepArgs的设置。即读取文件信息、设置匹配参数。随后以GrepArgs的实例args作为实参传入Analyse函数。

1. 

**Analyse函数**的主体部分负责根据接收到的各种参数分析文件内容与输出匹配信息。对每个文件的每一行调用matchLine函数，该函数返回成功匹配的区间（不匹配则区间首尾为-1）。Analyse函数根据参数决定输出信息（到strout中）。

1. 

**matchLine函数**接收正则表达式、待匹配行、是否忽略大小写作为参数。从待匹配行的首位开始调用match函数匹配正则表达式。match函数返回最大匹配区间的末尾位置。一旦match函数返回有效值，matchLine即记录匹配成功的区间并返回Analyse函数。matchLine保证了匹配到的字符串是首个匹配的子串。

1. 

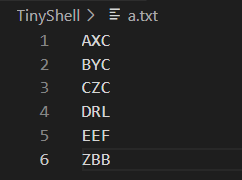
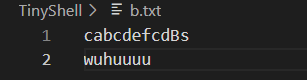
**match函数**接收正则表达式、待匹配子串a、是否忽略大小写作为参数，并返回最大匹配区间的末尾位置（匹配不成功则返回-1），match保证了匹配到的字符串是最长匹配的子串。

match的实现基于动态规划思想。bool dp[i][j]表示re的前i位与a的前j位是否成功匹配。若re[i]为‘.’，由于’.’可以匹配任一字符，则dp[i][j]取决于dp[i-1][j-1]是否成功；若re[i]为‘\*’，则只要dp[i-1][j]、dp[i][j-1]、dp[i-1][j-1]有一个匹配成功则dp[i][j]成功；若re[i]为普通字符，则dp[i][j]为真当且仅当dp[i-1][j-1]为真且re[i]与a[j]相同（若忽略大小写，则需二者在忽略大小写的情况下相同）。返回re全部匹配（dp的最后一行）的最后一个真即可。

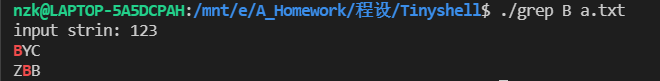
设re长度为s，a长度为t，该算法**时间复杂度为O(s\*t).**

1. **基本功能展示**

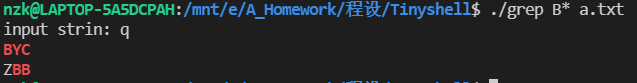
文件a.txt与b.txt（strin由手动输入）：

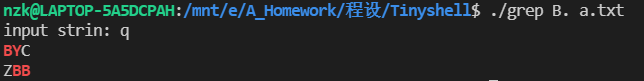
 

1. 基本字符串匹配：加粗红色显示

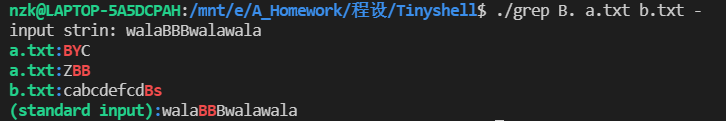


1. 通配符匹配





1. 多文件、标准输入匹配（多文件默认输出文件名）



1. -H显示文件名，-i忽略大小写，-n输出行号（文件名、行号绿色，标点蓝色）



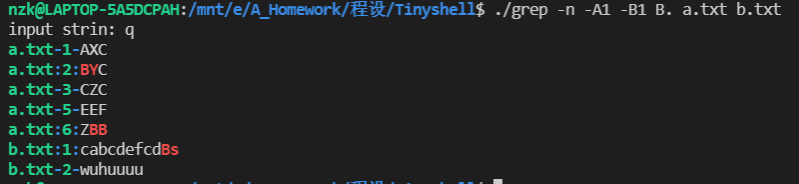


（参数放在文件前或文件后或多个文件之间均可）

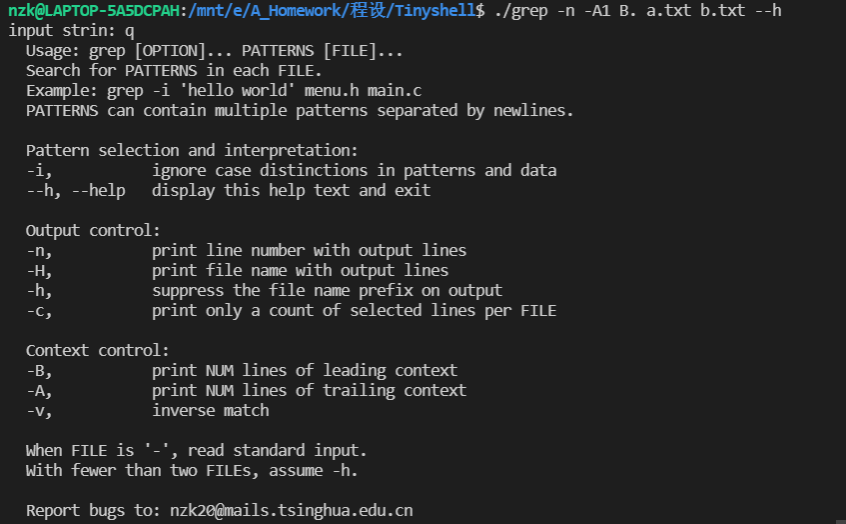
1. -c仅输出匹配的行数



1. -A[n]、-B[n]输出前后n行（“：”表示匹配成功，“-”表示匹配失败）

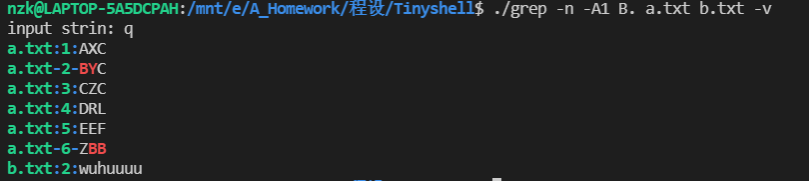


1. --h或--help输出提示（不论其他参数是什么）

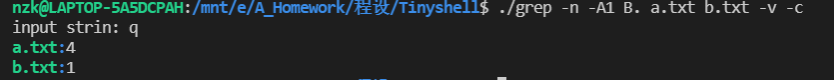


1. **扩展功能展示**

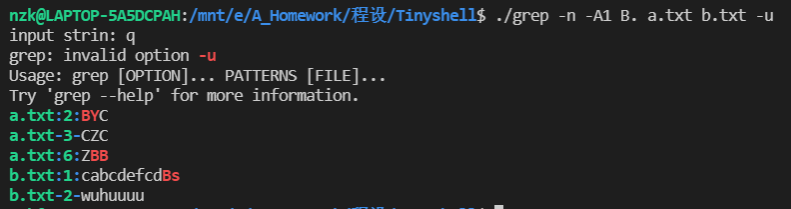
实现了‘-v’参数的处理，其功能为反向匹配，即输出不匹配的行。



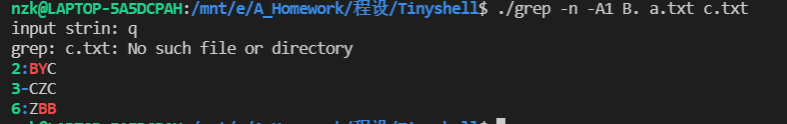
（这里与正则表达式匹配的地方仍用红色标出，但其行号符号变成了‘-’，原先不匹配的符号变成“：”）



1. **异常处理**
2. 参数错误：提示[invalid option]，但仍在可识别参数的基础上继续匹配。



1. 文件不存在：提示[no such file or directory]，但仍在已有文件的基础上继续匹配。



1. 无有效文件：提示使用‘-’作为标准输入。

