14:01

基本安排

- 添加 Delete 功能
- 考虑 Delete 对 preSelectedTree 的影响
- 样式的添加
- Edit 对样式的修改
- 考虑 `display` 模式下批量选中的添加, `visual` 模式

Delete 对 preSelectedTree 的影响

针对将选中的元素删除

通过 `ylen` 来判断左右是否有元素. `ylen[msubselected.y]-1` 的值是光标所在行的最后一位坐标.

若选中的是 node, 删除后:

- 如果右侧还有元素 `if m.subselected.x < ylen[m.subselected.y]-1`, 则 `subselected.x` 和 `subselected.y` 都不变
- 如果右侧没有元素 `if m.subselected.x == ylen[m.subselected.y]-1 && m.subselected.x > 0`, 但左边有元素, 则 `subselected.x` 减一, `subselected.y` 都不变
- 如果右侧没有元素 `if m.subselected.x == ylen[m.subselected.y]-1 && m.subselected.x == 0`, 左边也没有元素, 则 `subselected = preSelectedTree[len(preSelectedTree)-1]` , `preSelectedTree = preSelectedTree[:len(preSelectedTree)-1]`

若选中的是 tree, 删除后:

- 如果右侧还有元素, 则 `subselected.x` 和 `subselected.y` 都不变
- 如果右侧没有元素, 但左边有元素, 则 `subselected.x` 减一, `subselected.y` 都不变
- 如果右侧没有元素, 左边也没有元素, 则 `subselected = preSelectedTree[len(preSelectedTree)-1]` , `preSelectedTree = preSelectedTree[:len(preSelectedTree)-1]`

和 node 行为似乎一致.

具体的 Delete 功能

在 `display` 模式下, 按下 `d`, 删除光标下选择的元素, 即 `x == m.subselected.x && y == m.subselected.y` 的元素.

需要出现一个弹窗, 并询问是否删除. 有两个按钮, Yes 和 No. 弹窗出现在窗口正中.

- 如何计算窗口位置? 暂定为在 `Update()` 函数中用 `msg.Width` 和 `msg.Height` 来确定
- 如何实现弹窗? 需要将原来显示的字符进行替换, 参考之前 suggestionlist 和 display 的写法, 需要一个 `viewport`
- 如何实现按钮选择? 两个 box style 包裹字符串数组, 默认对数组中的第一个元素施加 `selected` style, 按下 `l` 或者 `tab` 之后取消第一个的 `selected` style 并施加到第二个上
- 如何删除? 用 `DelNode()` 删除 node, 需要先找到 fatherTree, 用 `DelSubTree()` 删除 tree, 同样要找到 `fatherTree`
- 按键的绑定? 在 `display` 模式下按 `d` 执行弹窗, 在弹窗显示过程中, 按下 `esc` 取消, 或者选中 `no` 后按 `enter` 取消, 选中 `yes` 后按 `enter` 删除

- 是否添加新 mode? 添加一个 `del` 模式 ☒
- 弹窗显示的提示文本? 需要加上 tree 或者 node 的信息吗? 加上 treeName 或者 nodeAlias
- 删除以及坐标更换单独一个函数, 进度 `del` 模式后, 先判断选择, 再决定是否 delete
- pop Window 实现起来有难度, 可以考虑底部显示是否 delete
- 使用 pop Window 需要考虑要替换的 byte 的位置, 如何计算得出? 先得出 `displayBytes` 的长度, 因为此时不会有 suggestion, 因此不用考虑. 需要一个函数, 根据 viewport 的宽和高来计算要替换的 bytes 的开始位置

Command Prompt 相关

- 主体思路与 vim 的 Ex 命令类似
- 需要考虑 display 的 width 和 height
- 用 : 开头
- 用 `enter` 执行
- 如果是删除操作, 则是 `:delete tree(TREENAME...)` 或 `:delete node(NODEALIAS...)`, 可接收多个参数等
- 同样需要添加 `edit`, `add`, `move` 等函数, 可借鉴命令行子命令
- `Command` 类型用于处理命令, 成员可能有:
  - textinput, `textinput.Model`
  - prompt char, `string`
  - command, `string`
  - args, `[]string`
- 添加 `command` mode? 按 `esc` 返回到 `display` mode,
- 应该只能在 `display` mode 下进入 `command` mode

初始化 Command 类型结构体

- textinput:
  - Blink 可有可无
  - 某些地方需要有颜色变化
  - No placeholder ☒
  - 一开始不 focus (即 Blur)
  - suggestion 的设置, 同 vim 一样, 命令行上方显示补全列表, 当前的高亮显示
  - `Keymap = textinput.Keymap{}`, 将默认的清空 ☒
  - 注意 trigger key stroke 为 : ☒
- prompt char:
  - 设置为 : ☒
- command 和 args:
  - 空字符串 ☒
  - 在按下回车后, 通过解析 `textinput.Value()` 返回的字符串拆分存储到 `command` 和 `args` 中

Update Command 类型结构体

- textinput:
  - 在切换到 `command` mode 之后, 在 `afterModeChange()` 函数中 `Blur` 其他 input, `Focus` command 的 input
  - 在 `updateSuggestionList()` 函数中更新 suggestion list
  - 在 `updateUIComponents()` 函数中返回 `cmd`

  - 在各键位下添加行为
  - command 和 args 只有在 `enter` 时更新

一些行为的描述

- 在 `display` mode 下, 按 `:` 可进入 `command` mode
- 在 `display` mode 下, 按 `d` 可进入 `command` mode 并输入 `:delete node(NODEALIAS)`, 此时回车后, 预输入 `:confirm yes|no`
- `comfirm` 功能可关闭
- 是否添加撤销功能?
- 是否为每一个 command 添加文档说明?
- 在 `command` mode 按下 `enter` 后, 会:
  - 获取 `input.Value()`, 交割一个函数如 `commandParse()` 分离 `cmd` 和 `args`, 然后传递给 `commandHandler()` 实际执行
  - 返回 `display` mode ☒

Bug 相关

- 终端大小变化后, UI 出现问题, 需要添加动态变化

欲添加的功能

- 一个成员用于指名当前正在使用的 textinput
- 增加一个额外的文件存储命令函数