用户手册

1. 基础命令

1.1 指令自动补全

使用方法

输入指令的至少两个字符后，输入*符号，即可代替后续的字符输入。如果查询到匹配的指令，则直接执行，没有则提示错误。

例如：

```
his* 等价于输入 history
fork* 等价于输入 forktest
ev* 1+2*3/4 等价于输入 eval 1+2*3/4
```

1.2 自动修复命令

在使用命令行进行指令输入时，操作人员难免会出现typo。对于一些较长的复杂指令，重新输入是费时费力的。于是，我们开发了自动修复命令的程序，当系统侦测到可能的typo时，会提示并询问用户是否出现了输入错误，提供更改好的指令，使得用户可以选择快捷输入。

使用方法

例如：

```
输入
catt README
系统会询问
if you want to exec cat, type "yep".
此时执行
yep
即可执行
```

```
cat README
```

1.3 查看历史命令

在使用shell功能的过程中我们总会遇到这样的需求：我们想查看历史指令。我们的history指令即是模仿history指令在Linux中的行为的。

使用方法

1. 查看所有的历史指令

   ```
   history
   ```

2. 查看最后n个历史指令

   ```
   history n
   ```

   系统会检测n的合法性并返回不多于n个的指令记录。如果n大于实际拥有的历史指令数目，则返回所有历史指令。

1.4 调用历史命令

除了查看历史指令外，我们还支持按照历史指令的记录序号调用历史指令。

使用方法

调用history记录的第n个指令

```
history !n
```

调用时会检查n的合法性，如果n的范围处于history记录的规模范围内，则会直接执行相应的指令；如果超出范围，则提示执行失败。

1.5 清屏指令

在使用文本编辑功能或者其它的一些场景时，我们需要对当前屏幕进行清屏操作。基于此需求，我们实现了`clear()`**系统调用**，并实现了`clear`指令。

使用方法

键入

```
clear
```

即可清理当前屏幕。

2. 文件管理

2.1 文件的复制

使用方式

```
cp file1 file2
```

意为拷贝 file1 到 file2.

2.2 文件的移动

使用方式

```
mv file1 file2
```

意为移动 file1 到 file2.

2.3 文件的重命名

使用方式

```
rename oldpath newpath
```

意为将 oldpath 重命名为 newpath。

3. 文档编辑

3.1 实现功能

- 实现读取已有文件与创建并保存文件
- 实现命令、编辑模式
- 实现任意位置的插入与删除操作
- 实现代码高亮和显示行号等用户友好的功能

3.2 用户手册

- 输入命令行edit filename ，若存在文件则打开，否则新建文件
- 进入visual mode，输入e进入edit mode，输入q退出，输入s保存文件
- 进入edit mode，具体操作下面详细说明，esc退出edit mode进入visual mode

edit mode中具体操作如下：

功能	实现方式	备注
写入字母	在draft中相应位置输入，按下enter	-
删除字母	在draft中相应位置输入‘@’，输入一个‘@’表示删除一个字符，按下enter	-
换行	上下键，输入一个字符后需要按下enter进行更新	输入一个字符，若想使字符生效则需要按下enter
左右移动光标	左右键，可以连续移动	-

功能	实现方式	备注
退出edit mode	按下esc，按下enter	-

4. 其它

4.1 命令行式计算器

使用方式

```
eval expression
```

expression指的是表达式，支持浮点数运算，支持运算符：+ - * / ! ( ) 。
例如：

```
eval 1.23+3.14/12.34*99.99
```