Ndd (Notepad-double-decrease) V3.8.2 用户说明书

Ndd Team

2024 年 8 月 - 2026 年 07 月

目录

1 软件简介 **1**

1.1 Ndd 名称由来和开发初衷 1

1.2 技术组件 2

1.3 Ndd 不会涉及政治言论 3

1.4 后续还会继续开源吗？ 4

2 Ndd 软件功能介绍 **6**

2.1 下载与更新软件 6

2.1.1 为什么不做自动下载更新 8

2.1.2 Ndd 可支持的操作系统 9

2.2 主体功能一览 10

2.2.1 windows/Linux/Macos/UOS/麒麟效果图 16

2.2.2 换行符号转换 16

2.2.3 切换文件打开方式 18

2.2.4 去除文件中空白，大小写转换 18

2.2.5 显示文件列表、显示空白字符、显示网址高亮 19
2.2.6 文件 Md5 及各种哈希码计算 20
2.2.7 Base64/Url 工具 20
2.2.8 json-js-xml 格式化工具 21
2.3 安装配置指导 22
2.3.1 Macos 下如何安装 dmg 安装包 22
2.3.2 Macos 安装包损坏报错解决 22
2.4 界面字体或图标太小的调整方法 24
2.5 选择 Qt5 还是 Qt6 版本 25

3 常用功能或使用技巧 26

3.1 界面主题和配色 26
3.1.1 标题栏修改为暗黑色 27
3.2 调整字体或语法配色 28
3.2.1 设置 txt 文件语法对应的字体 29
3.2.2 设置 txt 文件显示的背景色 31
3.2.3 修改编程语言的显示配色 31
3.3 换行符的查找或替换 33
3.4 文件编码、探测与转换 36
3.4.1 文件乱码处理 38
3.4.2 文件编码转换 40

3.4.3 为什么会编码错误，如何避免？ 42
3.4.4 选择文件保存为 gbk 编码后，下次打开还是识别 utf8 编码 42
3.4.5 批量文件编码转换 43
3.5 行号与书签风格 44
3.6 快捷键设置 46
3.7 常见快捷键举例 47
3.8 记录上次关闭时的打开文件 48
3.9 记录上次关闭时打开的文件夹 49
3.10 崩溃后恢复文档和定时保存 51
3.11 主题配色的导入与导出 53
3.12 列编辑 54
3.12.1 用例：使用列编辑功能批量生成维护命令 55
3.13 宏的录制与运行 57
3.13.1 宏录制与回放 59
3.14 文件和语法高亮关联 62
3.14.1 双击打开文件时使用 Ndd 打开 63
3.15 文件内容高亮标记 66
3.15.1 自定义高亮标记的颜色 67
3.16 自定义语言并关键字高亮 68
3.17 命令行打开文件，跳转到指定行 71
3.18 F3F4 选中快速查找 71

3.19 选中文字高亮所有字段 72
3.20 修改选中文字的底色 74
3.21 修改新建文件的默认语法 74

4 查找替换与常见正则规则 76

4.1 查找界面主要功能介绍 76
4.2 查找结果框显示效果配置 79
4.3 在目录中批量查找 82
4.4 常用正则规则介绍 84

5 大文件编辑与处理 88

5.1 打开大文件 88
5.2 大文件修改编码 92
5.3 大文件文本替换 94
5.4 万能文本处理器 95

6 用户相关问题汇总 97

6.1 打开文件显示乱码 97
6.2 打开一个 38M 的 sql 文件，查找很卡 97
6.3 大屏幕下面，界面字体非常小 98
6.4 断电后文件还能找回吗? 98
6.5 tab 如何使用空格代替 98

6.6 注册版和非注册的区别 . . . 101
6.7 赞赏多少金额会发注册码? . . . 102
6.7.1 对赞赏 Ndd 的说明 . . . 102
6.7.2 不能联网的用户，如何进行本地注册? . . . 104
6.7.3 为什么没有收到注册码? . . . 105
6.8 注册失败 . . . 105
6.9 注册码是如何绑定电脑的，重装系统算多个注册吗? . . . 106
6.10 为什么不走开源免费的道路呢? . . . 107
6.11 如何一键修改所有语言的全部字体和大小 . . . 107
6.12 中英文字体为什么没有对齐? . . . 108
6.13 英文操作系统下，字体和光标未能对应 . . . 109
6.14 日文 shift-jis 编码的文件会出现文字覆盖的现象 . . . 110
6.15 Macos 安装不上，报错包损坏? . . . 110
6.16 文字发生了重叠和拥挤，高 dpi 双屏切换后，文字错乱等问题。 . . . 110
6.17 英文单词换行时从中间断开 . . . 111
6.18 在目录中查找不到结果字段文件 . . . 112
6.19 文字背景出现颜色异常 . . . 114
6.20 升级软件后再次打开程序崩溃 . . . 115

7 在 Notepad-- 中使用 pinlang 116

7.1 Ndd 中 Pin 插件 . . . 116

7.1.1 Ndd 执行当前编辑框的 Pin 脚本 116
7.1.2 Ndd 执行 Pin 脚本插件 117
7.2 Ndd 支持操作编辑器的 Pin 方法 124
7.3 Ndd 内部方法函数 125
7.3.1 主程序窗口相关函数 125
7.3.2 编辑框页面相关函数 126

Chapter 1

软件简介

1.1 Ndd 名称由来和开发初衷

Notepad-- 国产轻量级文本编辑器，英文全称 Notepad-double-decrease, 代号 Notepad--，简称 Ndd。为何叫这个名字？还真有一段历史。

Ndd 软件作者网名“爬山虎”。为了保护个人隐私，或避免一些潜在的麻烦，笔者暂时只在网络上发言。2018 年，笔者使用的国外代码对比软件经常过期，于是开始自己开发一款国产代码对比软件 CCompare，以图满足个人代码对比需求。

然天有不测风云，有国外程序员主导开发的类文本编辑器 Notepad 系列软件，在国内拥有一批用户。不过该 Notepad 类软件作者，经常发布一些涉及“台独”和“疆独”的错误言论，被互联网上大批仁人志士所唾弃抵制。彼时许多人感叹，为何我泱泱中华大国，竟无人能够做出一款功能类似的替换软件呢？以此刹刹国外宵小之徒的威风？

听闻此情此景，作为一个程序员的笔者决定做些什么，用自身技术改变一些现状。“既然大公司瞧不上，小公司做不了，那就我来吧！”笔者在自己开发的 CCompare 基础上，进一步完善出 Notepad-- 国产文本编辑器软件。让国内正义热血之众，有更多的选择和替代方案。详细来龙去脉见笔者博文—— 《编写一个 Notepad--，去替换同类错误软件, 我是认真的》

Ndd 首次在 2019 年发布初始版 v1.7，经过三年多时间，从最开始的寂寂无闻，逐渐积累

用户，不断解决 bug 和完善功能。终于在 2022 年末的 Ndd v1.23 版本时，迎来了一波热度，众多相关网络媒体集中报道 Notepad-- ，一时 Ndd 被广大用户逐渐知晓。

Ndd v1.23 版本并不够完善，至少在当时还没有达到作者认可的成熟度，不过仍然引起了国外敌对势力的害怕和诋毁。境内外敌对势力，试图通过造谣、诋毁加“扣帽子”的方式，从舆论上攻击笔者和 Ndd。一时间，多个 Ip 显示在上海、香港或海外的傀儡互联网账号，在知乎或 github 平台同时发文造谣诋毁 Ndd。这是笔者没有意料到的事情。笔者纯粹做一款开源或商业软件，以商业竞争的方式去赢得市场，为什么会遭到一些不明势力的诋毁呢？

为了让国内外更多用户了解真相，笔者不得已发布相关澄清博文——见 gitee《Ndd 对诋毁者们恶意歪曲事实的说明》 。或见知乎 《Ndd 对诋毁者们恶意歪曲事实的说明》 。有部分喷子，连笔者的澄清都要喷。笔者实在不屑投入精力与一群乌合龌龊之众在网上争吵，后续将下来专心做好软件产品。笔者对 Notepad-- 中双减号含义的解释——**Notepad-- 的意义在于：减少一点错误言论，减少一点自以为是。**

截止到现在 2026 年 04 月 01 日为止，Notepad-- 最新版本为 v3.7.0，已经完善了许多特性和功能。最新版下载地址：gitee 最新版本下载地址

1.2 技术组件

Ndd 采用 C++ 语言编写，图形界面使用 Qt 库，支持 Windows/Linux/Mac 平台，可支持国产操作系统统信 Uos、麒麟 kylin 系统。编辑器核心组件使用了开源库 Scintilla。

有个经常容易被同行误会的事情，有些人既不了解也不调查，武断认为 Ndd 是 fork 其它开源库的代码，然后把名称改一改，就变成了现在的 Notepad-- 。笔者特别说明，这个说法是严重误导大众，相关澄清在博文——《关于 Notepad-- 》 。如果您也会编程，现在依旧可在 gitee 或 github 上看到 Ndd 的早期开源版本 v1.23 的代码，是不是直接 fork 或抄袭别人的代码，想必真相不难探究。

为什么要使用一些开源库，而不是从头完全开始呢？有些人还是会在网上说，归根结底还是使用了一些国外的组件或技术，并没有一笔一划全是国人开发。笔者认为，该问题并不是一朝一夕可解决的事情，完全从头开始，先不说技术难度的问题，光是工作量就非常巨大。

跨平台的图形界面库, 全世界做得好的，能够真正满足速度和跨平台特性的界面图形库，选择的余地本来就不多，Qt 可算是其中佼佼者，而国内在这方面几乎还是空白。该问题并不是喊喊口号就能立地解决，可能需要几代人的不懈努力才能彻底攻克并赶上欧美。Qt 和 Scintilla 开源组件都是源码开放、具备自主可控性，所以基于这些开源组件所开发的 Ndd 也具备自主可控性。

1.3 Ndd 不会涉及政治言论

Ndd 的作者遵守中国法律，不会发表政治言论。维护国家安全，遵守中国法律，是作为一个中国人，最起码应该做到的事情。Ndd 不会涉及到政治，也不会发布政治言论，不主动讨论“政治”的言论。Ndd 作者是一个老百姓，只想认真做产品，不会发表涉及“政治”的言论。

那么为什么 Ndd 会有一些涉及到看起来有“政治”的言论呢？那是因为在一些竞品作者发布“错误言论”的时候，许多用户站出来拥护 Ndd 去替换反对软件，其中难免涉及到一些个人情绪或爱国的言论，Ndd 被躺枪了。还有就是，反对软件的一些代言人，在网上恶意发帖造谣 Ndd，给 Ndd 扣一些“帽子”。为防止一些不明真相的用户被误导，Ndd 迫不得已做一些澄清。

因为利益关系，过去当某些竞品发表错误言论的时候，会不自然的给它们带来大量的流量和利益。但是因为 Ndd 的存在，现在当竞品作者发表错误言论的时候，那些以前的流量，会跑到 Ndd 这儿来了。所以一些竞品软件和它们背后的代言势力，因为利益流失而开始埋怨 Ndd。这部分乌合之众，不敢再肆意妄为的瞎发表言论，就把火迁怒在 Ndd 的身上，进而开始对 Ndd 造谣抹黑。希望广大用户能够明白其中的道理。

笔者重申: Notepad-- 不讨论且不涉及任何政治问题，开发 Ndd 是为让国人对编辑器软件有一种更多的选择。因为 Ndd 的存在，让跳梁小丑减少错误言论，直至闭嘴，这是笔者开发 Ndd 的附加收益之一。

最后说明，笔者完全彻底地认同“台湾是中国不可分割的一部分”，这是每个中国人都认同的基本事实，更是中国《反分裂国家法》规定的法律条文，并不是笔者发表的政治观点。笔者作为一个中国人，遵守和认同本国的《反分裂国家法》。有部分极端人士，反对笔者支持该

法律条文，笔者无论如何都不能同意他们的观点。笔者本着“治病救人”的出发点，劝导这一小撮愚昧之人，好好看看《中国近代史》，或许他们会有新的认知。

1.4 后续还会继续开源吗？

Ndd 过去一共开源过八个版本，分别是 v1.17-v1.23/v2.0。现在最新版本的 Ndd 并没有继续开源。当前 Ndd 暂时不再提供源代码，后续视情况可能会在某天完全开源。为什么这样呢？

笔者最开始开源 Ndd 有三个初衷：一是开放自己的源码，让更多的国人或同行参与进来，以贡献功能代码，加速软件功能完善；二是通过开源获得一些合法捐赠收入，以改善经济条件，让笔者有更多精力投入开发；三是上架 windows 市场或 macos 市场，占有一部分市场形成既定市场地位后，再彻底开源代码形成技术壁垒。

但是经过一些列实践和打击后，笔者发现之前的想法”too young, too simple”。开源出来的 v1.17 到 v2.0 八个版本代码，并没有吸引一些前来贡献稳定功能的同行，反而是招惹来一些“嘲笑者”、“索取者”或“谩骂者”。有人公开在网上批评说笔者的代码写的非常垃圾；有人说笔者应该把 CCompare 的插件对比功能也完全开源；更过分的是，有人故意误导大众，说笔者代码是直接抄袭后改名，收捐赠是乞讨，是骗取大众的赞赏费。可以肯定，这其中一定有一些来自竞争对手软件的走狗势力，也有一些国内眼红的同行。

Ndd 该做的事情还是笔者主要在做，很少有人解决 bug 或合入稳定可靠的功能或插件。更有甚至，个别极端的提交代码者，因为代码存在大量低级错误被拒绝合入后，不去完善自己的代码，反而是迁怒笔者；个别开源需求提交者，因为笔者暂时拒绝开发他们提出的需求，索性直接在开源项目上辱骂。

还有一些对编程知识一知半解却自认为高明的人，“指导”笔者说“这个功能做的烂，应该这样这样... 之类的话。”，笔者一旦以专业角度稍微阐明原理，此类“高明人士”便开始勃然大怒，“我当年写代码时你还在...”，这类人士完全不可理喻，实在没有一个沟通讨论的氛围。

相比之前，笔者不仅没有减轻开发工作量，反而还要投入一部分时间去完善存在大量低

级问题的提交，甚至还要面对一些提交者的谩骂。国内开发者大都既卷又忙，家庭生活压力大，很少有人认真投入时间参与开源开发，这是可以理解的；但是居然招来一部分谩骂者和嘲笑者，令笔者始料未及。

此外，在代码开源 2 年后，从 gitee 上共计收到捐赠 150 多笔，总金额不到 3000 元。想必要靠开源来养活自己，在国内怕是天方夜谭。最后，在国内上架 windows 或 macos 商城时，也遇到一些问题，包括一些资质或备案，还有开发者账号申请失败等问题，可谓壁垒重重。如果 Notepad-- 完全开源，怕是立刻就被抄袭发布到 macos 商城或 windows 商城中去了，压根轮不到解决不了资质问题的笔者。

在国内 IT 用工环境和普遍待遇没有提高到一定阶段，或者开发者素质整体没有上升到一定境界时，笔者对个人开发者做开源还是持劝退态度。个人同行还是老老实实先搞点副业养活家庭后，再去开源修功德吧。

经此种种深刻教训后，笔者痛定思痛，决定放弃所谓国内开源，摆脱一些无意义的争吵，还是做款简单个人软件，业余时间做一做开发，95% 的功能均免费，5% 的高级功能提示付费。通过商业行为售卖符合国情的平价注册码，同时接收部分热心人士的小额捐赠。保持初心，做一个普通开发者即可，没必要把自己摆在多么重要或高尚的位置，如此免遭一些恶意者的道德绑架。笔者将继续沉下心来做软件开发，回归“给国内仁人志士提供一种替代选择”的初心。

笔者在此要感谢一些热心开发者和捐赠者，他们提供了有价值的代码（比如 json view 插件开发者 LanZhao），或者捐赠后留言鼓励笔者。这些举动让笔者有了强大的开发动力，同时让笔者感到：虽然笔者的个人力量是有限的，但开展的工作却是有意义的。走国产可替代之路，光明正大，前途无量，与同行共勉！

2024 年 9 月 20 日

Chapter 2

Ndd 软件功能介绍

2.1 下载与更新软件

在软件主界面，点击 **菜单-关于**，弹出下图:



图 2.1: 菜单-关于页面

下载方式一，直接打开网址:

国内建议使用 gitee，国外使用 github。

国内的 release 版本下载地址：

https://gitee.com/cxasm/notepad-/releases

百度网盘（虽然慢，但是真免费）

https://pan.baidu.com/s/5MpFGSKtiH0wHb2_H-onYSw

百度网盘下面，根据不同的操作系统对版本进行了归纳，方便用户选择自己合适的版本。如下图所示



图 2.2: 百度网盘下载界面

国内的免费网盘 123 下载地址 (不再推荐，限流加不登录不能下载)

https://www.123pan.com/s/DkxzVv-BwJVh.html 123 网盘下面，根据不同的操作系统对版本进行了归纳，方便用户选择自己合适的版本。如下图所示

备注：123 云盘的下载速度在国内较快，但是好像也存在不登录则不能下载的问题，具体



图 2.3: 123 盘下载界面

得试一下。

国外使用 github 下载地址：

https://github.com/cxasm/notepad–

备注：如果 gitee 无法下载，可能是需要注册 gitee 账号，如果不怕麻烦，则注册一个，顺便给作者点亮星星。github 是不需要登录就可以下载的，但是国内网速不一定能够每次打开。

2.1.1 为什么不做自动下载更新

1）自动下载需要后台有服务器存放资源，目前国内没有免费的服务器资源。比如 gitee，123 云盘，都是需要登录才能下载的，该类网站都有反盗链接的限制，不登录没法后台直接下载。国外的 github 可以直接下载，但是因为国内的访问限制，也无法保证能够下载到。

2）Ndd 目前没有自己搭建服务器。Ndd 主打就是免费使用，虽然有少部分用户赞赏，但目前作者还是无法承受购买稳定快速的云服务器。

3）Ndd 是绿色软件，没有广告，关闭就是退出。自动下载，自动更新，在笔者看来都是在背后偷偷做一些事情，或者试图霸占用户的电脑资源，笔者本身不喜欢这类软件行为。如果有新版本，则这个关于里面，会有提示。如下图所示：

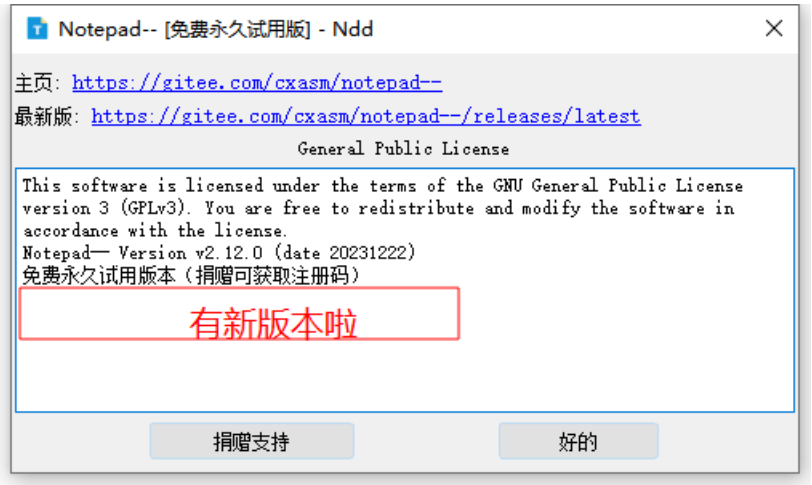


图 2.4: 关于页面中新版本提示

2.1.2 Ndd 可支持的操作系统

1. windows

   支持 win7 win10 win11 等操作系统。目前只支持 x64 架构，x32 不再支持。

2. Macos 系统

   可支持 mac10.3 mac12.3 及以后的操作系统。因为 Mac 目前有 x64 intel 和 mac-M 芯片，有 mac10/mac12/mac14 等众多系统版本，macos 自身彼此前后版本的兼容性较差。笔者推荐 Mac10/11 的老系统，和 Mac12/13/14 的新系统，各自使用对应的 Ndd 版本。在 2023 年 11 月以后，Ndd2.11 及以后，可能不再支持 Mac11 及之前的老系统。现在 Ndd 已经推出 Mac M 芯片相关的安装包。

3. Linux 系统

   因为 Linux 系统发布版太多太杂，无法支持一些小众发布版。目前支持的 Linux 系统有：国产 Uos 系统，国产麒麟系统，这部分主要支持 x64 架构的 cpu，也支持部分 arm64 的 cpu。

另外，对于 ubuntu, redhat 等操作系统，会不定期出版本，因为这些系统是国外 linux 产品，在这类系统上使用 Ndd 的人不多，所以这些用户只能自行适配 ndd 的安装包，或者找之前发布的 Ndd 的老版本。

其余版本，比如 uos/麒麟/redhat，不定期发布，云盘连接如下:

https://www.123pan.com/s/DkxzVv-BwJVh.html

subtwo-v2.x-redhat7.4-x86_64.AppImage 这个是在 redhat74 下面制作的 appimage 版，虽然体积较大，但是依赖性和兼容性会比较好，理论上是可以在大部分 linux 上直接执行。

麒麟操作系统，可以直接使用 uos 的发布版本；后续如果没有找到 kinly 的发布版本，还请直接使用 uos 对应的架构版本即可。

2.2 主体功能一览

1. 菜单-编辑界面中，有大量的行编辑、空白字符等操作菜单，如下图所示：

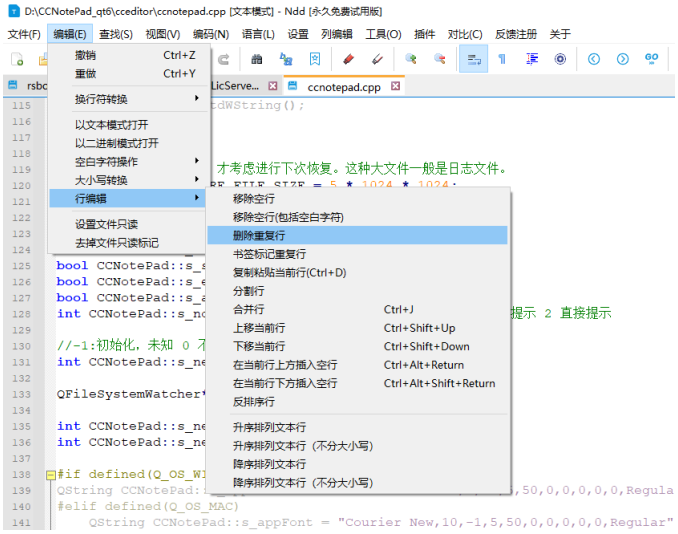


图 2.5: 菜单-编辑界面

2. 文件 Tab 标签上鼠标右键，也有一些特性功能，如下：

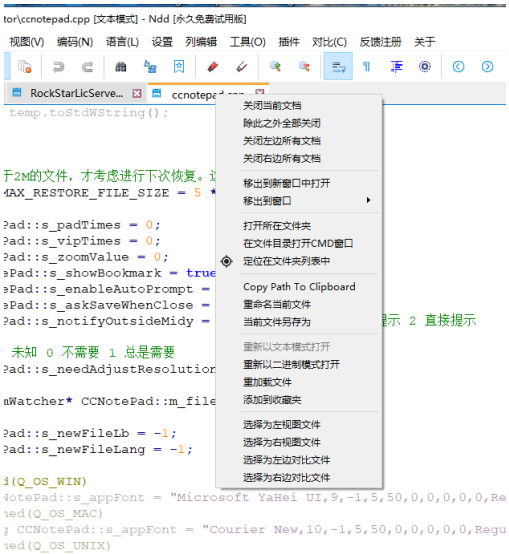


图 2.6: 文件标签右键

3. 在编辑框中的鼠标右键，也有一些快捷功能：

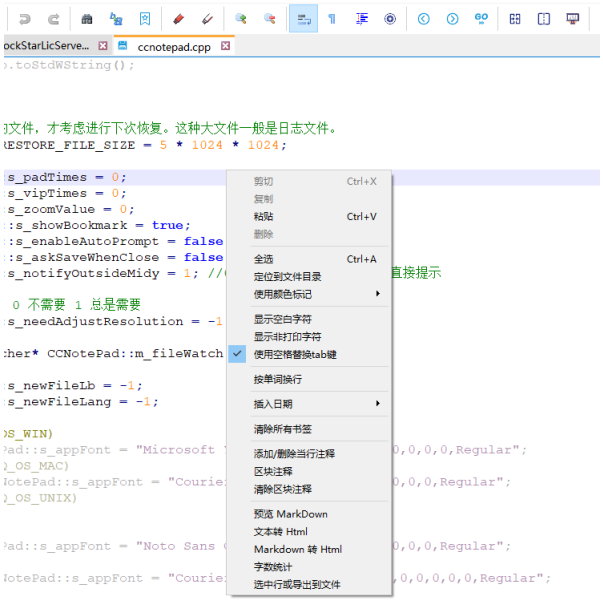


图 2.7: 当前编辑框中右键

4. 设置里面有软件字体、保存恢复相关的设置：操作见2.4

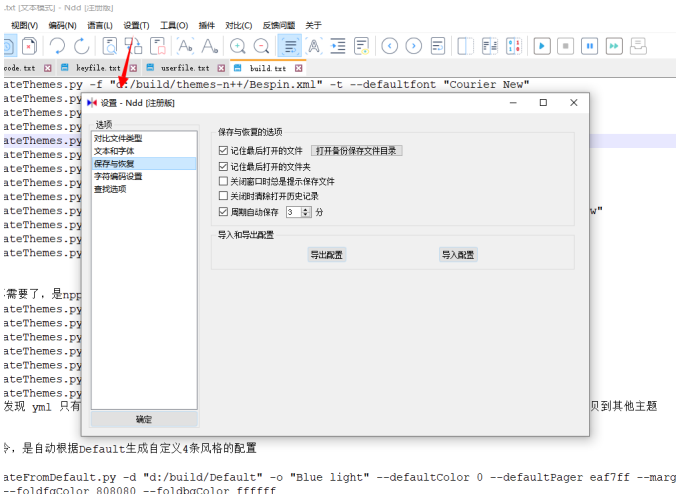


图 2.8: 设置界面

5. 修改界面主题、语法高亮颜色：操作见3.1

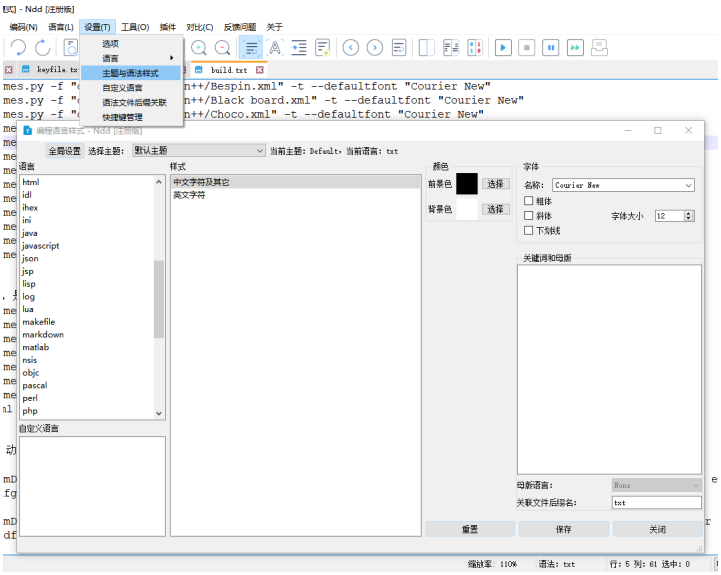
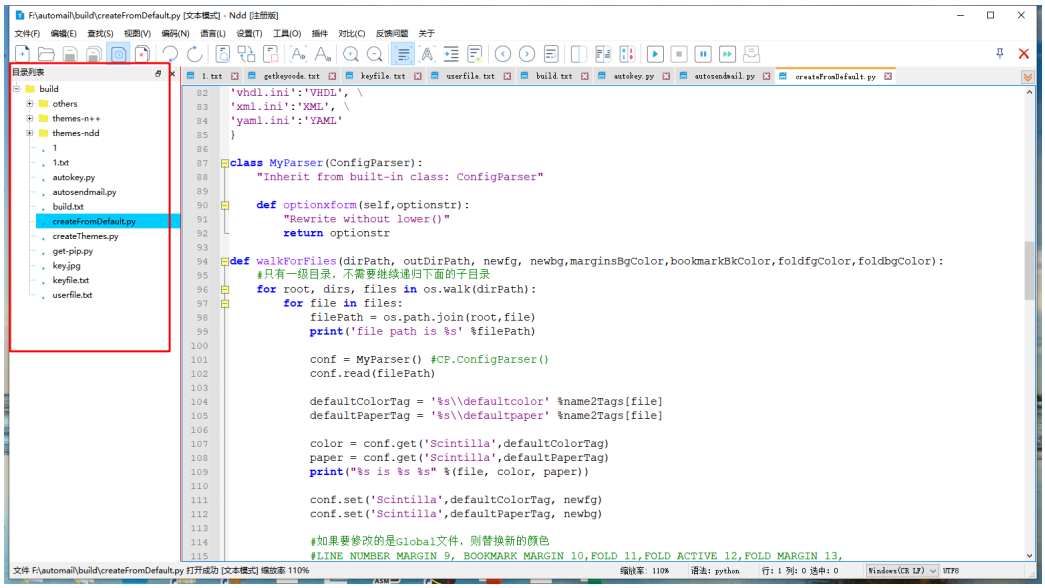


图 2.9: 界面主题、语法高亮

6. 文件夹或文件列表开启和关闭功能: 操作见3.9



```
'vhdl.ini':'VHDL', \
'xml.ini':'XML', \
'yaml.ini':'YAML'
}

class MyParser(ConfigParser):
    "Inherit from built-in class: ConfigParser"

    def optionxform(self,optionstr):
        "Rewrite without lower()"
        return optionstr

def walkForFiles(dirPath, outDirPath, newfg, newbg,marginsBgColor,bookmarkBkColor,foldfgColor,foldbgColor):
    #只有一级目录，不需要继续递归下面的子目录
    for root, dirs, files in os.walk(dirPath):
        for file in files:
            filePath = os.path.join(root,file)
            print('file path is %s' %filePath)

            conf = MyParser() #CP.ConfigParser()
            conf.read(filePath)

            defaultColorTag = '%s\\defaultcolor' %name2Tags[file]
            defaultPaperTag = '%s\\defaultpaper' %name2Tags[file]

            color = conf.get('Scintilla',defaultColorTag)
            paper = conf.get('Scintilla',defaultPaperTag)
            print("%s is %s %s" %(file, color, paper))

            conf.set('Scintilla',defaultColorTag, newfg)
            conf.set('Scintilla',defaultPaperTag, newbg)

            #如果要修改的是Global文件，则替换新的颜色
            #LINE_NUMBER_MARGIN 9, BOOKMARK_MARGIN 10,FOLD 11,FOLD_ACTIVE 12,FOLD_MARGIN 13,
```

图 2.10: 开启文件夹、文件列表

7. 宏功能,Ndd 支持宏功能，在工具栏上面：操作见3.13

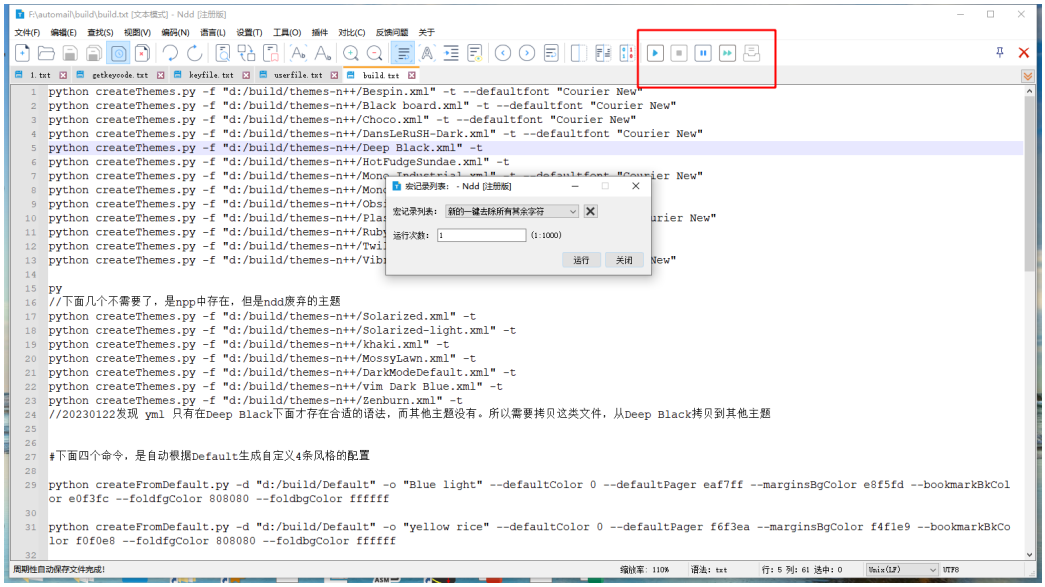


图 2.11: 开启文件夹、文件列表

8. 文件夹/文件对比，来自 CCompare，已经集成在 Ndd 中：

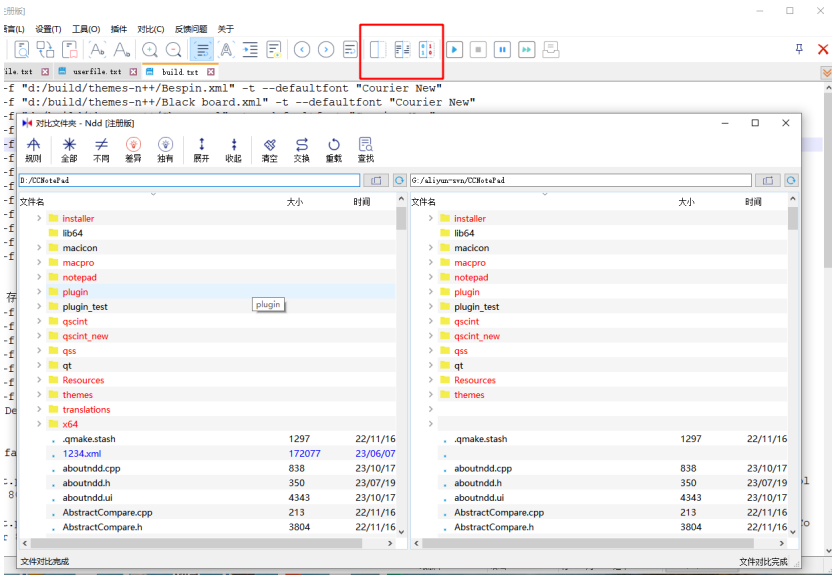


图 2.12: 文件对比

9. 插件功能：

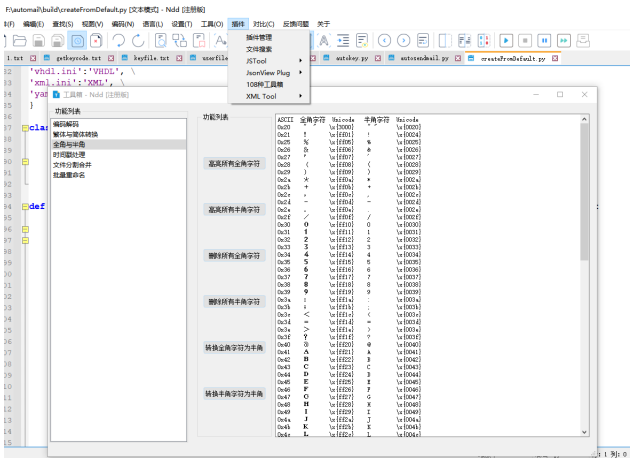


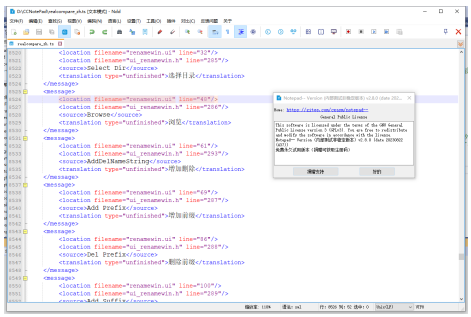
图 2.13: 插件

10. 捐赠作者与注册正版：

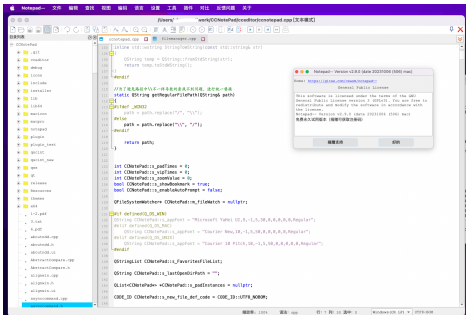


图 2.14: 捐赠与注册

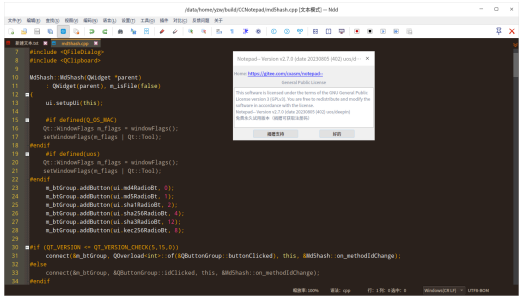
2.2.1 windows/Linux/Macos/UOS/麒麟效果图



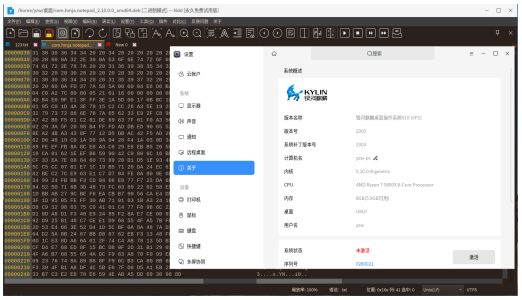
(a) windows 效果图



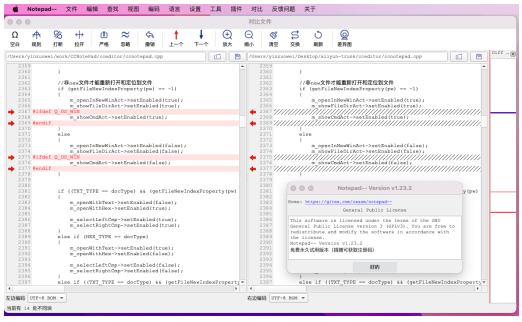
(b) macos 效果图



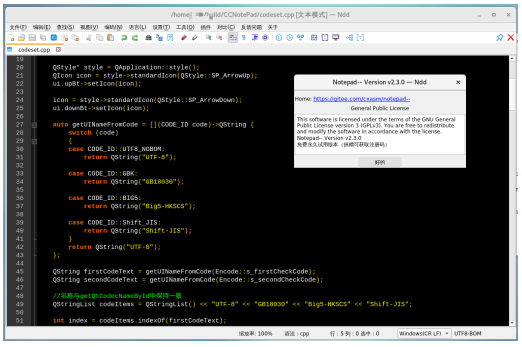
(c) uos 或深度效果图



(d) kinly 麒麟效果图



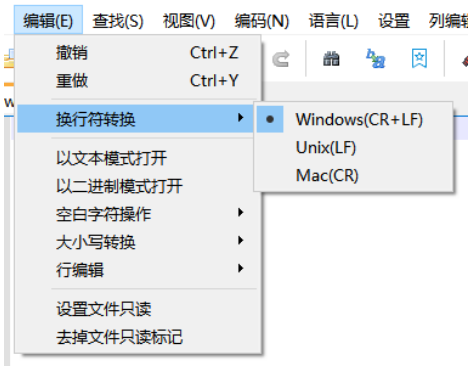
(e) 文件对比图



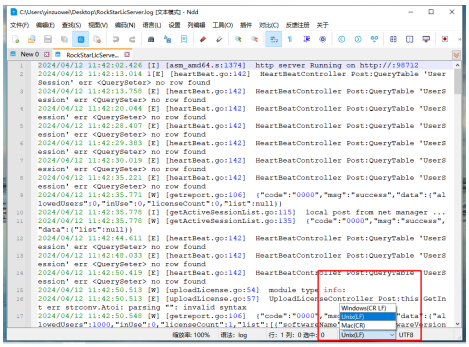
(f) redhat7 系列效果图

2.2.2 换行符号转换

windows 下默认换行符号是\r\n, Macos 或 linux 下默认换行符号是\n。打开一个文档后，如果需要对当前文档换行符号进行转换，则可通过如下两种方式进行：



(g) 菜单中切换换行符



(h) 状态栏中切换

如果当前文档比较大，则以上操作可能转换符号需要较长时间，或者转换会拒绝。此时可通过“万能文本转换器”执行换行符号转换，在菜单 **工具-万能文本处理器**，见下图所示：

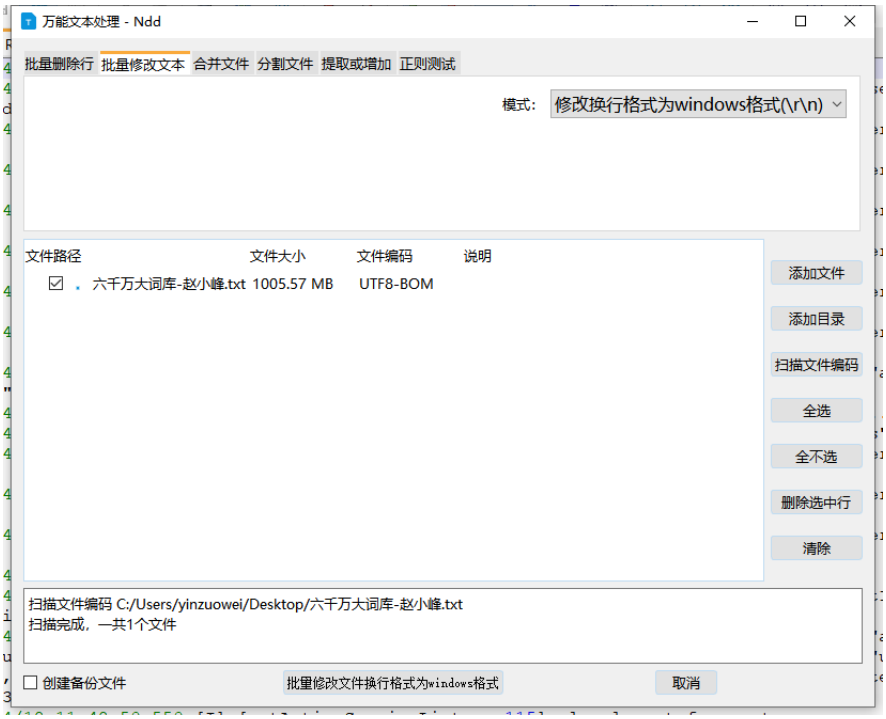
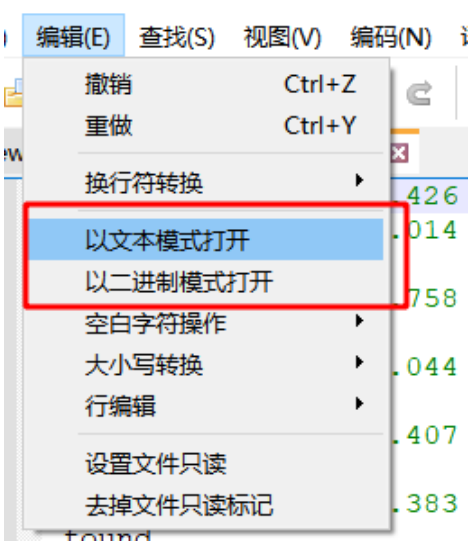


图 2.15: 万能文本处理中转换换行符

2.2.3 切换文件打开方式

打开文件后，可能显示为二进制格式，如果想用文本打开，如何切换？反之打开为文本格式，想要使用二进制打开，又该如何？有以下两个地方，均可对已经打开的文件，进行打开模式切换，如下图所示：



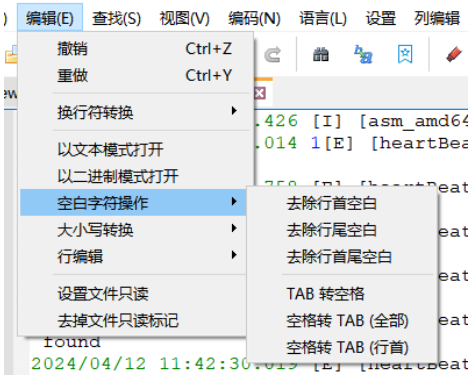
(a) 菜单中切换打开方式



(b) 右键 tab 文件标签切换打开方式

2.2.4 去除文件中空白，大小写转换

有时候需要去除文件中行首或行尾的空白。又或者选中文本后，对文本进行大小写转换。操作界面如下：



(c) 去除行中空白操作



(d) 选中文本大小写转换

还有上文提过的“万能文本处理”也有部分去除空白、数字的功能。

2.2.5 显示文件列表、显示空白字符、显示网址高亮

以上功能均在菜单 **视图**中设置，界面如下：



图 2.16: 显示或隐藏部分功能

2.2.6 文件 Md5 及各种哈希码计算

该功能在菜单 **工具-MD5/SHA** 中，界面如下：

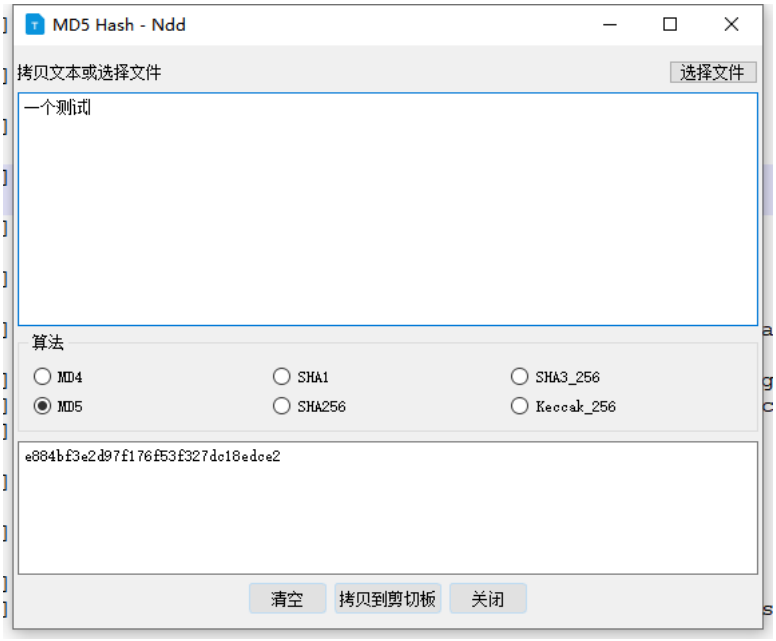


图 2.17: 文件 MD5/SHA 计算

2.2.7 Base64/Url 工具

该功能在菜单 **工具-Base64/Url** 中，界面如下：

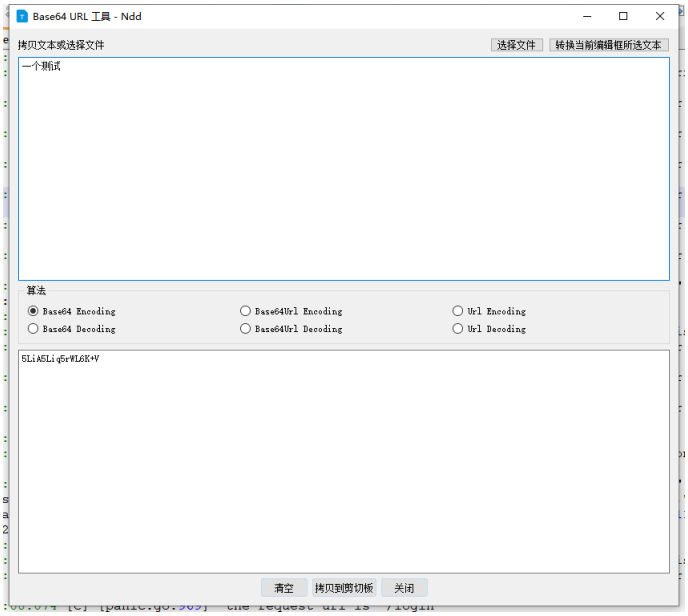
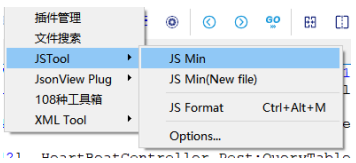


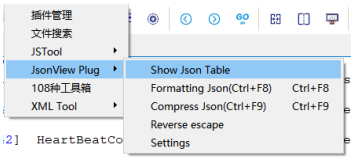
图 2.18: Base64/Url 工具

2.2.8 json-js-xml 格式化工具

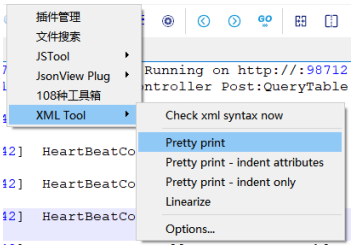
该功能在菜单 **插件**中，界面如下：



(a) js 格式化与压缩



(b) json 格式化工具



(c) xml 格式化工具

图 2.19: js json xml 格式化工具

2.3 安装配置指导

2.3.1 Macos 下如何安装 dmg 安装包

安装时双击 dmg 包，此时看到 dmg 里面有个 Notepad-- 程序，不要双击该程序。打开访达，找到应用程序，然后将 Notepad-- 程序直接拖进应用程序访达目录中，这样就安装成功了。备注：这是 macos 下安装应用程序的常见方法。

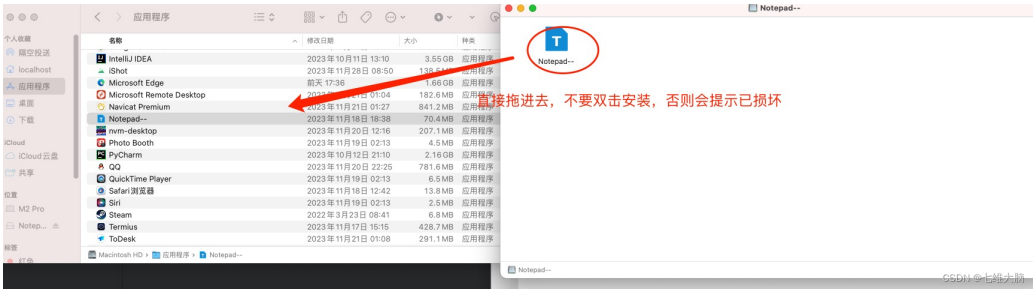


图 2.20: macos 安装方法

2.3.2 Macos 安装包损坏报错解决

参考网文地址：Macos 安装包损坏报错解决

Macos 安装 Ndd 的 dmg 时, 可能会遇到提示已损坏，如下图：



图 2.21: macos 报错

报错原因是因为 Ndd 没有使用 macos 开发者账号对应用程序进行签名，在国内申请个人开发者账号目前比较困难。

解决过程：

1. MacOs 安装 Ndd 报错解决方式如下

   ① 首先拿到 dmg 安装包后，先双击打开 dmg，会呈现 dmg 包中的 Notepad-- 。

   此时千万不要双击安装!!! 需要打开访达-找到应用程序，然后将上面的 Nodepad-图标直接拖进应用程序目录，这样就安装成功了，如下图所示：

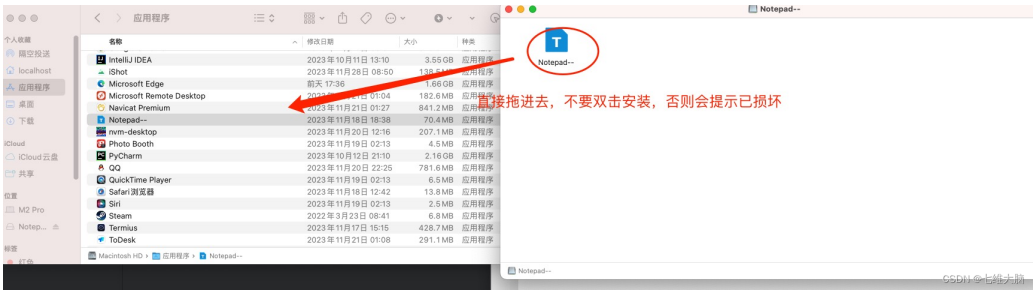


图 2.22: macos 安装方法

② 接着打开一个终端命令行，输入以下命令，然后按下回车，并输入电脑的开机密码。

sudo xattr -cr /Applications/Notepad–.app

按照以上两步操作后，即可顺利打开 Ndd 。此后老的 Dmg 包可以卸载或删除。

2.4 界面字体或图标太小的调整方法

有用户反馈，在高清屏幕下，界面的字体看起来非常小，操作不方便，下面介绍设置方法。

1. 调整界面字体大小

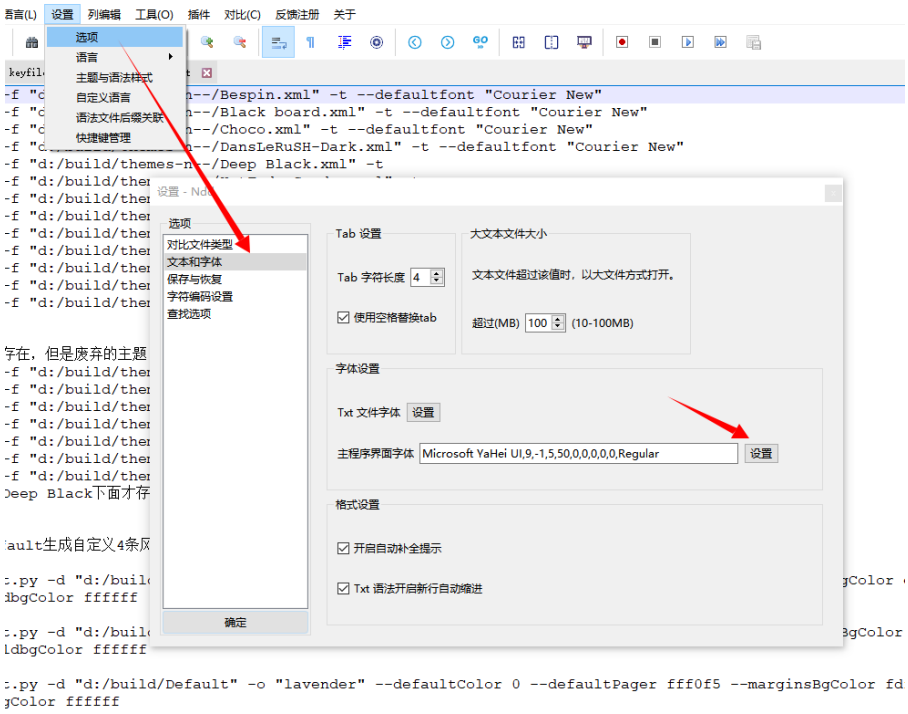


图 2.23: 调整界面字体大小

建议大家不要选择小众字体，在 windows 下选择微软雅黑字体，然后调大字号即可。

2. 调整工具栏图标大小

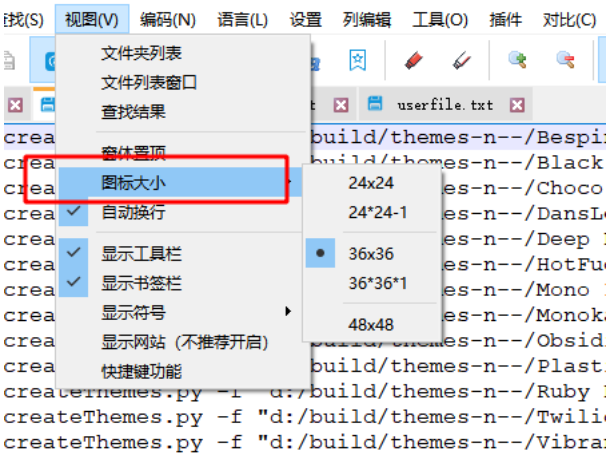


图 2.24: 调整图标大小

2.5 选择 Qt5 还是 Qt6 版本

目前 windows 和 macos 下的 Ndd 有 qt5 和 qt6 两个版本。到底该如何选择呢？

windows 下的 Qt5 版本可以支持 win7/win10/win11 系统，如果您是普通屏幕，非 2K 或以上分辨率的屏幕，使用 Qt5 版本即可。需要说明的是，windows 下只提供 x64 版本，不提供 32 位的老旧系统。反之如果您的屏幕是 2k 等高清大屏，当使用 Qt5 版本时如果界面呈现太小，则建议选择 Qt6 版本。

macos 下面提供的 x64 cpu 版本是基于 Qt5，其兼容性更好，可以在 x64 和 M 系列芯片运行，最低可在 Macos 10.12 的系统上运行。Arm64 cpu 的版本是基于 Qt6，只能在 M 系列的 mac 电脑上运行，不能在 x64 架构 mac 电脑上运行。经常有用户说，他使用的是新款 arm64 M 系列芯片，担心使用 x64 的 Ndd 程序速度会慢，其实不必担心此问题，Ndd 是采用 C++ 编写，其运行速度比使用 JavaScript 实现的 vscode 要快几个数量级。如果您不太懂如何绕过 M 系列的签名限制，大可直接选择 x64 的 Ndd。x64 Ndd 在 Macos 上只需要在设置中允许未签名的程序运行，即可直接运行。

Chapter 3

常用功能或使用技巧

3.1 界面主题和配色

有用户说，默认主题 Ndd 是灰白色的配色，而他喜欢在暗黑色下面使用软件，如何调整 Ndd 到暗黑模式呢？

依次点击**菜单-设置-主题与语法样式**，弹出如下界面框：

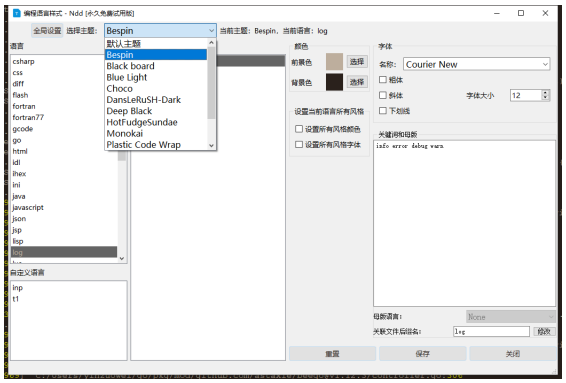


图 3.1: 切换软件界面主题

在上图3.1中，选择暗黑色系的主体，比如 **Bespin** 主题即可，其暗黑效果如下:

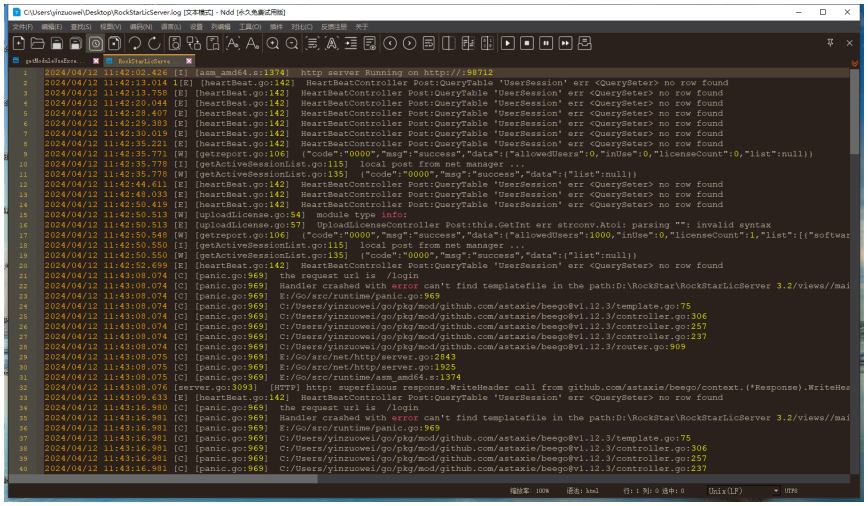
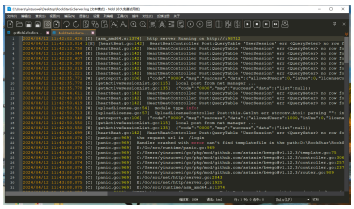
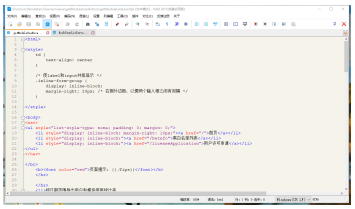


图 3.2: Bespin 主题效果

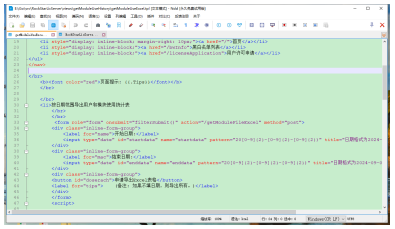
您可以选择自己喜欢的主题配色，需要说明的是，有些语法在某些主题下面可能缺少配色，此时可以换一种主题，或者自行调整对应语法下面的配色即可，具体方法见 3.2 调整字体或语法配色。以下展示几种主题效果如下：



(a) Monokai 效果图



(b) Yellow Rice 效果图



(c) Bean Green 效果图

图 3.3: 几种主题效果展示

3.1.1 标题栏修改为暗黑色

在 windows 系统下，通过上述方法修改 Ndd 为暗黑主体后，有用户说，这样改是可以了，不过窗口的标题栏还是白色的，看起来不协调，能否把标题栏颜色也修改呢？

该问题在 window 系统中设置即可。系统本身有提供标题栏的颜色设置，如果您是在暗

黑模式系统下，则在 windows 系统自行设置所有标题栏的颜色即可。如下图所示：

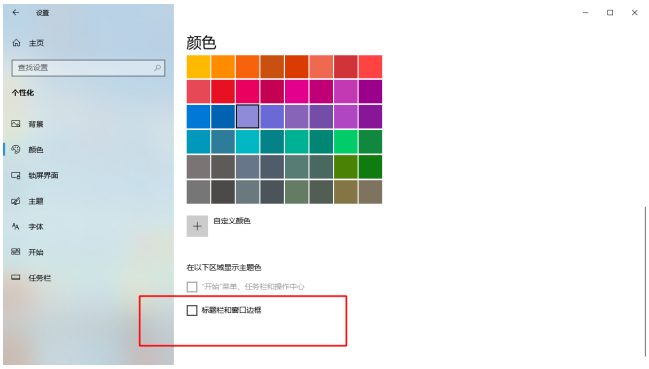


图 3.4: windows 设置标题栏颜色

本例中把标题栏设置为暗棕色，则标题栏显示颜色如下：

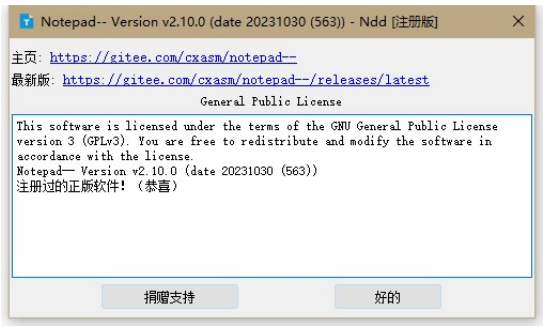


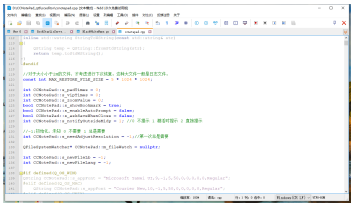
图 3.5: windows 棕色标题栏效果

macos 下则压根不存在该问题，因为 macos 的标题栏和窗口是分离的，标题栏菜单显示在桌面的最顶层，其颜色是自动跟随 macos 系统。

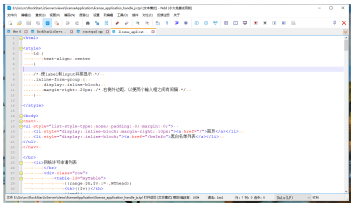
3.2 调整字体或语法配色

Ndd 是一款纯粹的文本编辑器，不是 ide 开发软件。不过 Ndd 内置了常见编程语法的高亮显示，比如在您打开 C/C++/Python 等编程语法文件时，Ndd 会自动识别文件的后缀类

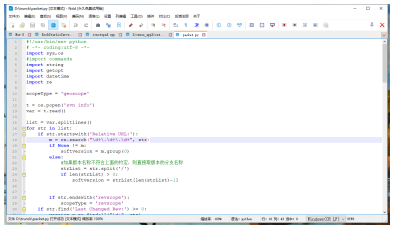
型，然后选择合适的语法，对源码文件进行语法高亮。默认情况下，如果是一个单纯以.txt 类型的文本文件，则默认使用 txt 语法高亮设置。如下所示，当打开 C++/Python/html 等编码文件时，Ndd 的语法高亮显示效果：



(a) C++ 语法高亮效果图



(b) Html 语法高亮效果图



(c) Python 语法高亮效果图

图 3.6: 几种编程语法效果展示

当一些程序设计人员，对常见的编程源码进行查看时，因为可以对语言进行语法关键字高亮着色，给人更好的体验感。有许多人建议 Ndd 继续扩展 ide 的功能，争取做到和 vscode 一样，笔者考虑过还是战略放弃该道路。因为不存在一种“放之四海而皆好用”的 ide, 几乎每种编程语言都有它专用的 IDE 开发环境。尝试做一个“全能型”的 ide，那一定是一个庞然大物，而且速度慢，大而杂，笔者放弃这样的发展方向。

3.2.1 设置 txt 文件语法对应的字体

对于广大的非编程人员，使用 Ndd 就是打开一个 Txt 文件的用户来说，默认编辑器使用 txt 语法。有人说，这个东东到底有什么用呢？举个例子，如果要把普通 txt 文本文件中，中文设置为黑体字体，颜色为黑色; 英文设置为幼圆，字体为蓝色的显示效果，具体效果如下图所示：

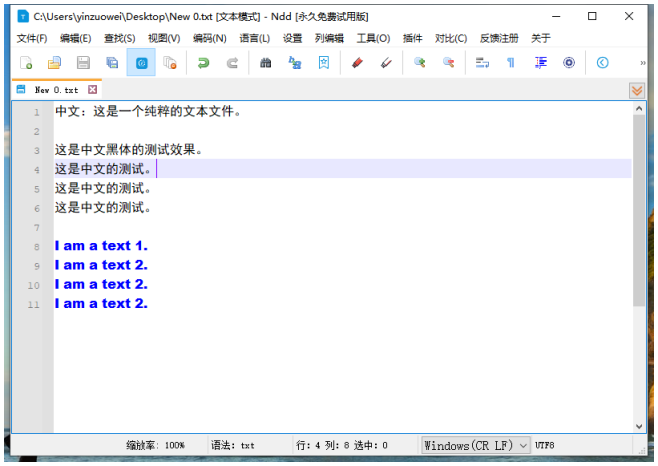
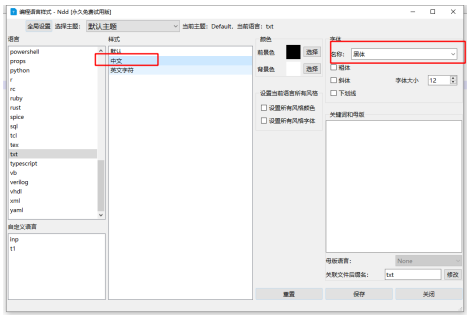
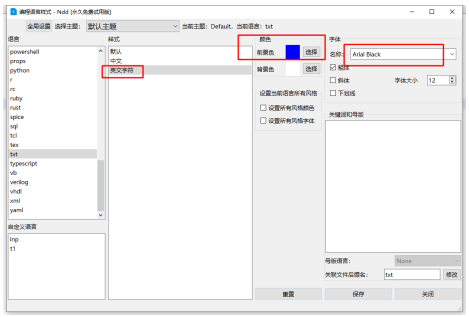


图 3.7: txt 语法展示

上图3.7是最终效果，那么该如何操作呢？依次点击 **菜单-设置-主题与语法样式**，语言选择 txt(会自动选择对应语法)，在样式中依次对中文和英文字符，进行字体、前景色的一些列设置即可，操作界面如下：



(a) 修改中文字段样式



(b) 修改英文字段样式

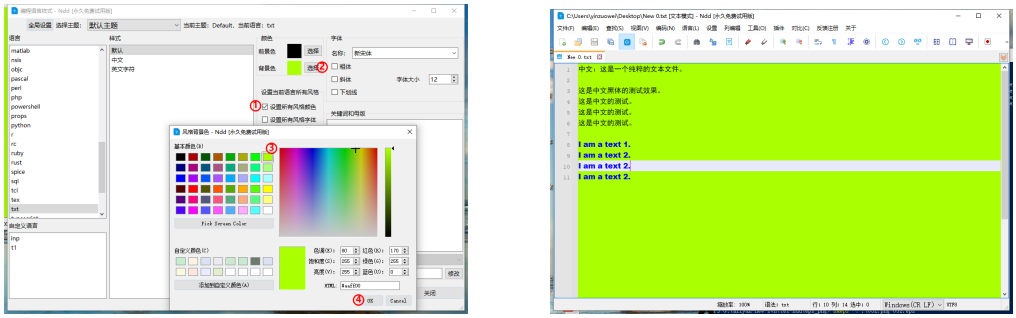
图 3.8: 修改 txt 语法中文和英文的着色方案

最后点击保存即可得到以上效果。**特别说明：笔者特别不建议大家选择小众字体，因为小众字体在某些情况下，计算字体的宽度和高度的时候，容易出现计算错误，导致显示看起来重叠紊乱。**

除此以外呢，在该设置界面中，还有一个全局设置，会影响所有语法的配色和一些查找结果的配色，后续在专门介绍该内容的时候会展开说明。

3.2.2 设置 txt 文件显示的背景色

再举一个例子，有用户说，希望调整当前 txt 语法页面显示的背景颜色，换成自己喜欢的背景色效果。在上面例子的基础上，笔者再做一个例子，把 txt 语法下面的背景色调整为亮绿色，依次如下左图所示，依次操作图中的①②③④后，即可得到下右图的效果。



(a) 修改 txt 背景颜色 (b) txt 亮绿背景效果图

图 3.9: 修改 txt 语法中文和英文的着色方案

说明，上图第①步，勾选“设置所有风格颜色”，其作用范围是当前 txt 语法的所有样式，如果不勾选该选项，则需要对“默认、中文、英文”三种样式分别依次做以上设置，如此操作要麻烦一些。提供该“设置所有风格颜色”，即是一次性把设置的颜色，应用在当前语法的所有样式上。备注：该界面是以 2.18 版本做讲解，有些老版本可能缺少对应选项。

细心的用户可能会发现，只是编辑框的颜色被修改了，而左边的线号和折叠部分的颜色没有被修改，这里也是可以修改的，见3.5 行号与书签风格。就是在前文说过的全局设置里面修改。后面讲解全局设置时，再详细介绍。

3.2.3 修改编程语言的显示配色

上面几个例子，已经针对 txt 文本的显示配色做了详细讲解。笔者再以 cpp 语言举例，介绍一下 cpp 显示样式是如何在 Ndd 中修改的。

基本概念： txt 文本，本来是没有语法样式的，笔者针对 txt 文本在国内有中文和英文字符的特性，针对中文和英文两种样式可做显示着色设置。对编程语言而言，不同的编程语言

有它自己的特性字段，比如 c++ 语言的样式风格如下：

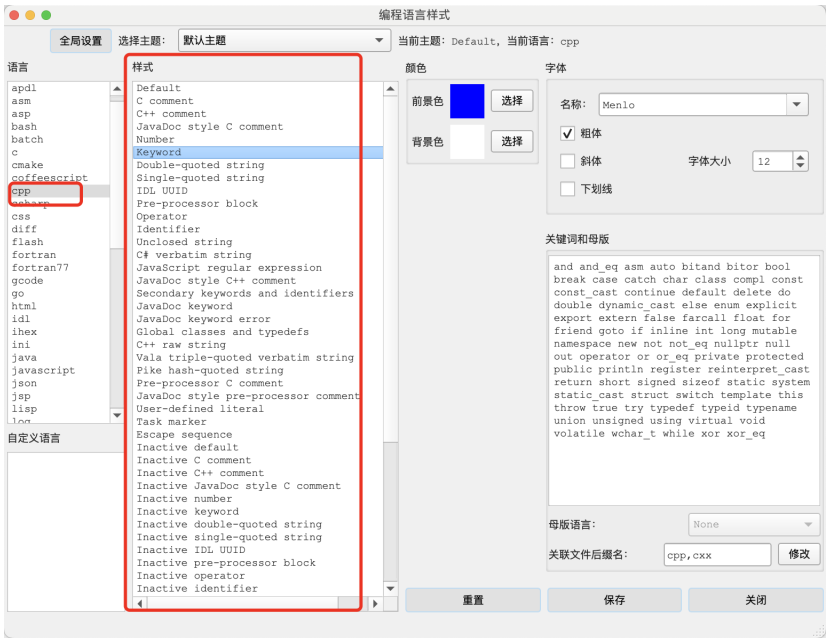
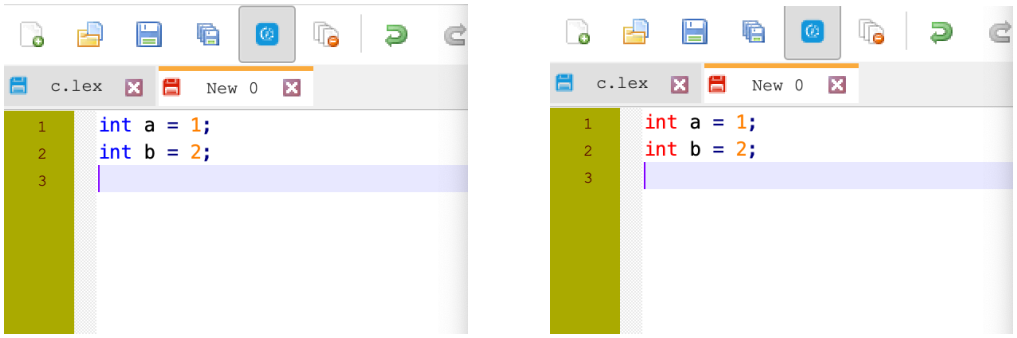


图 3.10: cpp 语言的样式设置

上图中 cpp 语言的各种编程特性样式，均可以自行修改字体和前景背景颜色，字体样式等。比如笔者针对 cpp 中的关键字，分别设置背景色为蓝色和红色后，其效果如下：



(a) 修改 cpp 关键字为蓝色　　(b) 修改 cpp 关键字为红色

图 3.11: 修改 cpp 语法中关键字的配色

其它样式的修改操作类似。**备注：**有时候设置字体为粗体会无效，那是因为系统中有些

字体缺少粗体。

3.3 换行符的查找或替换

一些对电脑不太熟悉的用户，在使用 Ndd 时可能对换行符号的相关操作感到迷糊。加群上来就噼里啪啦的说：“我在查找框中，使用\r\n 压根找不到字符，而其他同类软件是可以滴 ~”。现象如下图所示：



图 3.12: 菜单-编辑界面

对文档的换行符号简单介绍一下，文档中的行之所以能够换行，那是因为遇到了\r\n 或\n 这两类特定字符。所以一般来说，文档的换行字符格式有以上两种格式，而 windows macos linux 等操作系统，其给定的默认换行字符，略有不同。

1. **windows 系统下面的换行符**

   一般来说，windows 新建一个文档后，给定的默认换行字符类型是\r\n。这里说的是默认情况，用户后续也可手动把换行字符切换为\n。

2. **macos 或 linux 系统下面的换行符**

   macos 或 linux 这类 Unix 系统，其新建一个文档后，给定的默认换行字符类型是\n，即没有前面的字符\r。需要说明的是，在早期的 macos 中，其给定的默认换行字符是\r，

后续 mac 公司为了和 linux 保持统一，默认换行由**\r** 切换为**\n**，所以这里还有一段隐藏的背后历史。

3. **如何确定文档目前的换行字符**

那么问题来了，当打开一篇文本文档后，如何确定文档当前的换行字符类型呢？使用 Ndd 打开文档后，界面有如下提示, 下图红色箭头标记处，告诉用户当前换行的符号类型。

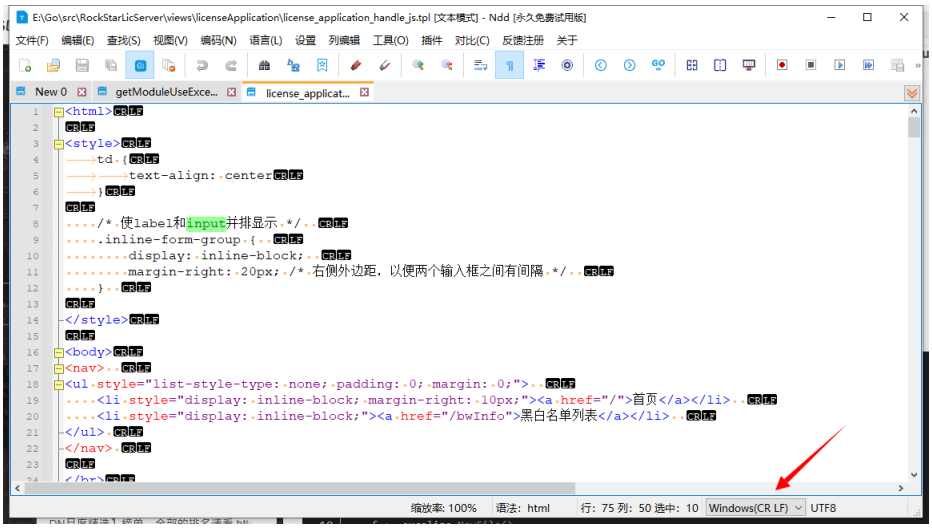


图 3.13: 状态栏的换行符类型

上图3.13中，提示当前编码类型是 windows 换行类型，即 **\r\n**。

另外一种显示空白字符的方法，点击工具栏的如下按钮，可以把空白字符显示出来，如下图3.14所示：

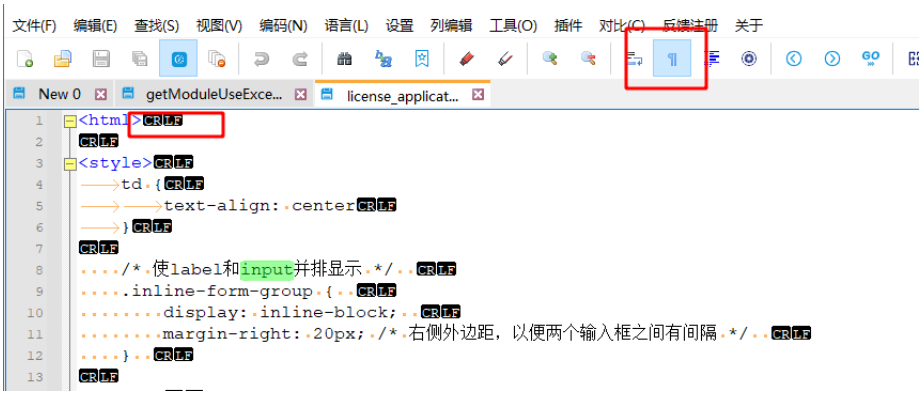


图 3.14: 工具栏显示空白字符

当你看到每行的尾部处，有个 CRLF 时，就表示换行符号类型是**\r\n**。

反之如下图3.15，如果只看到一个 LF, 则表示换行符号是**\n**。

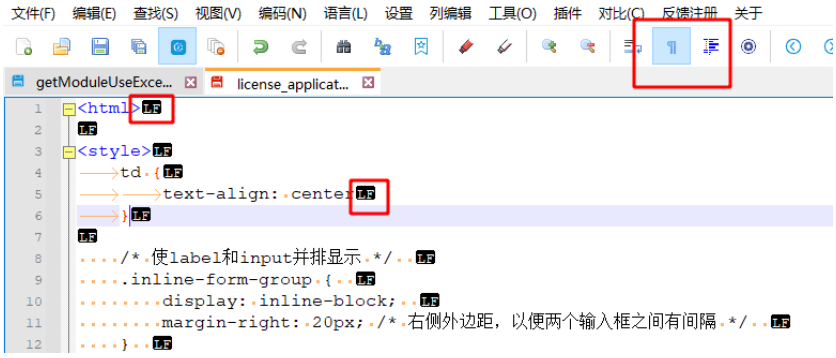


图 3.15: 换行符类型是\n

通过以上的介绍，确定当前文档的换行符号类型后，则在 Ndd 查找或替换的时候，**选择查找模式为正则模式或扩展模式**，即可正确查找对应的换行符号，一目了然。如下图3.16所示：

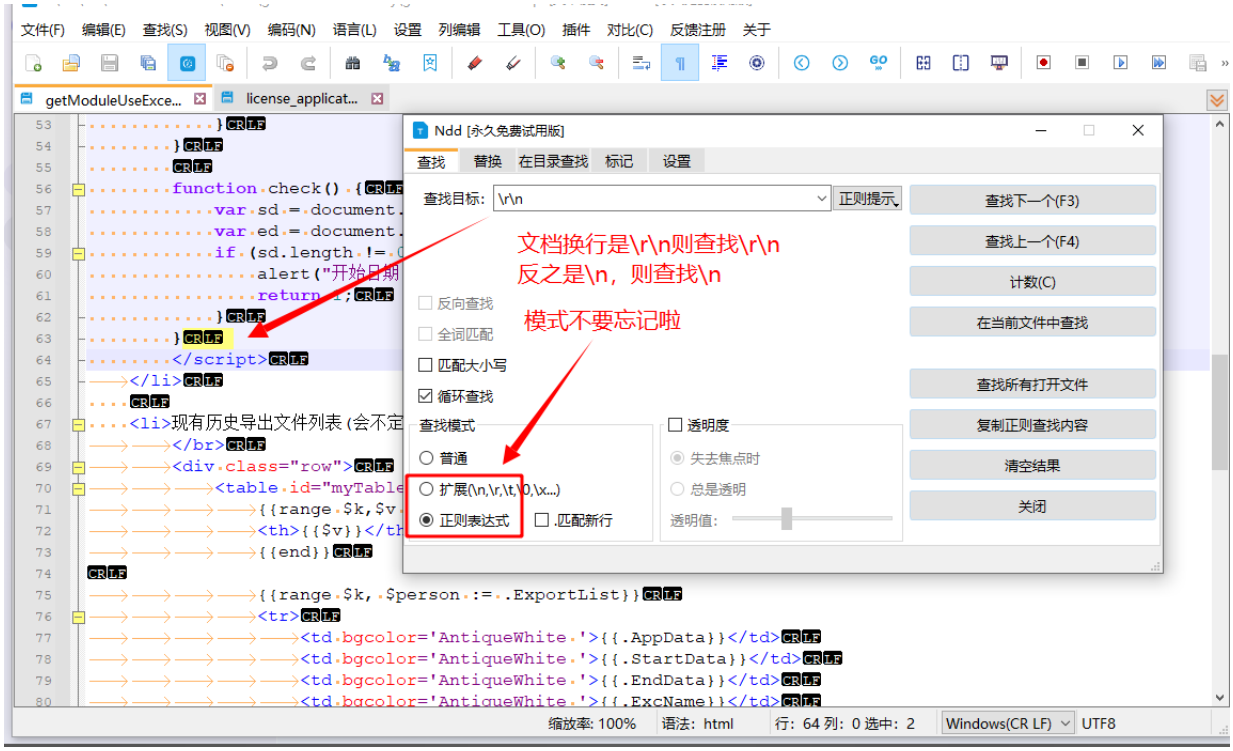


图 3.16: 选择正则或扩展模式查找对应换行符

3.4 文件编码、探测与转换

用户在 Ndd 中打开文件后，会自动探测该文件的编码。文件打开后，在软件界面的如下位置，会显示当前文件的字符编码，见下图 ①②标记处：

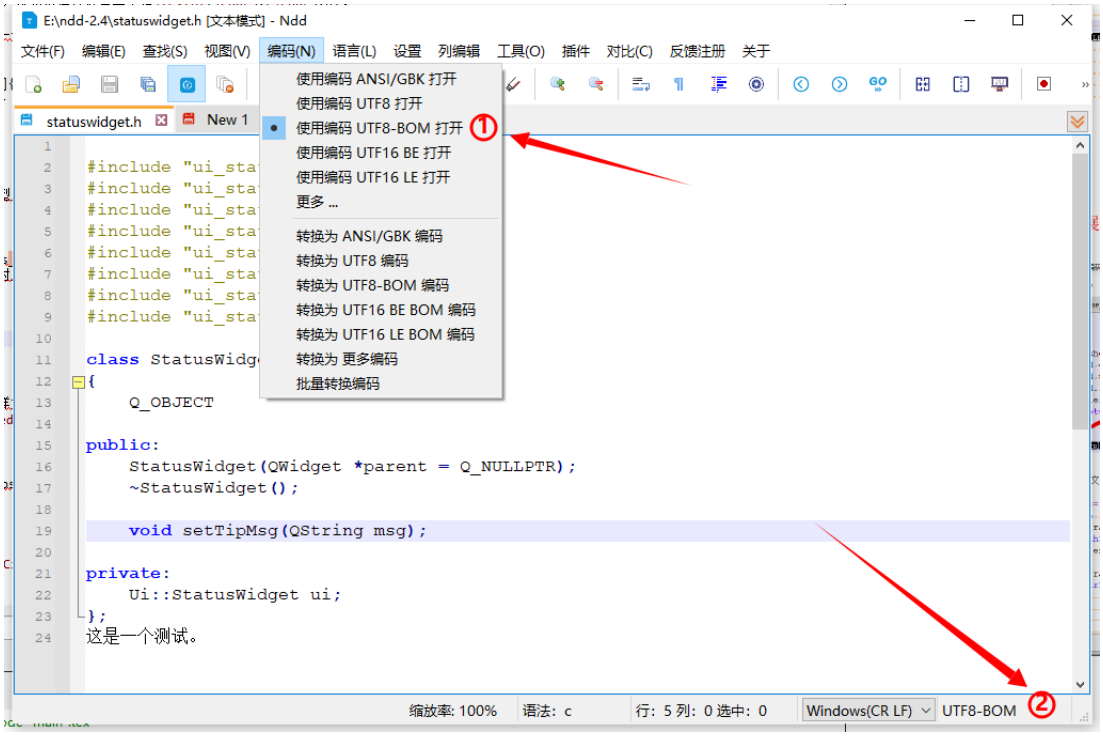


图 3.17: 打开文件后的编码显示

需要特别说明的是：文件编码可能有成百上千种，在打开速度和精度的权衡下，Ndd 尽快探测常见的文件编码。不过编码探测仍然可能会失败，此时打开文件，可能看到乱码字符。如下图所示：

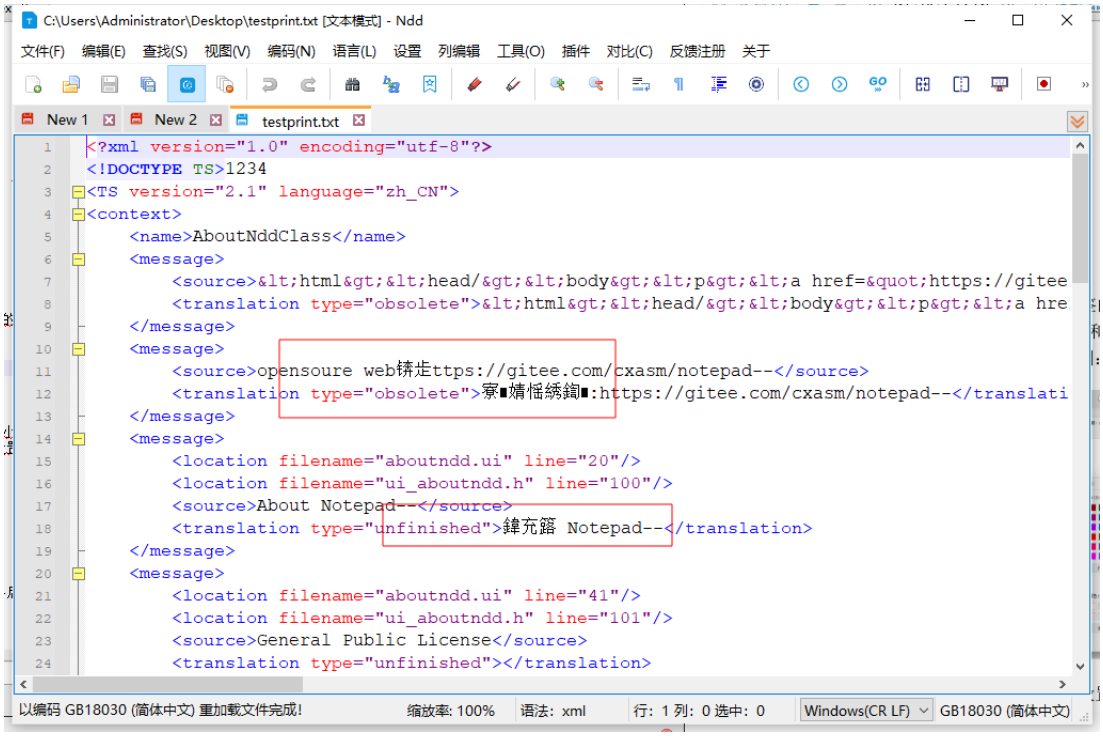


图 3.18: 打开文件乱码

出现该现象后，该如何处理呢？

3.4.1 文件乱码处理

打开文件后乱码，说明 Ndd 自动探测文件编码失败了，此时需要手动选择“打开文件时所用的编码”，在 **菜单-编码**菜单下面逐一选择如下编码，看看切换打开编码后，文件是否还有乱码，如下图所示：

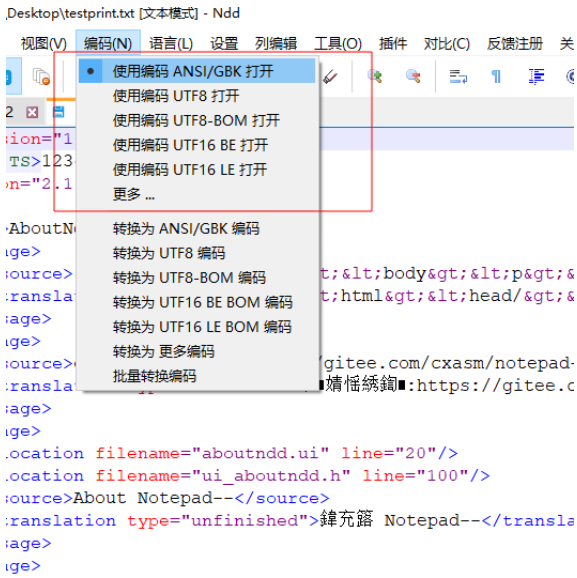


图 3.19: 手动选择打开文件时用的编码

上图的红色框框种，逐一尝试可能的编码，一旦选择后，Ndd 将重新使用所选编码打开文件，如果看到文档中乱码消失，则说明编码选择正确啦。上例子当我们尝试选择 uft8 编码后，文件乱码消失, 操作界面和效果如下：

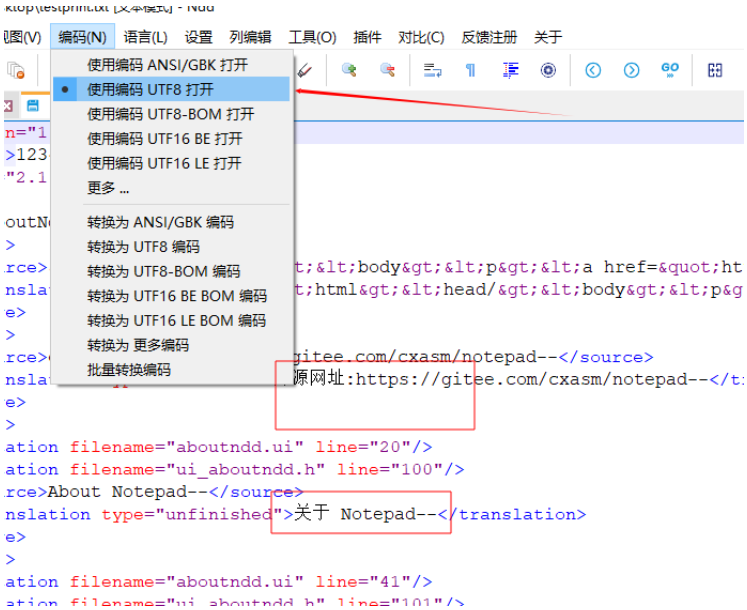


图 3.20: 手动选择 utf8 编码后乱码消失

上图中原本是乱码的字符，现在显示正常的中文。说明 Ndd 自动打开文件时，最开始探测文件是 gbk 编码，结果错误。现在手动告诉 Ndd 文件的原始编码是 utf8 后，打开正常。

一般在中国，常用的编码就是 GBK 和 Utf8 编码，这两种是最常见的。推荐大家使用 utf8 编码，而且给上 bom 头，这样无论如何下次打开文件都不会乱码啦，为什么会这样呢？

小知识：什么是 utf8 编码，什么是 bom 头呢？

简单来说，在文字编码协议没有被国际统一之前，每个国家都发明了自己特有的字符编码，比如中国有 GBK，美国有 ASCII，俄罗斯有斯拉夫字符编码……这就给文件编码的统一带来混乱和麻烦。一个中国的 gbk 文件，在中国的电脑上显示得好好的，如果发到俄罗斯的电脑上，就可能显示乱码了。因为俄罗斯的电脑可能默认使用斯拉夫字符编码打开咱们的 gbk 编码文件。

任何事情都有一个从乱到治的过程，于是 unicode 编码国际组织就出现了！该国际协议字符编码组织说，以后都不许乱来啦，大家都遵循使用统一的万国码——即 unicode。具体实现 unicode 编码的方案，有几种比较流行的，分别有 utf8,utf16,utf32 等，目前最常用的是 utf8。只要大家使用 unicode 编码，比如都使用 utf8，则全球的电脑都能识别该编码，统一使用 utf8 编码打开，就不会文件乱码。

不过还是有个编码探测的过程，即打开文件时，其实电脑也不知道该文件到底是不是 utf8 编码，仍然需要探测，所以探测是可能失败的，具体原因不详细多说。这个时候，bom 头就开始发挥它的巨大作用了。如果在文件的开头几个字节处，用特定的字节去明确说明本文档的编码，即所谓的文件 bom 头。那么不管哪国的电脑打开该文件，都读取识别文件头部的 bom 标记，就不用探测编码了，直接识别该 bom 头标记，就知道该使用何种编码打开文件。所以 utf8 有 utf8-bom，utf16-be 和 utf16-le 也有它们对应的 bom，不过最常用的还是 utf8-bom 头。不过可惜的是，咱们国家的 GBK 编码，没有 bom 识别头。

详细编码的知识，用户如果感兴趣，可自行学习相关详细资料。

3.4.2 文件编码转换

前文例子说，一个 utf8 编码的文件，被错误识别为 gbk 编码，导致打开错误。如果不做点什么，那么每次打开该 utf8 编码的文件，都会被错误识别为 gbk 编码。该如何避免此问题

呢？

答案是在第一次手动选择正确编码打开该文件后，接着立刻转换该文件编码为 utf8-bom 编码。这样就是明确告诉编辑器，下次打开文件直接按照 bom 所指明的 utf8 编码打开，不要探测编码。如此不仅省去探测过程，还保证不会编码错误，可谓一石二鸟。手动转换文档为 utf8-bom 编码，操作如下图：

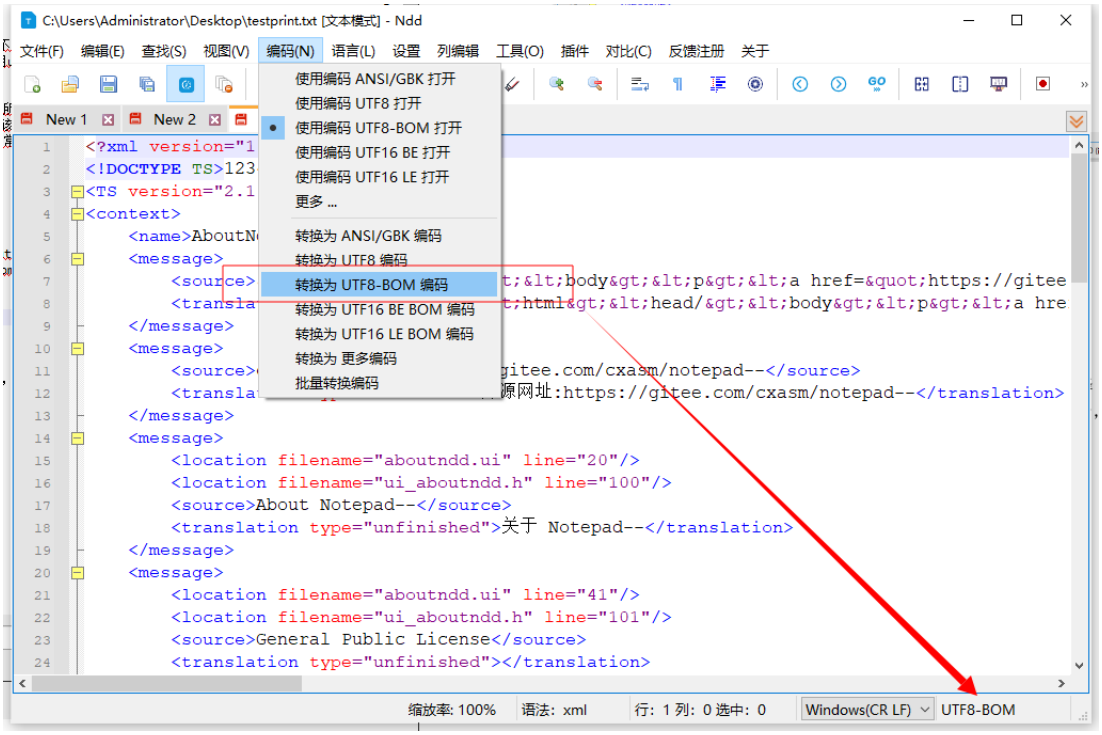


图 3.21: 手动转换文件的编码为 utf8-bom

选择 **编码-转换为 utf8-bom 编码**，如此操作后，会把当前编辑框文件转换为 utf8-bom 编码。Ndd 状态栏的右下角，也会显示目前该文件编码为 utf8-bom。注意要保存一下文件。那么下次打开该文件，则直接识别 bom 为 utf8 编码，直接打开文件，再也不会编码错误了。

这里务必一定要注意前置条件：“文件已经正确显示无乱码”，才能做接下来操作。即打开一个文件后，如果发现编码错误而文件乱码，是不能转换编码的，因为一开始编码就是错误的，在乱码的基础上转换编码，是错上加错!。必须先要按照 3.4.1文件乱码处理 节中指导，手动重新选择该文件的正确编码后，才能开始下一步转换编码。

3.4.3 为什么会编码错误，如何避免？

编码错误的原因有很多，一是编码本来就有成百上千种，有些小众编码因为历史原因虽然用的少，但是还有大量历史文档在使用，比如 windows12x0,IBMxxx 等系列的历史编码，探测不可能面面俱到。二是有些编码存在较大交叉的部分，比如汉字“未知”两字，其 gbk 编码和 utf8 编码都是合法的，如果一个文件种只有这两个汉字，其余全是英文字符，则压根不能精确判断该文档到底是 gbk 还是 utf8，存在混淆的可能。需要文档中有更多的汉字，才能进一步精确判断。

如何避免乱码呢？建议养成良好的习惯，非特殊场合下，个人有决定权的所有文件，都统一使用 utf8-bom 编码，这样就万无一失啦。

3.4.4 选择文件保存为 gbk 编码后，下次打开还是识别 utf8 编码

有用户反馈，明明手动保存了一个文件的编码为 GBK，下次使用 Ndd 打开该文件后，还是识别为 utf8 编码，怎么回事，是不是有 bug？

这种情况发生在文档中全是英文字符，没有任何中文，或者中文极少的情况下。为什么呢，纯英文的字符本来就是 acsii 字符，而 gbk 和 utf8 是 acsii 的超集，即 ascii 的字符，在 gbk 和 utf8 编码中是完全一样的。如果一个纯 ascii 的文件，即可以说是 ascii 编码，也可以说是 gbk 编码，还可以说是 utf8 编码，此时没法区分，说该文档是这三种编码中的任何一种都行，具体就看编辑器自己的实现。Ndd 默认来说优先识别为 utf8 编码。

如果要 Ndd 优先使用 gbk 编码，而非 utf8 编码，则可在 [**设置-字符编码类型**] 中，把下面三个地方，都修改为 GBK 即可，如下图：

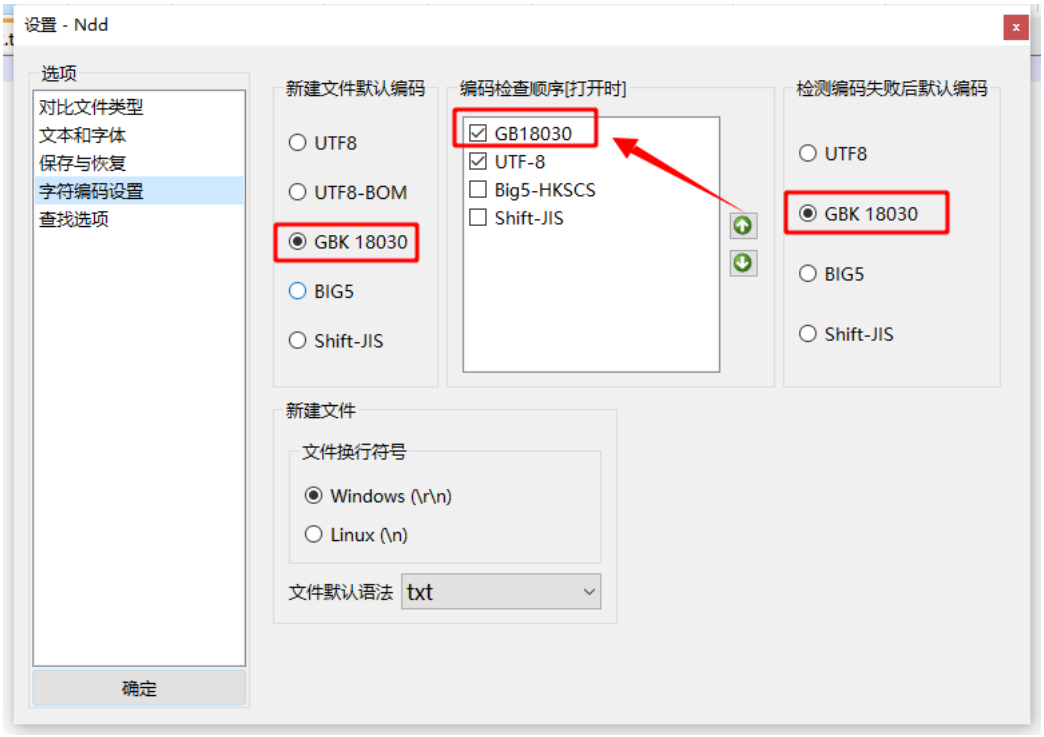


图 3.22: 设置 Ndd 默认优先使用 gbk 编码

有用户说，我不要 GBK18030 编码，能不能要 GBK2312 编码？答案是没有必要，因为 GB2312 是 GBK18030 的子集，GBK18030 支持 7 万多个中文字符，而 GB2312 只能支持 6763 个汉字，GB2312 包含的汉字 GBK18030 全有。故实在是没有必要多此一举退回到 GBK2312，这压根不算一个问题。

3.4.5 批量文件编码转换

在 [**菜单-编码-批量转换编码**] 位置，界面如下：



图 3.23: 批量转换编码

建议对处理文件类型中，过滤选择一下特定的文件类型，这样缩小处理范围后，不仅速度更快，还可以避免一些非文本类型的后缀文件。**这个界面只能处理不超过 10M 大小的文件，超过该大小的文件，请使用 5.2 大文件修改编码 。**

3.5 行号与书签风格

在主体全局里面，分别设置行号和书签的样式，比如背景色，字体大小等。比如之前说的，设置编辑框背景色后，希望左边的行号和书签的颜色，也和背景色保存一致，依次点击下图①②③④后，设置行号的背景色如下图：

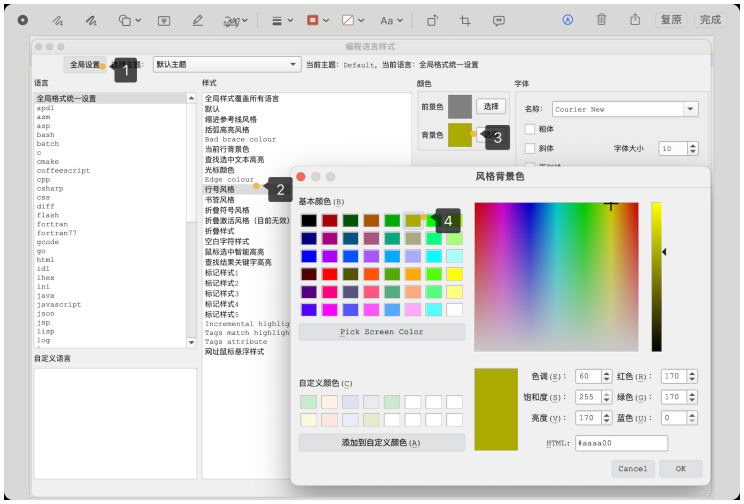


图 3.24: 设置行号的背景色

再点击下图①②③④后，类似修改书签的背景色如下图：



图 3.25: 设置书签的背景色

最终左边的空白背景色效果如下：

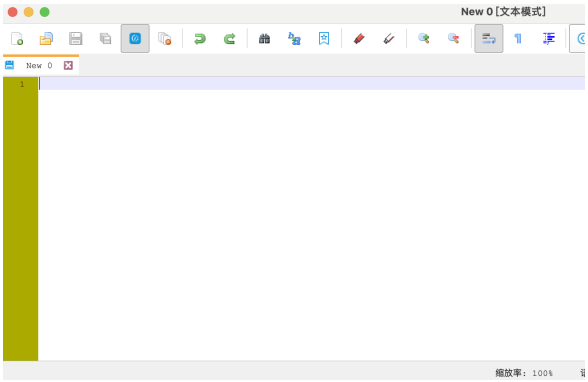


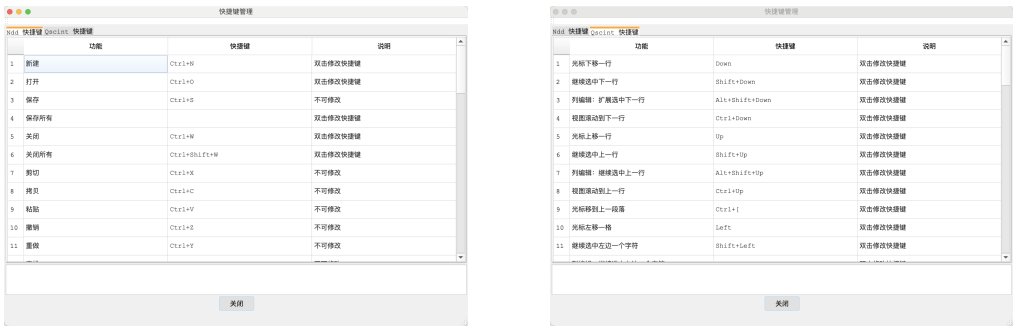
图 3.26: 设置编辑框左边空白背景色

如果不是 txt 语法，则其它编程语言的编辑框左边还存在折叠栏，也是类似的设置，修改折叠栏的背景颜色即可。限于篇幅有限，暂时不能对所有样式进行举例，感兴趣的用户可自行实践摸索。

有用户说，行号看起来字体太大，留白太宽了，能不能变小？ 只需要在全局里面，把行号样式的字体大小调小即可缩短留白宽度。书签可以在视图中进行隐藏。

3.6 快捷键设置

Ndd 自带一些快捷键，点击菜单 **设置-快捷键设置**，看到如下快捷键窗口：



(a) Ndd 快捷键 (b) scintilla 内置快捷键

图 3.27: 软件内置快捷键

在 windows 系统下，为了不拖慢软件的启动速度，内置 scintilla 快捷键不能修改；maocs 和 linux 系统的内置 scintilla 快捷键可修改。

笔者对快捷键的理念是：快捷键一定要短，最好只需要 2 个手指头就能操作，这样才快捷！凡是需要 3 个手指头的快捷键，已经不如鼠标点击来的方便。基于此，对于常用的功能，尽可能使用两个键的快捷键，而且是要在双手最多各出一个手指的情况下，需要方便完成组合击键的才好使。因此，大于 2 个键的快捷键，有些奇怪组合不便于操作的快捷键，笔者从 Ndd 中进行了删除。

3.7 常见快捷键举例

这里暂时以 windows 系统下的键盘举例。macos 不好统一说明，因为 macos 现在有 macos mini 电脑，macos 笔记本电脑，macos 一体机，一些人可能会为 mac 设备外接一个 windows 的键盘，情况会混乱一些。

1. 复制（**Ctrl+C**）、粘贴（**Ctrl+V**）、剪切（**Ctrl+X**）
2. 文本转换为大写（**Ctrl+Shift+U**）文本转换为小写（**Ctrl+U**）
3. 语法折叠（**Alt+1**) -> （**Alt+8**) ；展开（**Alt+Shift+1**) -> （**Alt+Shift+8**)

   其中 1-8 分别对应语法下的 8 种折叠层次。
4. 进入列编辑模式

   windows 和 macos 下是 Alt+ 鼠标左键在编辑框中框选; linux 下是 Ctrl+ 鼠标左键框选
5. 通过键盘进入列编辑框选范围

   同时按下 **Alt+Shift** 键，再通过键盘的上下左右键在编辑框中进行框选。需要退出列模式时，只需要鼠标在编辑框其它地方单击即可。
6. 列编辑模式下，光标到行头或行尾

这种方式有时候特别有用，比如希望在多行的前面或尾巴加上输入一些内容。先通过5或 6 的方式进入列编辑，windows 下通过 **Home** 或 **End** 键，分别让光标进入多行的头或尾。macos 下通过快捷键 **Ctrl+H** 到达行头 **Ctrl+E** 键到达行尾。（备注：Ndd 2.18 该功能暂时失效，在 2.17 2.19 3.0 之后等版本中是可行的。）

以上快捷键是用户反馈比较集中的几个。还有一个问题，有些用户说希望插件也能支持快捷键，目前暂时没有这样做。一是因为 2 个键的快捷键实在有限，插件往往只能使用 3 个键的组合，实在不快捷；二是插件可能比较多，而且是第三方开发者开发，如果随意支持快捷键，则很可能导致冲突问题。所以目前暂时不对插件安排过多的快捷键支持，可通过鼠标点击来完成操作。

3.8 记录上次关闭时的打开文件

不同用户可能有不同的叫法，比如“关闭软件后，下次打开要记住上次关闭时的文件”，“下次打开要恢复上次关闭时的文件”，诸如此类的说法。目前该配置选项默认是开启的，如果需要手动关闭或开启该功能，可在 **设置-保存与恢复**选项中勾选或取消勾选配置，如下图所示：

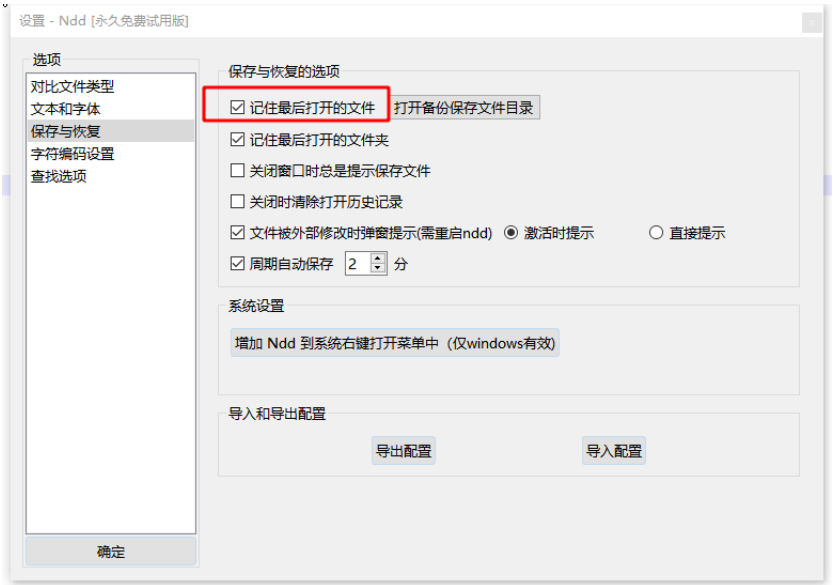


图 3.28: 记住最后打开的文件

一旦开启该选项，则关闭软件时，不会提示是否保存，没有保存的文件会自动写入到缓存文件中。下次打开时，会把上次关闭时的打开文件恢复出来，同时恢复光标上次所在位置。**备注：为了不拖慢打开速度，对于超过 5M 大小的文本文件，还有二进制文件，这两类文件不会进行自动恢复。**

3.9 记录上次关闭时打开的文件夹

该功能与记录上次关闭时的打开文件 类似，不同的是恢复的是文件夹。有些用户习惯在 Ndd 中打开文件夹，其显示效果如下所示：

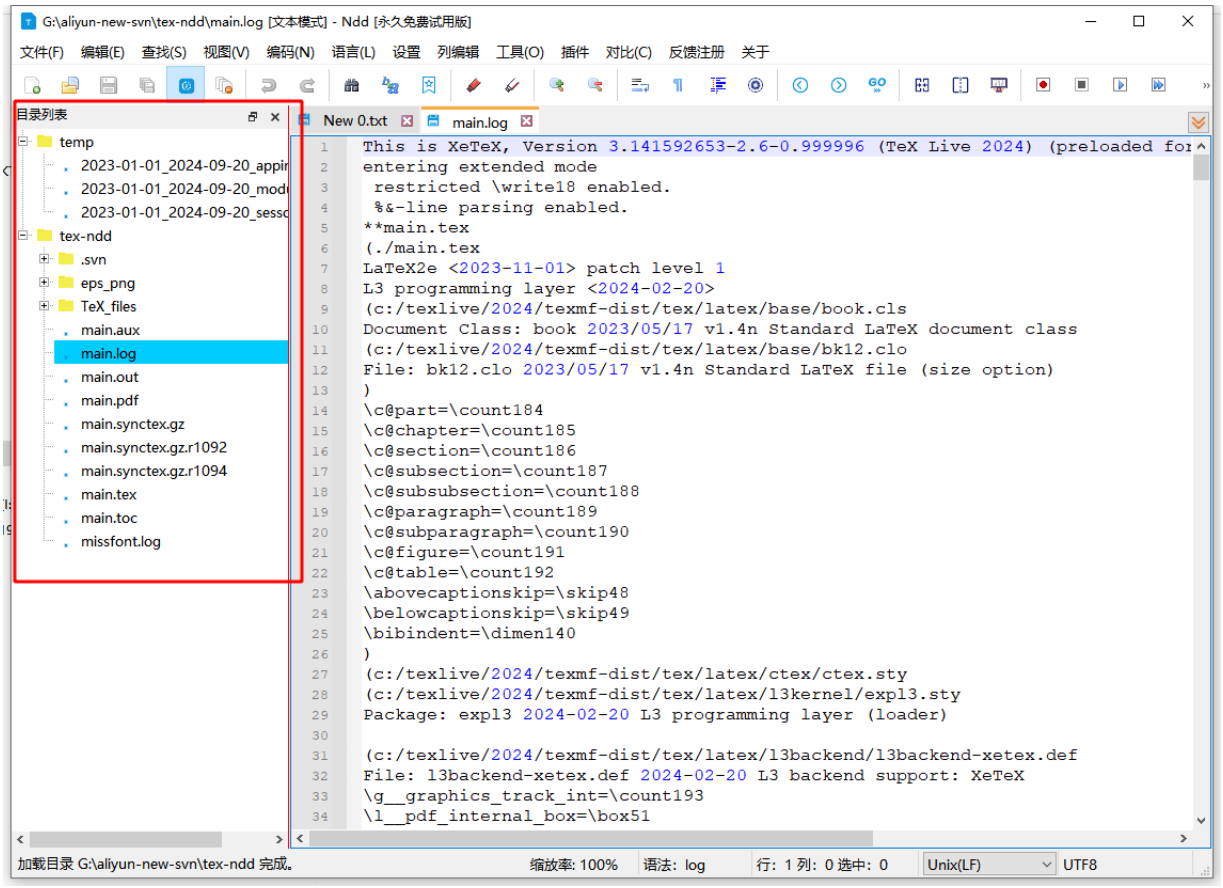


图 3.29: 左侧目录列表显示

有些用户说，把文件夹拖入到 Ndd 中后，关闭软件后下次打开，上次的文件夹没有恢复出来，需要在设置中勾选如下选项即可：

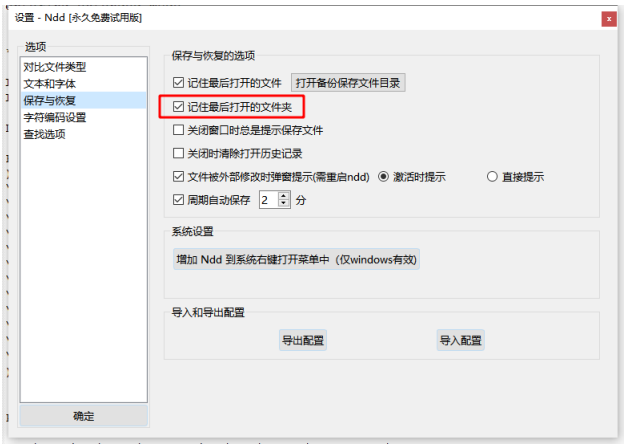


图 3.30: 记住最后打开的文件夹

小技巧：如何快速定位某个编辑框文件在左侧文件夹中的位置？如下图操作即可。

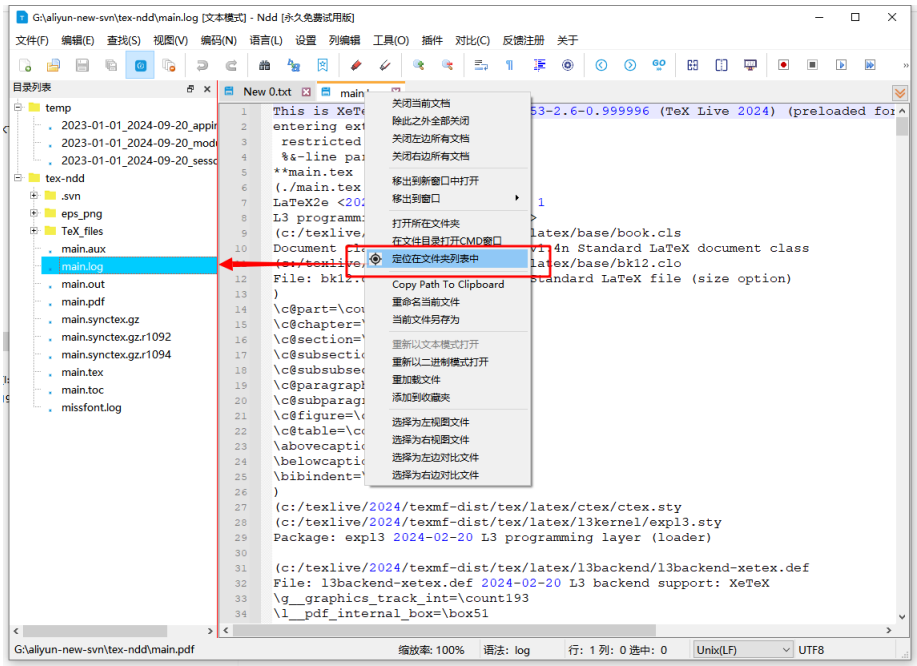


图 3.31: 文件夹视图中快速定位文件

3.10 崩溃后恢复文档和定时保存

因为有了功能 3.8记录上次关闭时的打开文件, 有时可能会遇到一个问题：如果软件不小心崩溃掉，下次打开后上次没有保存的文件还能恢复吗？Ndd 默认会开启软件自动保存，新建的临时文件每隔 3 分钟，会自动保存到一个缓存目录，相关设置和路径见下图：

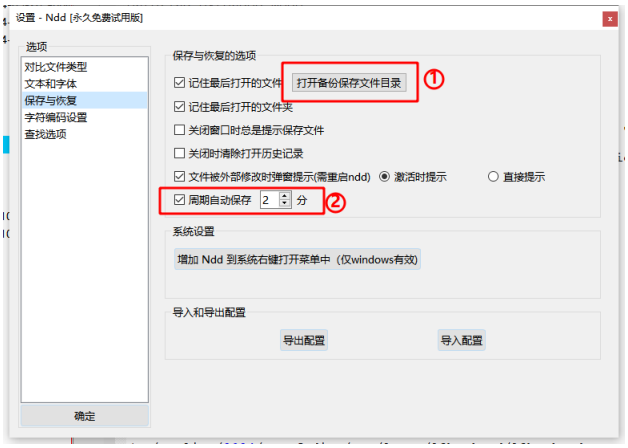


图 3.32: 缓存路径和定时保存

自动保存的开关和时间设置，在上图中的标记②处，最小保存时间为 2 分钟。

临时文件的相关缓存保存路径，在上图中的标记①处的按钮，可以点击该按钮，弹出对应的目录，见下图所示：

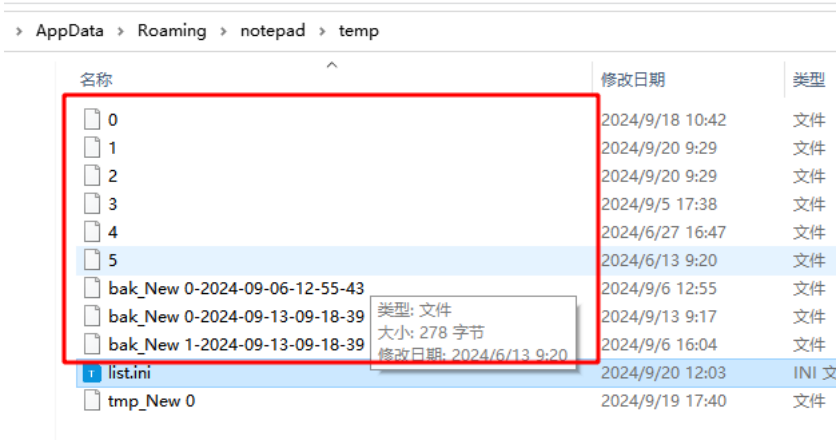


图 3.33: 崩溃后下次打开的提示

上图中的红框中的文件，就是上次崩溃时打开的临时文件的缓存。如果你需要找回临时文件中的内容，可以逐一打开这些文件，检测并找回其中的内容。

温馨提示：新建临时文件，建议不要存放重要的信息，比较重要的文本信息，还请保存为非临时文件。非临时文件，在写入之前，会在当前文件先写入一个缓存 swap 文件，如果此时崩溃，则缓存 swap 文件会记录崩溃前的信息，可以比较安全的找回崩溃之前的文本信息。

Ndd 崩溃后，下次打开软件，会弹出提示询问用户，是否去找回文件。见下图所示：

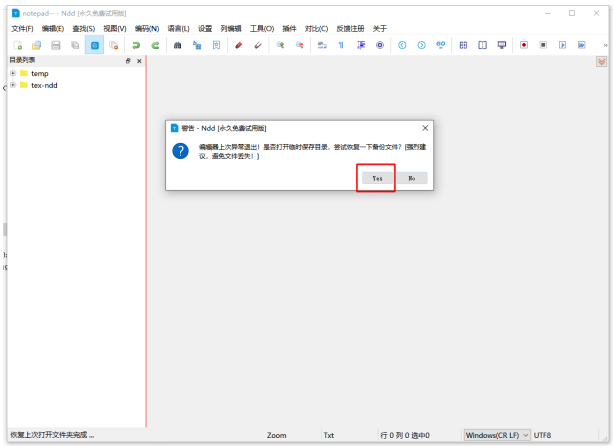


图 3.34: 缓存路径中的临时文件

此处有些用户会比较奇怪，说我点击了 Yes 后，压根就没有恢复出来。其实点击 Yes 后，效果和图片 3.33 的效果是一样的，只是弹出缓存文件的路径，具体文件的内容，需要自行手动到该目录下面去寻找。还是之前的建议，既然是临时文件，则不存放重要信息，养成良好的习惯，可以避免许多非必要的技术性失误。

3.11 主题配色的导入与导出

有些高级用户，对主题的配色和语法设置比较仔细，会详细的定制自己喜欢的配色。好不容易设定好的配置信息，如果换电脑后，或者重装软件后，配色直接丢失，又需要重新配置，实在是一件麻烦的事情。Ndd 提供主题配色的导入与导出操作。一旦用户设定好自己喜欢的主题语法配色后，可把配置导出到其它地方。下次重装软件后，只需要把之前的配置导入，即可恢复之前的配色方案。操作设置界面如下：

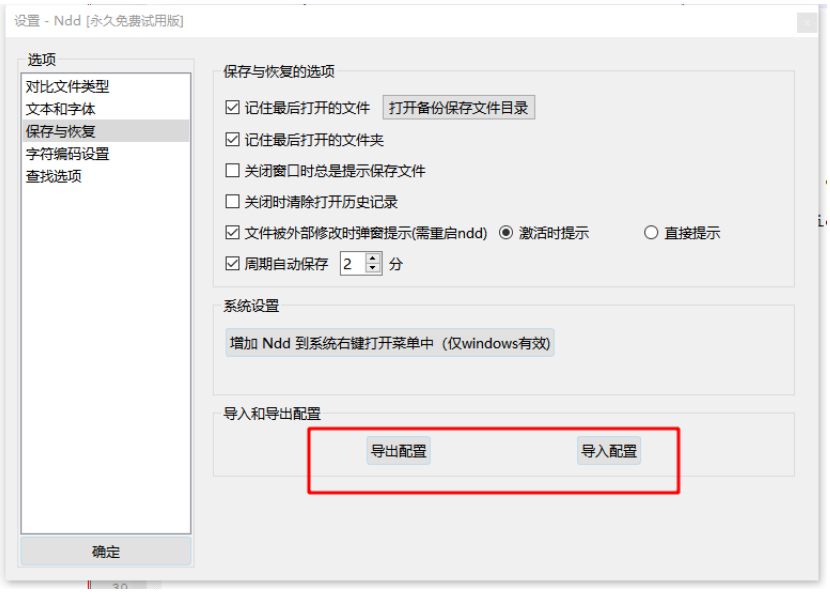


图 3.35: 配置导入导出

3.12 列编辑

Ndd 有列编辑模式，在 windows/macos 下面，使用 Alt+ 鼠标左键，即可在编辑框中进入列编辑模式。

linux 下面使用 Alt+ 鼠标无法进入列编辑模式。那是因为 linux 下面的 Alt 键，系统有特殊的操作含义，为了不与 Linux 系统对 Alt 键的冲突，Linux 下面需要使用 Crlt+ 鼠标左键进入列编辑模式。

具体的详细快捷键方式，还请参考 常见快捷键举例 中列编辑快捷键的相关内容。

除此以外，在菜单**列编辑-列块编辑 (快捷键 Alt+X)**中，还有进行列插入的功能，界面如下：

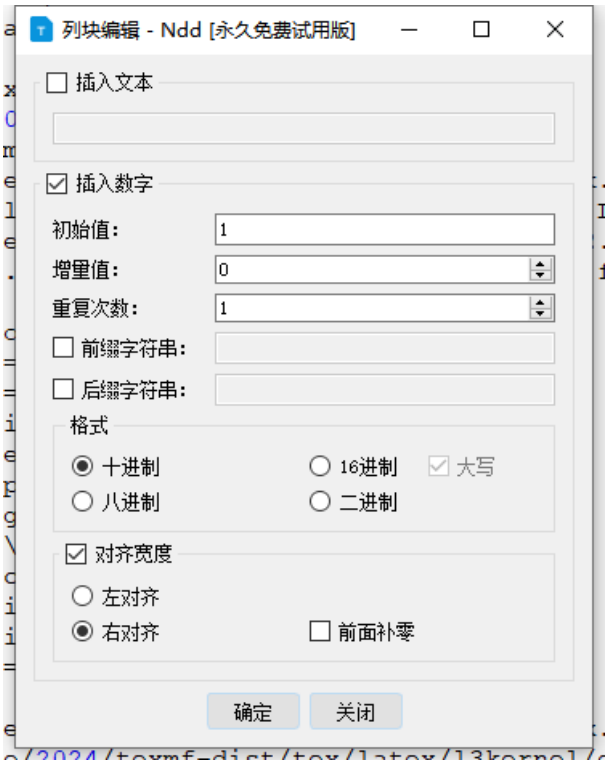


图 3.36: 列块编辑

该功能既可以从光标开始往列方向插入内容，也可以先选中一块列范围后，在该范围中

进行插入内容。

3.12.1 用例：使用列编辑功能批量生成维护命令

应用场景

在日常工作中，维护人员可能会要批量生成一些配置命令，示例如下:

```
添加12个分组

ADD GROUP:GID=1,FCN=646322;
ADD GROUP:GID=1,FCN=646322;
ADD GROUP:GID=1,FCN=646322;
ADD GROUP:GID=1,FCN=646322;
ADD GROUP:GID=1,FCN=646322;
ADD GROUP:GID=1,FCN=646322;
ADD GROUP:GID=1,FCN=646322;
ADD GROUP:GID=1,FCN=646322;
ADD GROUP:GID=1,FCN=646322;
ADD GROUP:GID=1,FCN=646322;
ADD GROUP:GID=1,FCN=646322;
ADD GROUP:GID=1,FCN=646322;
```

原始内容

但这里文中的“GID”要按顺序递增编号，否则除第 1 条命令外，其余命令都执行失败了。常规的作法是从 1 写到 12，在这里工作量貌似也不大，但如果命令再多一点，工作量就会比较可观，并且手工编辑容易引入错误。

操作步骤

1. 步骤 1: 按住“Alt”键，并用鼠标选中要修改的编号“1”这一列，如下图所示：

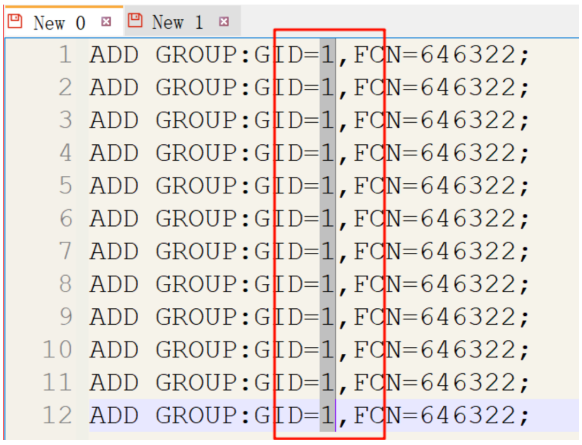


图 3.37: 示例

2. 步骤 2：选择**列编辑-列块编辑**

在弹出的窗口中选择“插入数字”，并将“增量值”设置为 1，确保数字按 1 递增，单击“确定”按钮，即可生成数字序列

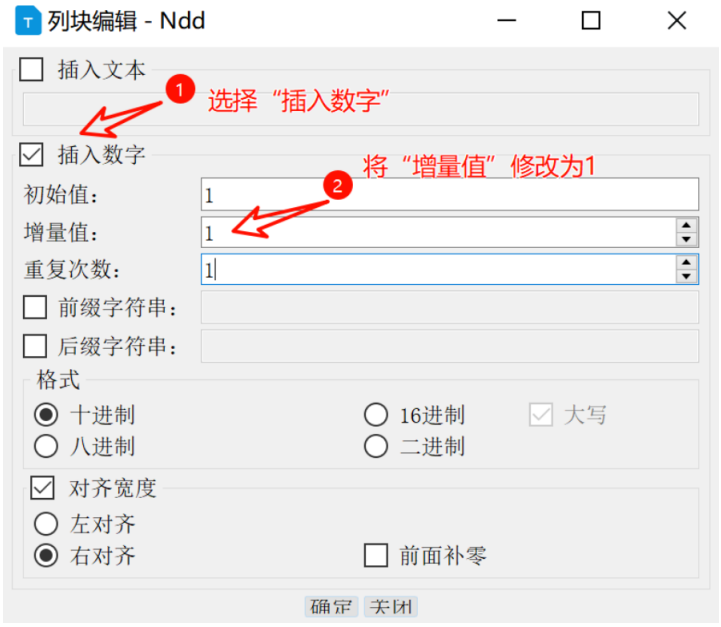


图 3.38: 列方式递增插入数字

说明:

- “重复次数”一般情况下设置为 1，得到的序列将是 1、2、3、4、5、6....
- “重复次数”设置为 2，则得到的序列是 1、1、2、2、3、3...
- 其他情况依次类推，可以根据实际需要进行设置。

插入序列后的文本样例:

```
ADD GROUP:GID= 1,FCN=646322;
ADD GROUP:GID= 2,FCN=646322;
ADD GROUP:GID= 3,FCN=646322;
ADD GROUP:GID= 4,FCN=646322;
ADD GROUP:GID= 5,FCN=646322;
ADD GROUP:GID= 6,FCN=646322;
ADD GROUP:GID= 7,FCN=646322;
ADD GROUP:GID= 8,FCN=646322;
ADD GROUP:GID= 9,FCN=646322;
ADD GROUP:GID=10,FCN=646322;
ADD GROUP:GID=11,FCN=646322;
ADD GROUP:GID=12,FCN=646322;
```

修改后

3.13 宏的录制与运行

Ndd 支持宏功能，支持宏的录制和回放运行。为了保持 Ndd 的轻快，没有重复的把工具栏的一些功能，放置到菜单栏中。Ndd 宏功能在工具栏上面，一些用户可能会误以为 Ndd 不支持宏。宏的功能区如下图所示：

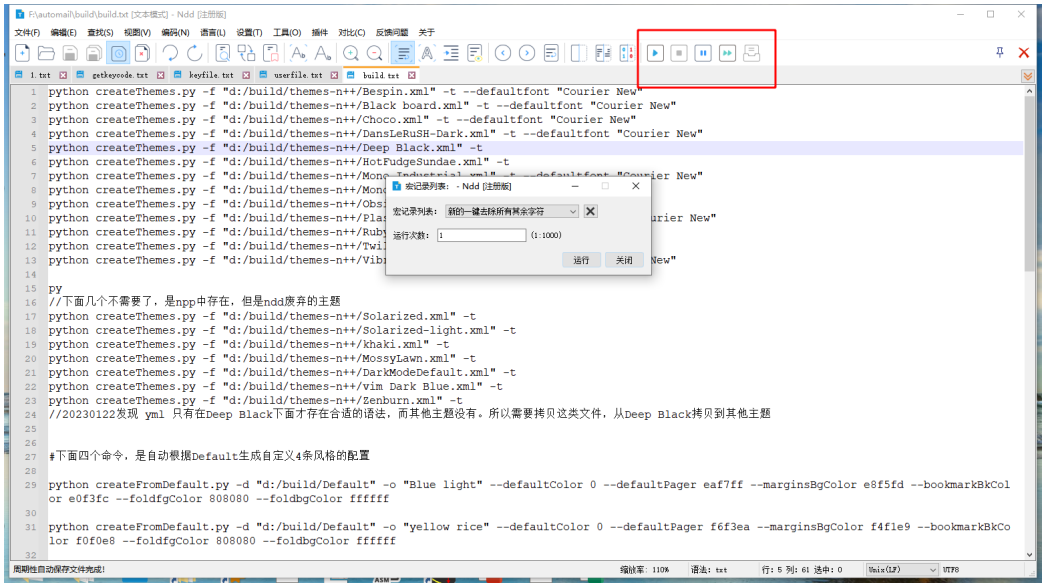


图 3.39: 宏的录制运行

宏可以用来做什么呢？

这里给一些小白用户普及一下相关知识和用法。笔者录制了一个宏，功能是**一键去除中文和空白**。如下图：

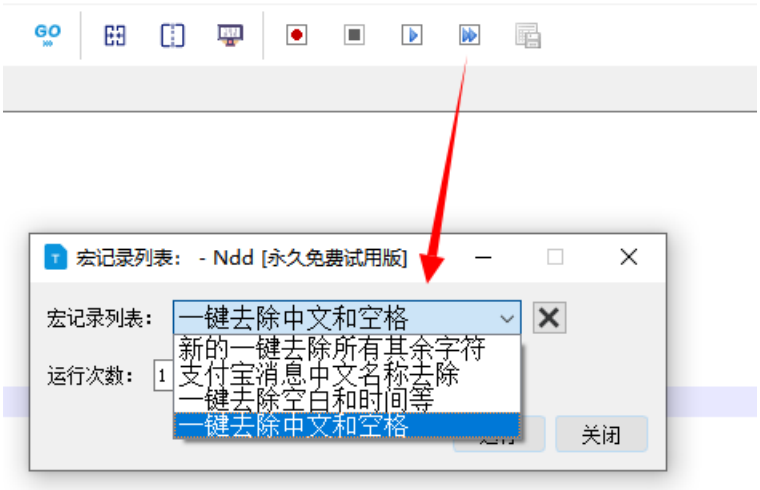


图 3.40: 宏的录制运行

在没有录制宏时，如果用户要做这个事情，则至少要做两次如下操作：

1. **在查找替换中，使用替换把中文字符替换为空**
2. **在查找替换中，使用替换功能把空白字符（制表符和空格）替换为空**

那么以后每次用户要做该功能时，都要如此操作两次，实在麻烦！能否把这两个操作，保存为一个操作，然后下次要用的时候，直接调出来执行，就可以起到同样的效果？提高生产率！不错，宏就可用来做这活。

简单来说，宏可以把多个操作记录起来，形成一个组合命令，下次可直接调用。 用户只需要有这个概念就够用啦。宏特别适合做一些查找和替换的操作。虽然宏很有用，能节约不少繁杂的多次操作过程，但宏也有它的一些局限。如果您需要处理的多个操作组合，与查找和替换相关，则特别适合使用宏。反之，有些操作可能不适合宏，或者说宏没有支持到那么细的粒度。

3.13.1 宏录制与回放

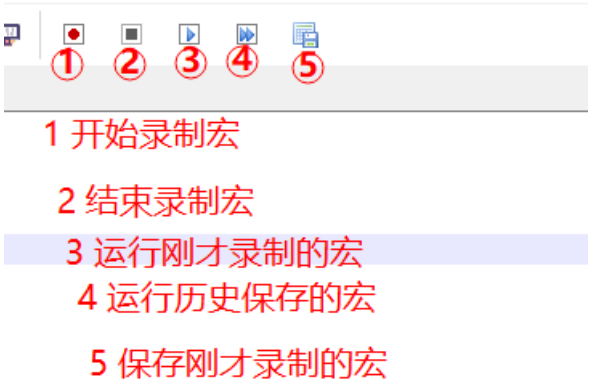


图 3.41: 宏录制与回放

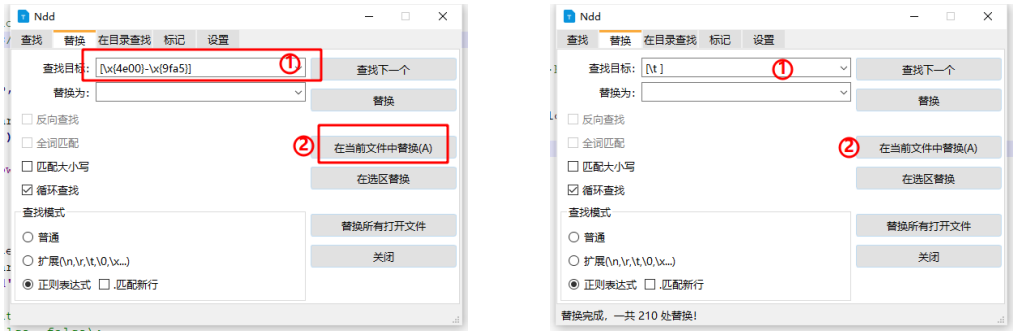
上图中的①②③④⑤分别讲解如下：

1. **开始录制**

   要开始录制一段宏，先要点击“开始录制”工具栏按钮。此时在编辑框做的查找或替换等操作，便会被记录下来。

   比如上面的例子“一键去除中文和空白”，录制过程如下：① 点击开始录制按钮。② 先打开查找框，在替换中输入表示所有中文的正则规则，[\x{4e00}-\x{9fa5}]，点击“在当前文件中替换”按钮，此时当前文档中的中文被去除。③ 接着再输入正则表达式 [\t] (备注：\t 后面有个空格字符)，点击“在当前文件中替换”按钮，此时文档中的制表符和空格两类空白字符，也被去掉。

   如此两步操作，即达到了“一键去除中文和空白”的效果。操作如下图所示：



(a) 去除中文 (b) 去除空格和制表符

图 3.42: 录制宏：一键去除中文和空白

2. **结束录制**

   录制完成后，点击结束录制，如此后续的操作便不会继续被录制。

3. **运行刚才录制的宏**

   用户此时点击该按钮，即可运行刚才录制的宏。起到的效果自然是“一键去除当前文档中的中文和空白”。如果后续还需要调用该宏，需要点击按钮⑤，把此宏保存到 Ndd 中，方便后续调用。

4. **运行历史录制的宏**

   用户可以先选择3.41 宏录制与回放 中的⑤，把刚刚录制的宏保存起来。一旦宏被保存，后续点击该按钮，则可以调出过去录制的宏，选择其中一个，进行再次调用。

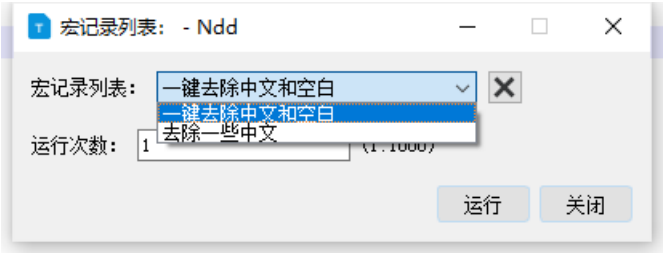


图 3.43: 运行历史录制的宏

5. 保存之前录制的宏

这个按钮应该先讲，就是刚刚录制的宏“一键去除中文和空白”，如果不保存，则只是在内存中。一旦 Ndd 关闭后，或者重新录制新的，之前录制的宏将丢失。如果后续还需要运行刚录制的宏，则需要保存该宏。此后在“运行历史录制的宏”界面中，则可以再次调用该宏。保存宏的逻辑也非常简单，就是在输入框中给刚刚录制的宏取一个名称，然后点击“保存宏”按钮即可。

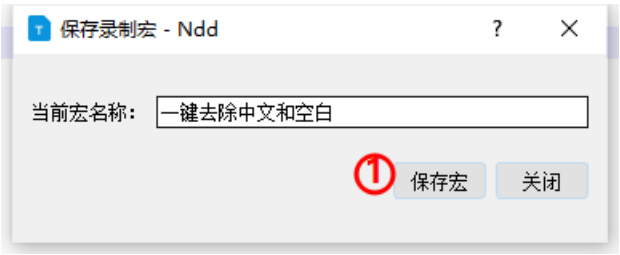


图 3.44: 保存刚才录制的宏

如果宏录制后，无法保存，或无法运行，或者运行后结果和预期不一样，这说明在宏的录制过程中，有些操作是不支持录制的。总的说来，宏的作用很大，但并不是万能。

说明：有些用户说能不能开放宏的格式，或者把宏保存在外面，让用户自行编写？

Ndd 会把宏的操作码保存在内部的文件中，该文件的确存在。记录的操作码，阅读起来比较困难，一般人怕是没有精力去研究这种宏指令，还是通过自动录制的方式，将用户的操作转换为宏操作码更加简单。

Ndd 的理念是，尽量提供简单的方法，而非提供复杂的方法，尽量为广大非专业编程人

员提供大众工具，而不是为了体现所谓的技术，而搞出一种很少有人会用的“屠龙之术”。后续还有类似的一些设计或操作，都会体现此理念。

3.14 文件和语法高亮关联

这里的关联，不是说鼠标双击打开一个文件时，自动使用 Ndd 打开（见 双击打开文件时使用 Ndd 打开）；而是说打开一个编程语言的文件时，让其自动选择该语言对应的语法高亮。这是两个不同的概念。

举例来说，打开一个.c 的文件，如何让 Ndd 使用 c 语言的语法高亮？或者使用 ndd 打开一个文件后缀为.c1 .c2 的文件，如何让 Ndd 也使用 c 语言的语法高亮？

如下所示，在**设置-语法文件后缀关联**界面中，对语法文件进行后缀关联。

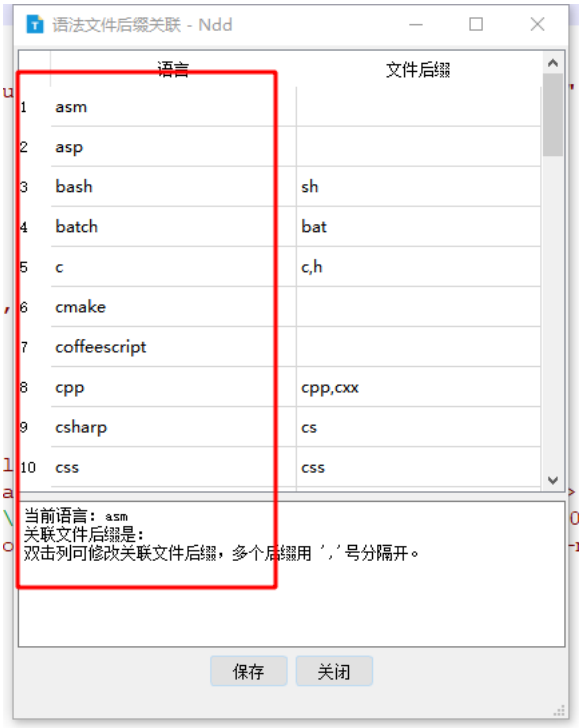


图 3.45: 文件和语法关联

左边是支持的语言，右边是语言对应的文件后缀。用 c 语言举例来说，默认会把.h .c 的文件后缀直接使用 C 语言进行语法高亮。如果您有一个文件，它的后缀是.test，但是也需要使用 C 语言进行语法高亮，则只需把 test 追加在上图的 c,h 后面即可。注意多个后缀之间，使用英文状态下的逗号进行分割。

这样设置后，下次使用 Ndd 打开后缀为.test 的文件时，Ndd 会自动选择 c 语言的语法进行高亮。以此类推。

3.14.1 双击打开文件时使用 Ndd 打开

方法一：在 3.6.0 版本之后，使用管理员权限运行Ndd，可以通过如下方式关联文件双击打开时，使用 Ndd 打开。

依次点击菜单 **设置、打开文件关联 Ndd**，如下图所示：

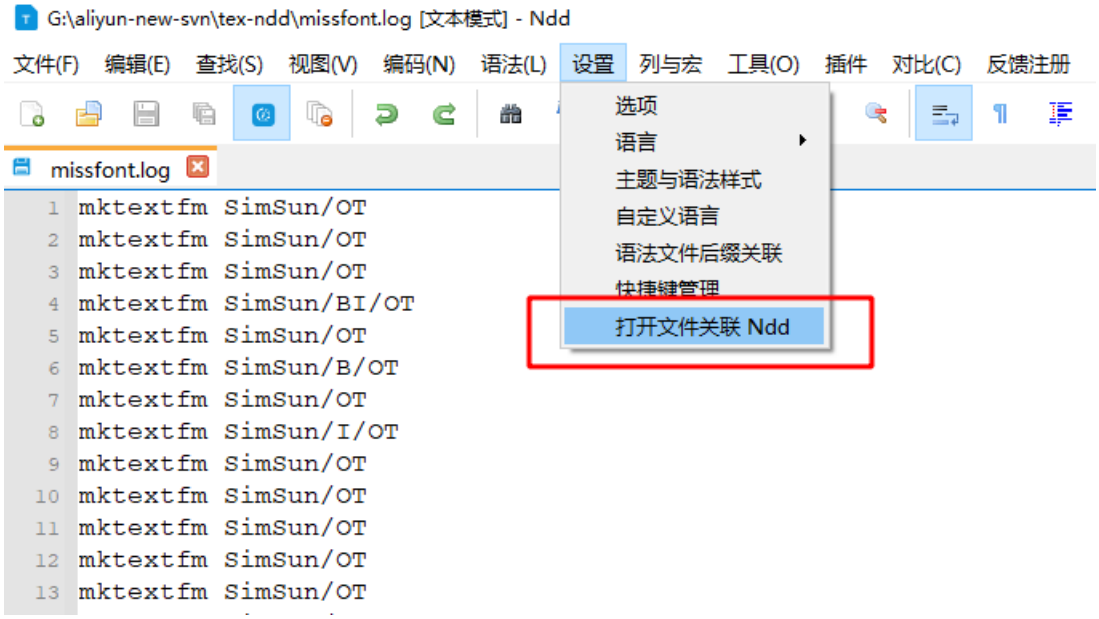


图 3.46: windows 下双击文件使用 Ndd 打开设置方法

在弹出的子窗口中，点击新增，把需要使用 Ndd 打开的文件后缀填入输入框中，可一次性填入多个，以英文冒号分割。比如希望文件后缀为 cpp h txt 的三类文件，双击时使用 Ndd

打开，则填入内容如下：

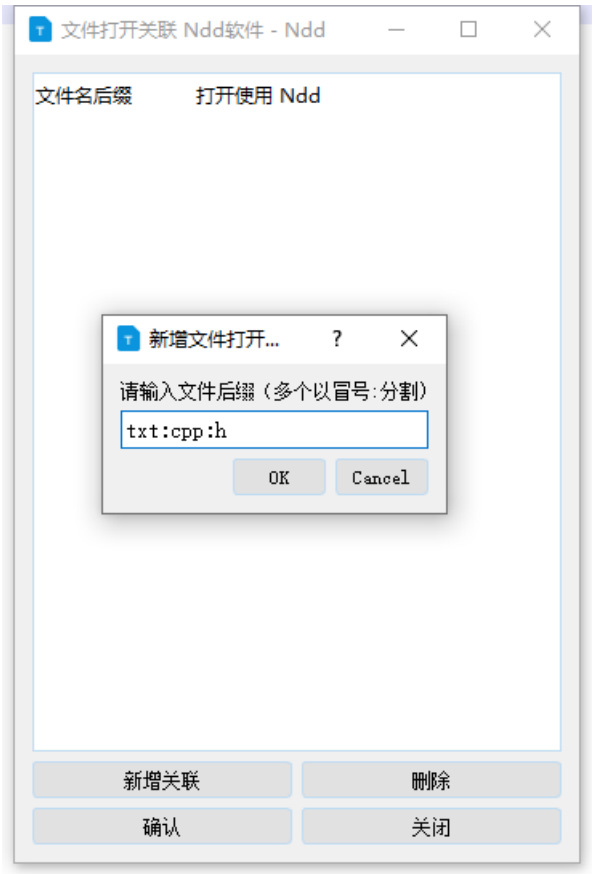


图 3.47: 举例关联打开 txt cpp h 的后缀文件

再添加确认，关闭按钮，即可，如下图所示：



图 3.48: 举例关联打开 txt cpp h 的后缀文件

之后在 windows 系统下，双击文件打开 txt h cpp 等后缀的类型文件，会自动使用 Ndd 打开。

说明：① 有些后缀的文件可能有多个关联程序，此时双击文件可能会有选择选项，第一次可能需要选择一次。② 使用 Ndd 关联打开后，文件的图标会在 windows 下变化，但不一定是实时变化，可能下次电脑重启后会变化。② 该方法只对 windows 系统有效，而且必须是以管理员权限运行 Ndd，非管理员权限可能会失败。

方法二：windows 系统提供的方法

这里只提供 windows 下面的方法，双击一个文本文件，如何自动使用 Ndd 打开呢？这个设置需要在系统中设置。鼠标在该文件上面右键，选择属性，然后弹出如下界面，在打开方

式中选择更改：

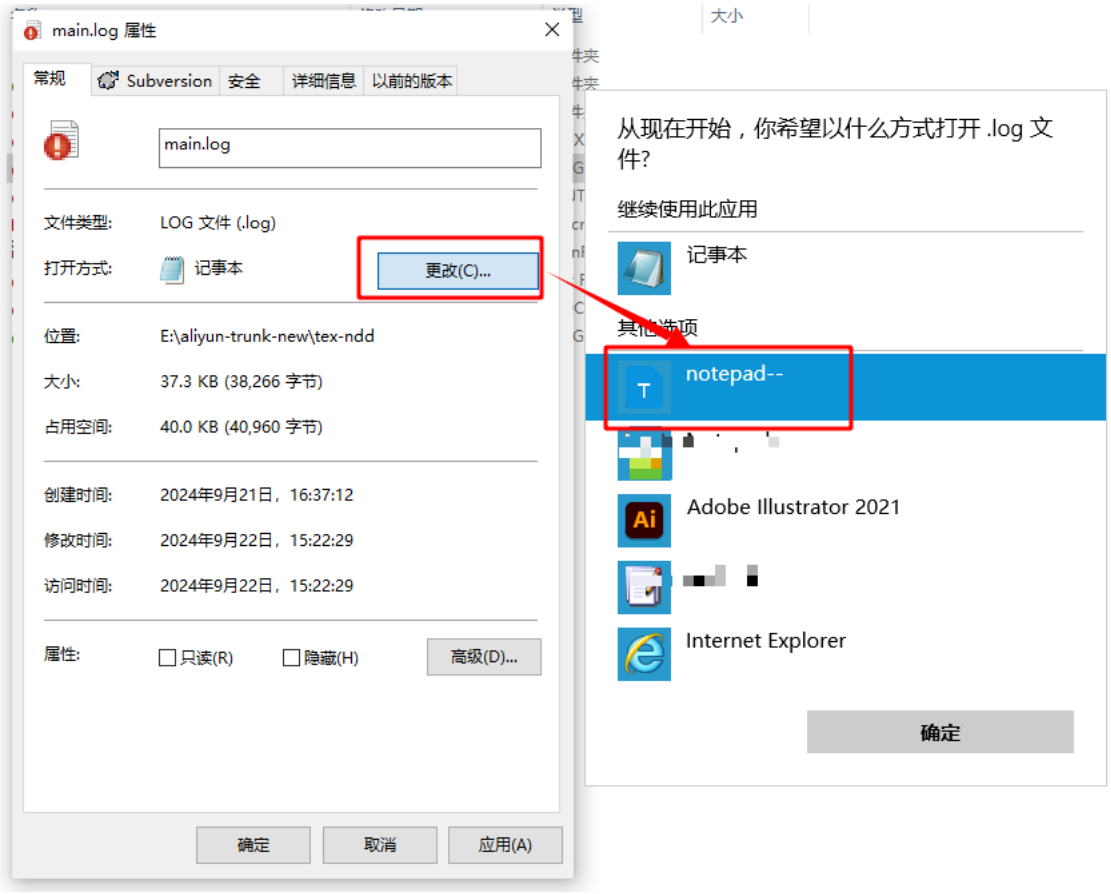
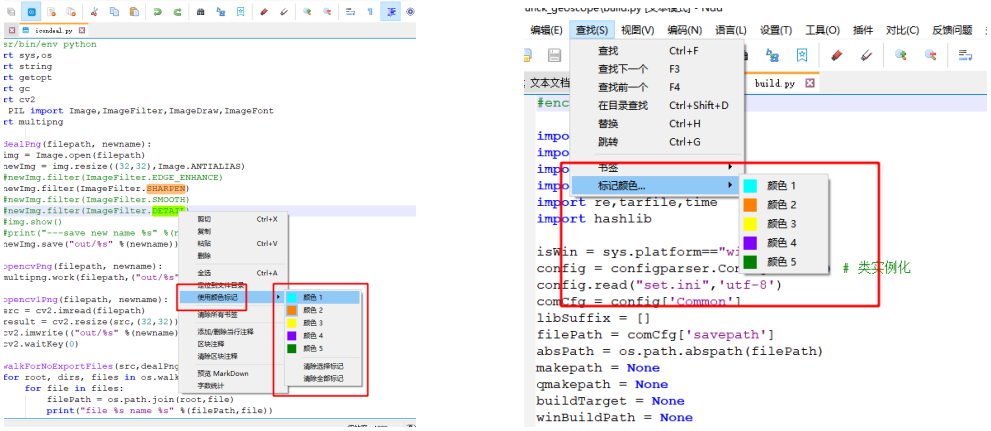


图 3.49: windows 下双击使用 Ndd 打开

更改的程序选择 Notepad-- 的程序即可，如果找不到，在下面选择“此电脑上的其它程序”，找到 Notepad-- 即可。这样后续双击再打开同类后缀的文件，系统都会使用 Ndd 打开。

3.15 文件内容高亮标记

ndd 每种主题支持 5 种不同的颜色，进行文本高亮。每种主题下面，文本高亮的颜色，都是可以自定义的。高亮功能在鼠标右键，和菜单里面都有。如下图所示：



(a) 右键颜色高亮标记 (b) 菜单查找中的高亮标记

图 3.50: 高亮标记文本内容

一般使用右键即可，选择文本后，右键选择颜色进行高亮。也可以使用快捷键 F8 进行单词的高亮选择，默认颜色就是当前选中的高亮色; 反之 F7 为取消高亮，这两个快捷键对应菜单工具栏的如下两个按钮：



图 3.51: F8 F7 高亮和取消高亮

说明：F8 按一次是把光标下的单词高亮，继续按一次 F8 则是取消高亮。F7 则是取消所有的高亮。

3.15.1 自定义高亮标记的颜色

前文说过每种主题支持 5 种高亮颜色，而且该颜色都是可以自定义的。标记样式 1 到 5，在下图界面中分别修改自己喜欢的背景色就好。自定义的界面如下：

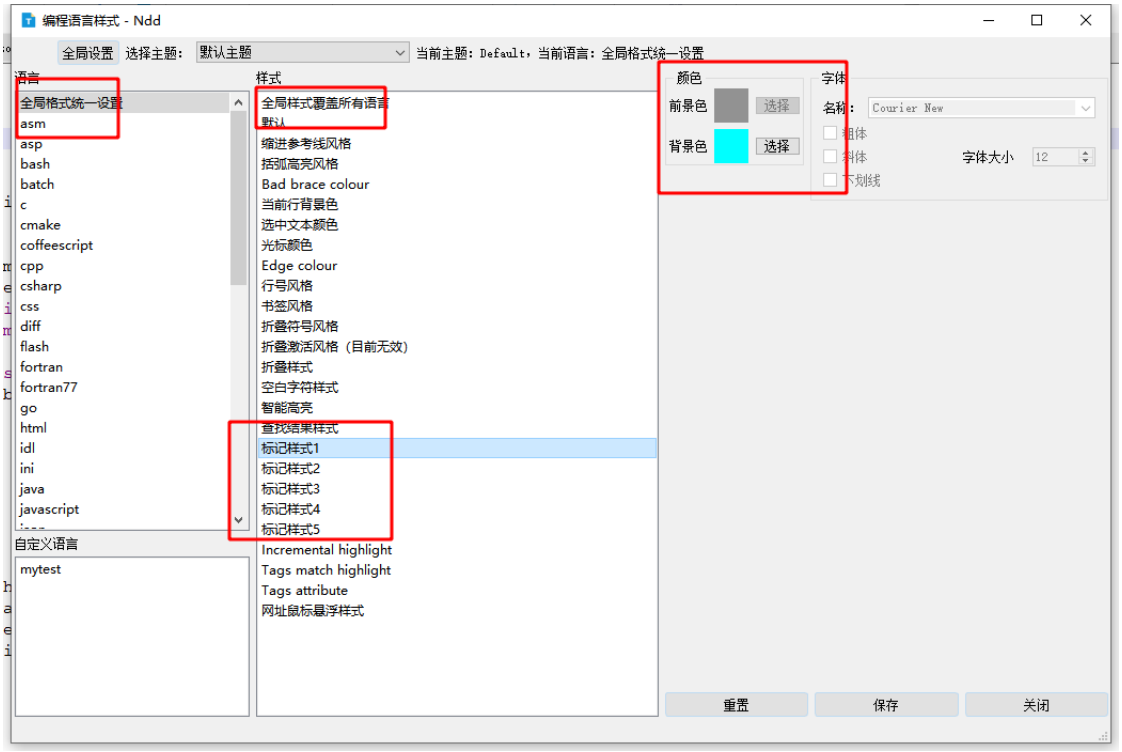


图 3.52: 自定义标记颜色

如上图在全局设置中，分别设置标记样式 1-5 的背景色即可。

3.16 自定义语言并关键字高亮

Ndd 目前支持简单自定义语言，支持关键字高亮；支持类 C 语言的语法高亮。

在菜单 **设置-自定义语言**功能界面中，自定义一个简单的语言，然后指定自定义关键字，对关键字进行高亮。设置步骤如下：

1）依次点击新建、输入自定义的语言名称。如下图所示：

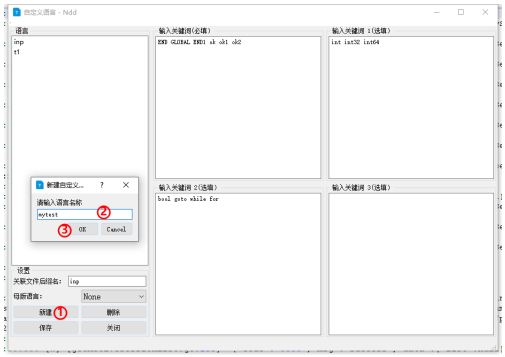


图 3.53: 新建自定义语言

注意这里自定义的语言名称，千万不能和已有语言名称重复，否则冲突！比如已经有 txt 语言了，就不能再自定义一个叫 txt 的语言，必须改成其它不存在的名称。

2）接着在下图 ①②处输入自定义的关键字，目前可支持 4 组关键字。在③中输入该语言关联的文件后缀类型。比如例子中自定义语言名称是 mytest，则关联的后缀名可以取 test ts1 等，注意不能和其它现有语言后缀文件类型冲突！最后保存。如下图：

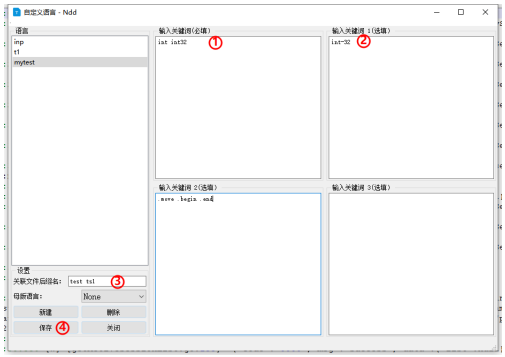


图 3.54: 自定义语言关键字和关联后缀

3）之后凡是打开自定义语言关联后缀的文件，例子中是.test 或.ts1 时，Ndd 会自动使用对应的语法格式进行高亮，目前只支持关键字高亮。对于类 C 语言，则可以支持函数识别，部分语法高亮等。如下图所示：

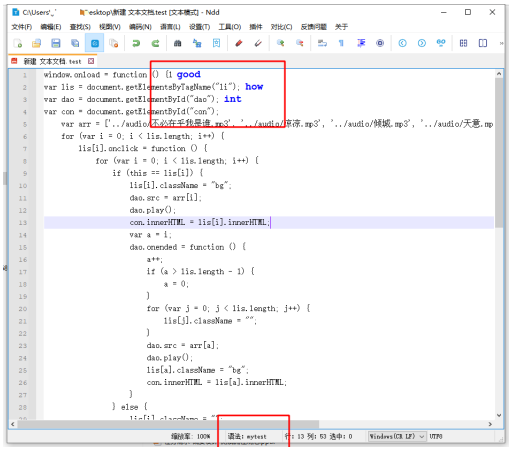


图 3.55: 自定义语言关键字和关联后缀

4) 修改自定义语言关键字的高亮颜色或字体

一旦自定义语言完成后，它就是一个普通的编程语言了。在 **设置-主题与语法设置**中，可选择自定义的语言后，对其关键字的字体和高亮颜色进行设置，如下图所示：

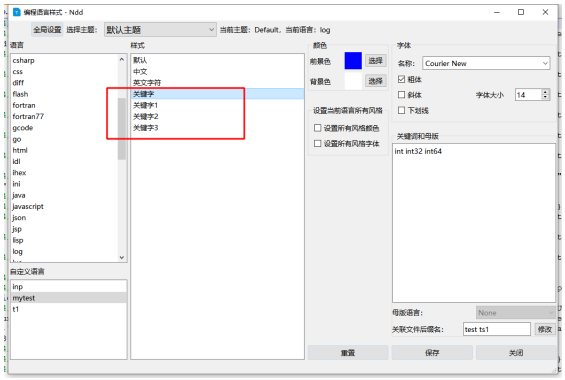


图 3.56: 设置自定义语言关键字高亮

由于笔者当前不准备进一步发展 ide 等高级功能，所以目前自定义语言功能十分有限。

3.17 命令行打开文件，跳转到指定行

命令行下执行命令：Notepad– filepath -n linenum

filepath 文件的路径，-n 行号

这样会打开对应的文件，并且跳转到对应的行号。

3.18 F3F4 选中快速查找

如果希望选中文本后，或者直接在编辑框中光标处，进行快速 F3 进行查找，则设置如下：

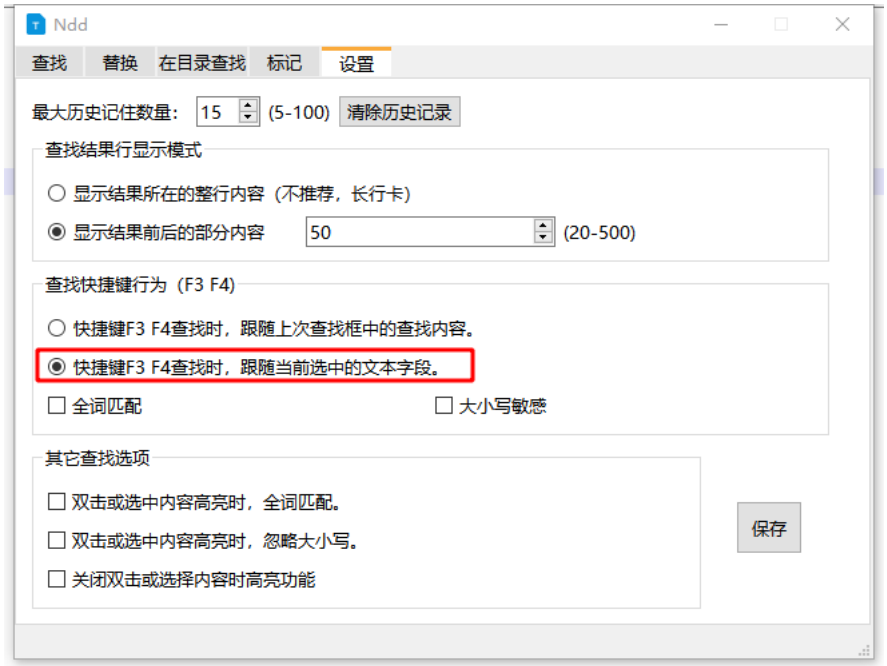


图 3.57: F3F4 快捷查找

Ndd 提供了查找界面，默认情况下查找的内容，是跟随查找框中的查找关键字。有些用户，比如笔者自己，在一些场合下，比较喜欢直接查找当前光标下的单词，或者鼠标选中一小段文字后，进行快速查找，则需要修改该默认配置。使得 F3(查找下一个)，F4(查找上一个)

按下时，直接获取当前选中的字段，然后在编辑框中进行快速查找。

上图3.57中，F3F4 查找时，有两个单独的选项，①全词匹配，② 区分大小写，如果某天你发现 F3F4 快速查找没有达到自己的预期，则需要检查这两个查找选项。

备注：快速查找只发生在“查找框”关闭的情况下。如果“查找框”已经显示出来，则按下 F3F4 是快捷调用查找框中的查找按钮。

3.19 选中文字高亮所有字段

鼠标选中一段文本放开后，编辑框中所有同样的字段，均进行高亮显示，效果如下所示：

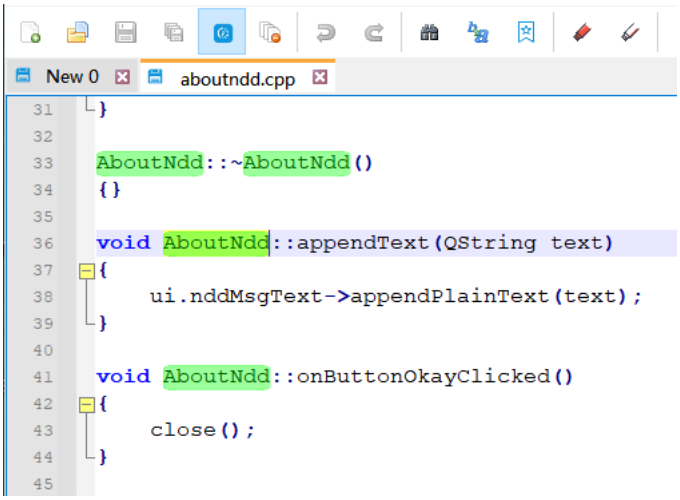


图 3.58: 选中高亮

该功能有关的选项配置如下：

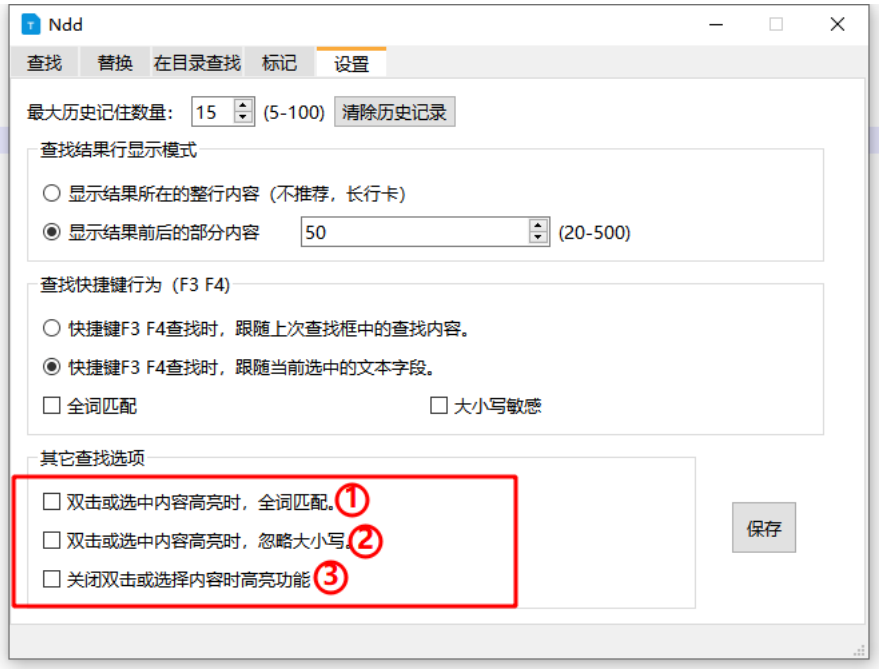


图 3.59: 选中高亮选项

上图3.62中，有三个单独的选项，①全词匹配，② 忽略大小写，③ 关闭该功能。

备注：在 2.20 及之后版本，对工具栏按钮的 F8 功能，其查找高亮的选项也由该配置决定。在 2.20 之前版本，F8 高亮配置跟随查找框中的标记页选项，2.20 中修改为跟随此处设置。

3.20 修改选中文字的底色

鼠标选中一段文本时，选择文字的底色默认是黄色，效果如下：

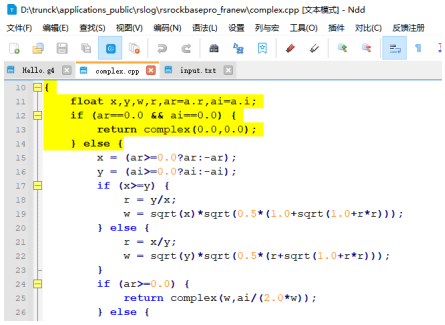


图 3.60: 选中高亮

修改该底色的配置如下：

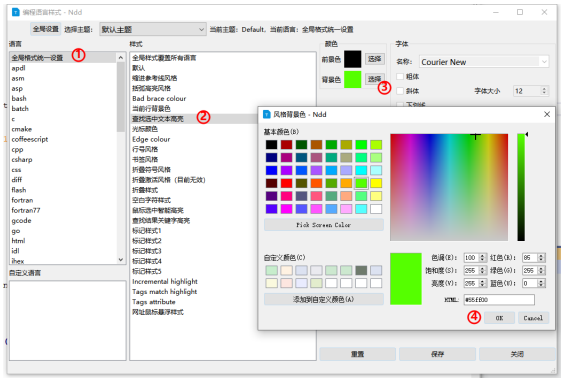


图 3.61: 选中高亮选项

3.21 修改新建文件的默认语法

新建一个文件时，默认使用 txt 语法。由用户说，平时使用 sql 比较多，能否默认新建文件使用 sql 语法。

修改该默认语法的配置如下：


设置 - Ndd
选项
对比文件类型
文本和字体
保存与恢复
字符编码设置
查找选项
新建文件默认编码
UTF8
UTF8-BOM
GBK 18030
BIG5
Shift-JIS
编码检查顺序[打开时]
UTF-8
GB18030
Big5-HKSCS
Shift-JIS
检测编码失败后默认编码
UTF8
GBK 18030
BIG5
Shift-JIS
新建文件
文件换行符号
Windows (\r\n)
Linux (\n)
文件默认图法
txt
确定

图 3.62: 选中高亮选项

Chapter 4

查找替换与常见正则规则

查找和替换在 Ndd 中真是太重要、太常用了，笔者单独写一个章节进行着重介绍。有些小白用户对正则规则不是太清楚，笔者将对正则做一些常用举例或知识普及。

4.1 查找界面主要功能介绍

点击工具栏的查找按钮，或按下快捷键 **Ctrl+F**，可弹出查找界面：



图 4.1: 查找框界面

由于上图中的功能按钮大部分都比较直观，不用说一般用户也都知道。笔者重点对图中功能 ①②③④⑤ 点进行详细介绍。

[1] 计数

顾名思义，统计目标字段在当前文本中出现的次数。

[2] 在当前文件中查找

会把查找的结果，列在主窗口下面的结果框中，如下所示：

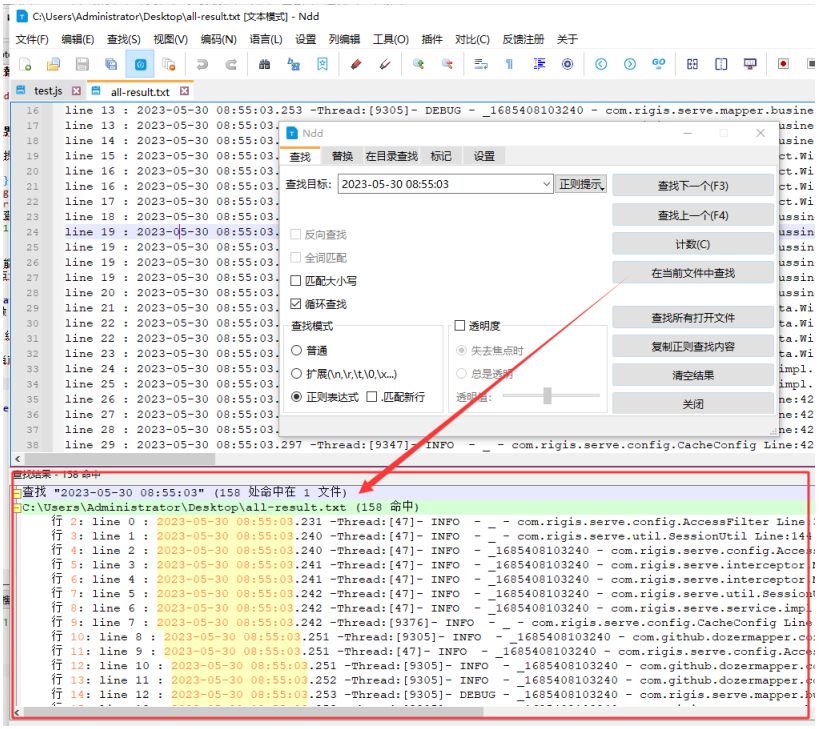


图 4.2: 在当前文件中查找

只需要双击结果框中的一条记录，即可定位到结果在查找文件中的位置。**结果框也是有一些属性配置的，比如结果框可放大、缩小、调整字体、调整高亮颜色、结果长度是显示整行，或显示结果前后一定数量的内容**，均可以通过配置来修改。见 4.2 查找结果框显示效果配置

[3] 扩展规则

如果是查找如下特殊字符，比如制表符\t, 换行符号\n, 回车符号 \r 等，因为这些字符本身不可见 (隐式)，为了能在查找目标输入框中显式查找这类字符，可以输入以上带 \ 的转义符来显式替代。**备注：正则模式可以包括该模式，几乎可直接选正则模式。**

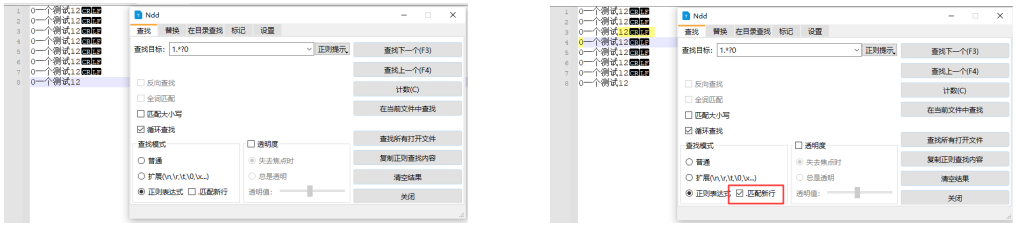
[4] 正则表达式

即在查找目标中输入的正则表达式，比如查找空白字符 [\t]，查找所有中文字符 [\x{4e00}-\x{9fa5}] 等这类正则规则。后续将对常见的正则表达式进行介绍。

[5] “.” 匹配新行

“.” 在正则中，是匹配任意字符的意思，不过这个“任意字符”默认却不包括换行符号。注意不同的正则库其对 “.” 是否包含新行的定义可能不同，Ndd 的 . 在这里默认是未包含新行的。

举例来说，如下例子，下图中的左右区别。正则规则 1.*?0 ，表示查找 “1 后面有任意个字符，然后（非贪婪）接着 0 字符”，找这样规则的目标。左图 . 未勾选包含新行，则查找不到目标（即遇到\r\n 无法匹配，认定正则查找无命中）。右图中 . 勾选包含新行，则会把\r\n 换行符号也视作 . 的一部分，则可以查找到目标。二者的细微差别在此。



(a) . 未包含新行效果　　(b) . 包含新行效果

图 4.3: . 是否包含新行的含义对比

[6] 复制正则查找内容

有时候进行正则查找时，不是希望查询到结果所在的位置，而是希望提取出查询结果，则使用该功能即可，如下图所示：

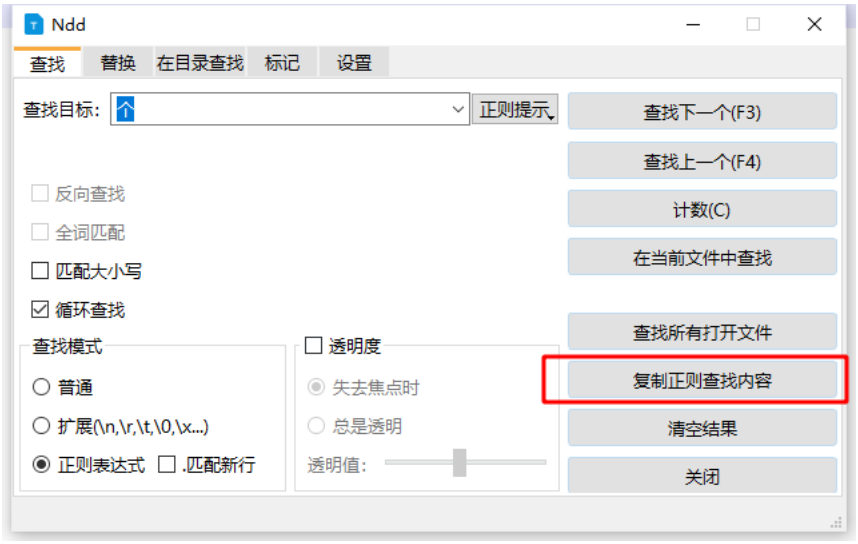


图 4.4: 在当前文件中查找

该功能会把所有查找到的正则结果，复制到系统的剪切板中，用户在其它空白编辑框中粘贴查看结果。

4.2 查找结果框显示效果配置

用户对查找结果框提问比较多的点有如下几个：

① 字体小看不清，界面显示能否放大缩小

② 字体修改，不喜欢默认的宋体字体，在哪儿调整

③ 查找的结果没有显示整行内容，看起来是截断了部分

④ 结果高亮背景、前景色如何自行调整

下面分别做介绍：

[1] 调整显示缩放率

方法 1 在结果框中，按住 Ctrl, 再滑动鼠标滚轮，即可对界面进行放大缩小，而且设置是可以持久化的，下次打开会保存本次的设置。

[2] 结果框默认字体调整

在 Ndd2.18 版本之前，默认字体是宋体。有些 macos 用户说不喜欢宋体，能否有地方调整。在 2.18 版本之后，自动让结果框字体跟随全局默认字体。如下图所示：

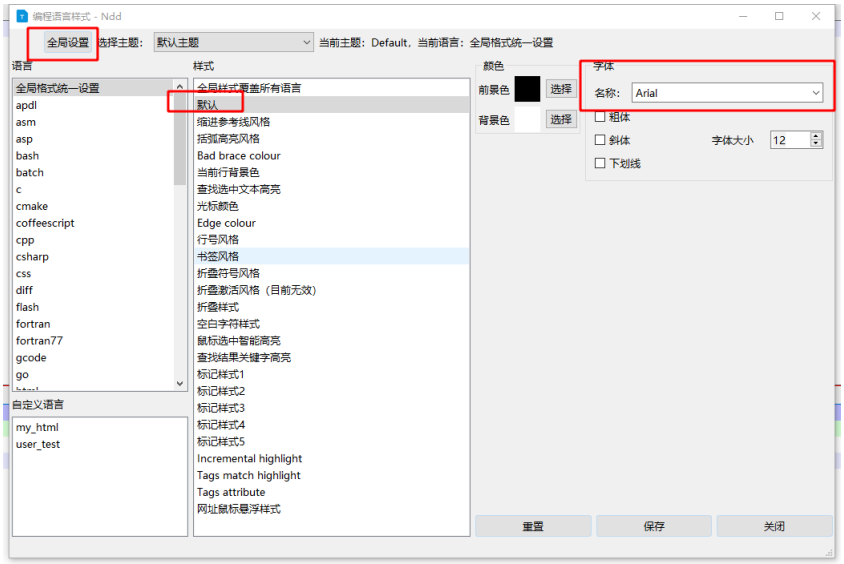


图 4.5: 结果框字体设置方法

一般来说，2.18 之后不需要调整结果框的字体了，因为已经跟随了当前主题的全局默认字体。如果要修改，可直接修改全局默认字体即可。备注：如果要调整全局默认字体大小，则需要慎重，因为会影响全局默认字体在编辑框中的展示，如果不清楚这样做的结果，还请不要随意调整。

[3] 显示结果不全，被截断？

为了提高查找结果的显示速度，在大约 2.14 版本时，查找设置提供了一个选项，是显示结果所在的整行，还是显示前后部分长度的内容。位置如下所示：

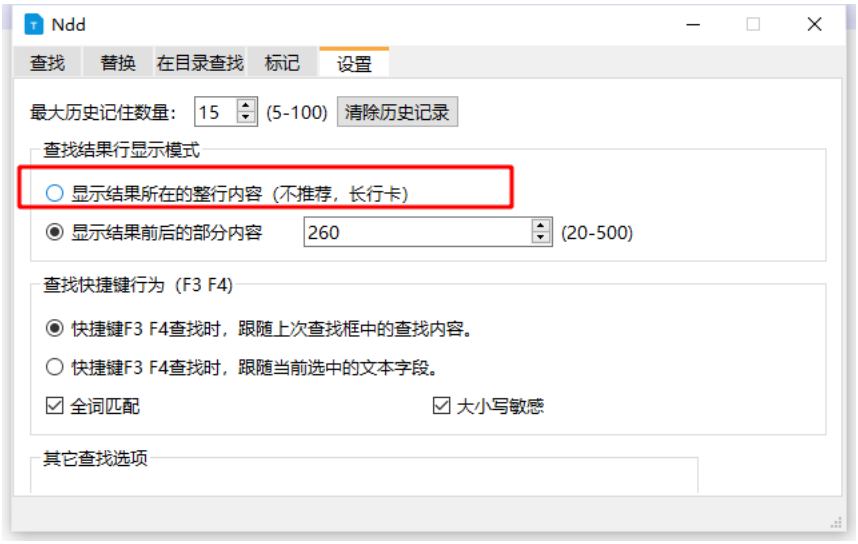


图 4.6: 显示结果长度设置

如果选择上面的整行显示，则会把结果所在的行全部展示在结果框；反之部分显示，则只显示结果字段前后一定长度的内容。这样的好处：在对某些长行文件查找时，避免长度过长导致查找结果过长，避免目标高亮无法出现在视觉中。笔者本人比较喜欢后者，把部分显示调整为 100 个字符，这样不仅显示速度快，而且也不耽误常见情况下对目标字段附近区域的内容展示。

[4] 调整结果高亮背景、前景色

查找结果框中的目标字段，是有高亮前景或背景的，如此让用户定位结果更加清晰。高亮的前景色和背景色，用户可以在 **[设置-编辑语言样式-全局设置-查找结果关键字高亮]** 位置自行调整，如下图所示：

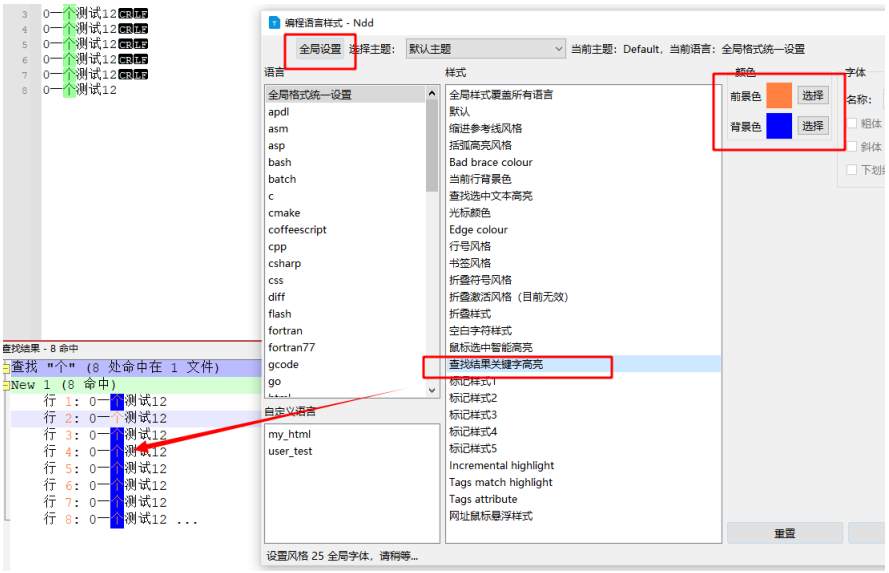


图 4.7: 设置结果字段的高亮色

备注：调整后如果不喜欢，希望恢复默认值，只需要选中目标样式，点击该界面右下角的重置按钮即可。

4.3 在目录中批量查找

详细介绍在目录中查找的功能，界面如下图所示：

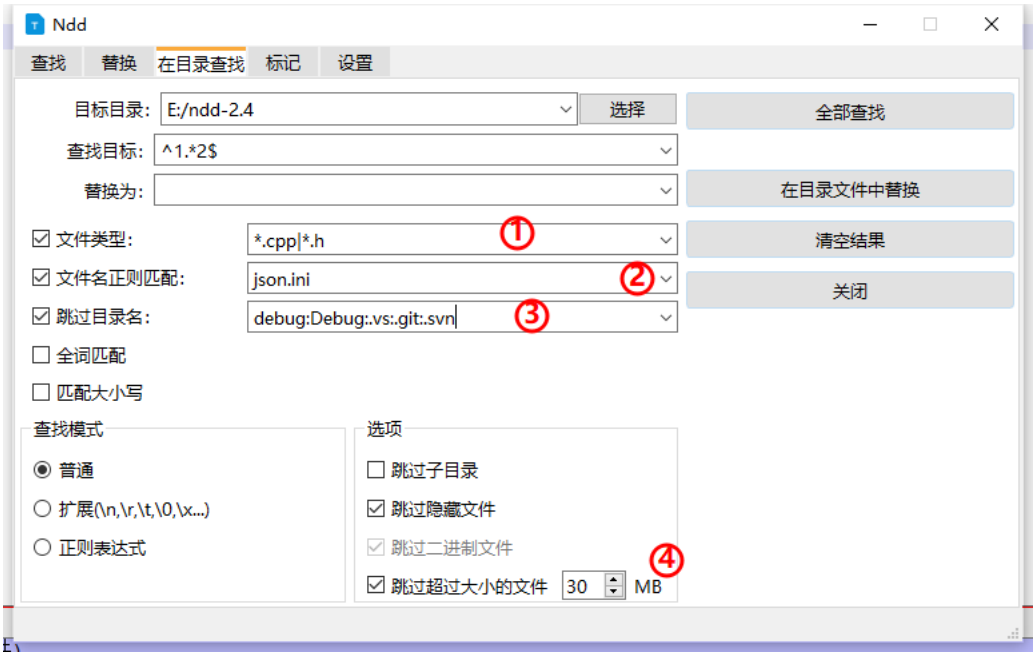


图 4.8: 在目录查找

① 查找的文件类型

为了提速，只查找特定的文件类型，则可使用该选项。比如 C++ 编程人员，往往只对 C++ 源文件和头文件中查找，则规则为 *.cpp|*.h 。依此类推。善用此过滤条件，可加快目录查找的过程。

② 文件名正则匹配

本质作用和① 是类似的，不过适合更复杂的文件名过滤，使用正则规则。

③ 跳过目录名

这里非常有用，对于目录下面有大量子文件夹的查找场景，如果明知道一些中间过程文件夹是不需要查找的，则可排除在这类文件夹中的查找，缩短查找时间。比如笔者是 C++ 程序员，在查找时要对 Debug Release Svn 等目录进行跳过，则按上图中填入跳过一系列文件名即可。

备注：多个字段之间，最开始是使用的“:”号来做分割，在 2.17 版本后，统一使用“|”

号做分割，特此说明。

④ 跳过超过 30MB 大文件

因为人手工写的文件，一般很难超过 1M（1M 即一百万字节，折合汉字大于 30 万到 50 万个），所以一般大文件都是机器生成的。在查找过程中，默认跳过太大的文件，以加快查找速度。这里默认是勾选的，即对查找文件大小有限制。对非注册用户而言，修改该项条件有一些限制。

4.4 常用正则规则介绍

本节不会对所有正则规则面面俱到，只对实际使用中，经常需要的正则规则做介绍。以图带领部分用户进入正则的大门。正则本身是有一定通用逻辑规则，已经有“编程的味道”在里面，部分小白用户可能觉得门槛太高。感兴趣的用户，可以自行学习正则相关的内容进行自我提高。

何为正则查找？即“要查找的关键字，不是一个固定的字段，而是一种满足规则条件下的查找。” 比如要查找所有中文，要查找所有空白字符，这类查找只能在正则规则下进行查找。下面给出几个常见而有用的正则例子。

[1] 查找所有中文

[\x{4e00}-\x{9fa5}]

[2] 查找以 1 开头，2 结尾的任意字符串

^1.*2$

其中 ^ 表示“以什么做开头”，而 $ 表示“以什么做结尾”。更多类似规则\w \d \W \S * . 等正则语法，具体可自行查找正则相关文章或书籍

[3] 正则环视

所谓环视，用大白话讲，就是在查找结果的同时，还要对结果前后的字段进行约束。

举例来说，假设你不光要查找 123 字段，还要满足 123 的前面有个 A，或者 123 的后面有个 B，只有满足这种前后条件的结果 123 字段，才是你需要的结果字段。这就要使用到

环视的技巧。其实了解这个基础概念和使用场景后，你就知道自己在哪些场景下需使用环视。

[备注：以下内容收集来自网络]

环视的作用相当于对所在位置加了一个附加条件，只有满足这个条件，环视子表达式才能匹配成功。

环视按照方向划分有顺序和逆序两种，按照是否匹配有肯定和否定两种，组合起来就有四种环视。顺序环视相当于在当前位置右侧附加一个条件，而逆序环视相当于在当前位置左侧附加一个条件。环视约束条件如下：

环视表达式	说明
(?<=Expression)	逆序肯定环视，表示所在位置左侧能够匹配 Expression
((?<!Expression))	逆序否定环视，表示所在位置左侧不能匹配 Expression
((?=Expression))	顺序肯定环视，表示所在位置右侧能够匹配 Expression
((?!Expression))	顺序否定环视，表示所在位置右侧不能匹配 Expression

用法举例如下：比如需要把 isinf 函数替换为 std::isinf，但是有些地方已经被替换过了，现在需要找到单纯的 isinf 字段，而忽略 std::isinf 字段。这种可以使用逆序否定环视的功能，即查找 isinf 字段的左边不能出现 std::, 具体写法为 **(?<!std::)isinf** 。

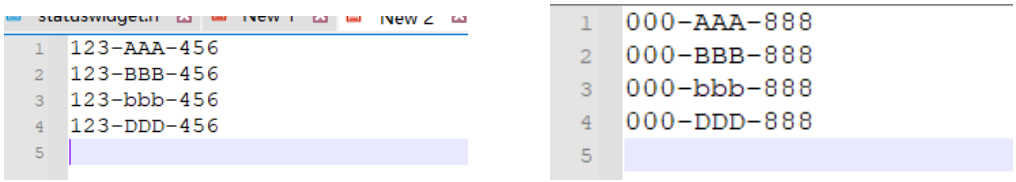
类似如果需要查找 isinf, 不过目标字段右边不能是 x 字符，则可使用顺序否定环视的功能，具体写法为 **isinf(?!x)** 。

笔者不打算展开介绍环视的所有功能，更多的是让读者知道：“有这么一个工具，它可以在什么场景下使用。”因为介绍正则的书籍或网络资料实在太多，限于篇幅有限，此处不赘述。

[4] **分组替换**

说明：这个用法很有用，强烈建议读者好好掌握。先举例来说，这个概念可以做什么？

假设有下图左边的字符串内容，替换规则是，找到满足 123-AAA-456 或 123-BBB-456 这样的字符串，然后对其中的 123 替换为 000，对其中的 456 替换为 888, 而中间的字母字符保持不变。可以实现吗？



(a) 原始内容字符串 (b) 替换后需要的结果

图 4.9: 分组替换需求举例

使用分组便可以完成该要求。在写这个正则规则时，本来可以直接写成 123-.+-456 即可查找，但是为了进行对“部分内容的替换”，要对查找字段进行分组，分组如下：

(123)-(.+)-(456)

每个 () 就是一个分组，上例中的三个 () 分别对应分组 $1 分组 $2 分组 $3 ，形式为 **($1)-($2)-($3)**。替换的要求是，对分组 $1 替换为 000，分组 $2 保持不变依旧为 $2，分组 $3 替换为 888，则替换结果的正则写法为：

(000)-($2)-(888)

最终的在替换界面中的写法如下：



图 4.10: 分组功能详解

笔者理解的简单分组概念：即使用 () 来区分每一个所谓的组，然后分别使用 $n (n 为序

号，从 1 开始) 的命名规则来代称每个组，后续在查找的时候，可以使用 $n 来指代原始查询到某组的内容。如果您的**替换需求类似是：部分关键字内容需要替换，部分关键字内容保持不变，则使用分组来区分它们，把“需替换”和“不需替换”的部分区分开，在替换字段中即可保持“不需要替换”的部分保持不变。**这种类似的替换场景，则可用分组替换正则去完成。

其实这个例子，已经把正则分组的用法讲完了。更详细的用法可翻阅相关专业资料。

Chapter 5

大文件编辑与处理

Ndd 的设计理论上可以支持任意大小文件的编辑。需要说明的是，文件可分为文本文件和二进制文件，前者打开后可直接阅读，后者打开是二进制编码。这里说的大文件编辑，一般是指文本文件的编辑处理。对于二进制文件，Ndd 也可以采用二进制的方式进行编辑。不过总的来说，二者并不可混用，即不可使用文本编辑的方式去编辑一个二进制文件；反之亦不推荐。

5.1 打开大文件

当打开一个比较大的文件时，比如超过 1G 的文件，或者点击 [**菜单-以指定方式打开文件**] 时，会有一个文件打开方式的提示：

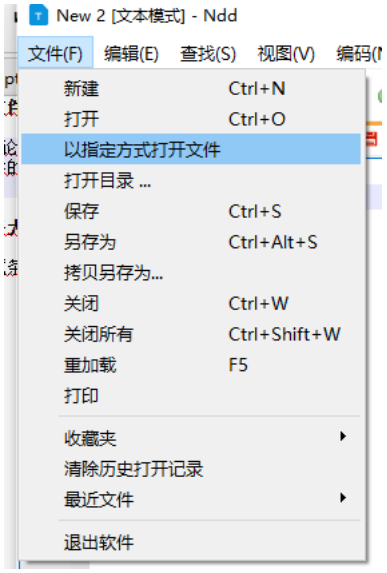


图 5.1: 指定方式打开文件

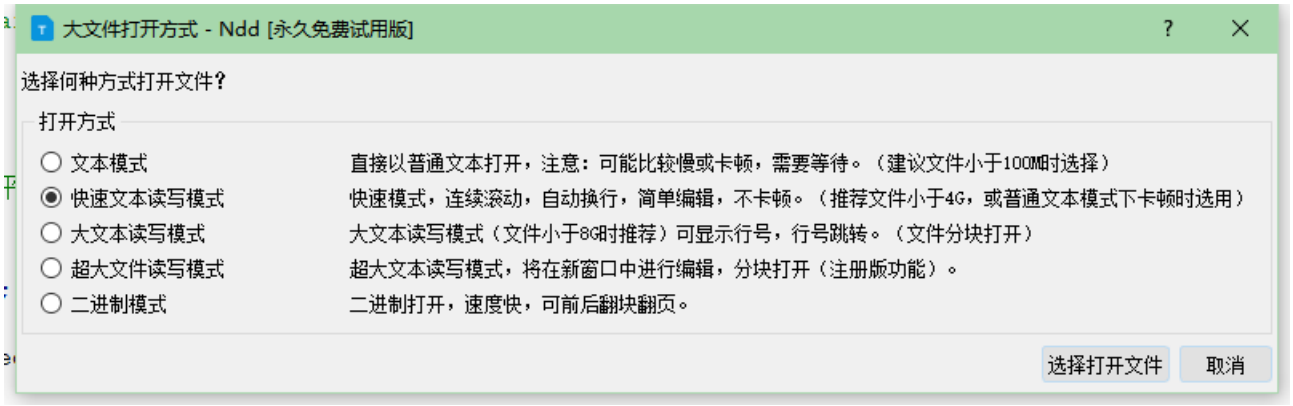


图 5.2: 大文件打开询问方式

此处对不同文件大小，打开时采用或推荐的默认方式如下：

文件大小	默认打开或推荐方式
(size <= 100M)	使用普通编辑方式打开
((size < 6G))	使用快速文本编辑方式打开（4G 可编辑，6G 只读。连续滚动）
((size < 8G))	使用大文本编辑方式打开，分块滚动
((size > 8G))	使用超大文件编辑方式打开，分块滚动
((二进制文件))	使用二进制文件方式打开，分块滚动

表 5.1: 不同大小对于的打开方式

可能有用户说，为什么不统一为一个呢？实在是不同场景有不同的用户体验，无法做到统一。小文件需要编辑快速，使用大文件打开则“杀鸡用牛刀”，体验更慢。另外就是，打开大文件需要更复杂的技术和内存、或文件缓存，这些都是对速度和体验会有影响的。

如上图所示，一般来说，小于 100M 的文件会直接使用普通模式打开。不过超过 40M 的文本文件，默认会关闭自动换行和语法高亮以提高反应速度。此模式可全文查找、实时替换，也是最常用的模式。

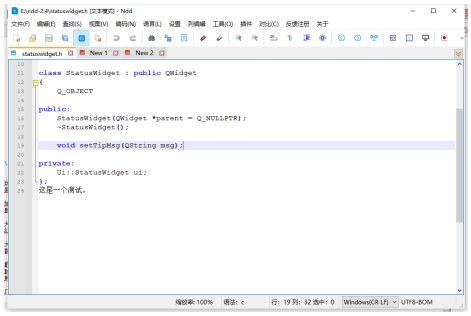
大于 100M 小于 6G 的文件，3.7 中新增快速文本模式打开，此模式可行号跳转，全文查找，连续翻动等。小于 1G 支持全文实时直接替换，大于 1G 则需要使用大文本处理器替换。如果需要连续翻页，查找，简单编辑，则可使用该模式。

大于 100M 小于 8G 的文件，也可使用大文本方式打开，此模式可行号跳转，全文查找等。不支持全文实时直接替换，需要使用大文本处理器替换。此类文件不可能是人工手动编写，几乎都是机器自动生成。如果不需要连续翻页，检查查找，简单编辑，则可使用该模式。

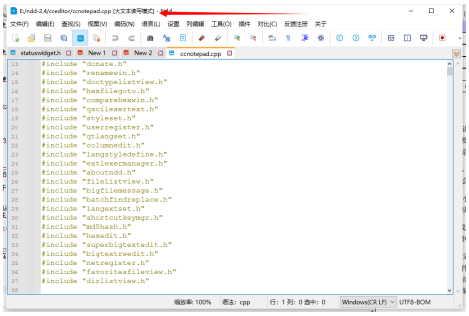
大于 8G，或者超过 10G 的文件，一般无条件以超大文件方式打开，此模式下可全文查找，但是行号跳转需要先遍历行号。不支持全文实时直接替换，需要使用大文本处理器替换。

最后一种“文件分块模式”，一般用的比较少，但是个特例。有些文件，比如 json 和 sql 的压缩文件，文件可能在 10M 或 100M 大小，但是这类文档全文没有一个换行符号，导致一个巨大的文字全部直排在一行，此种情况会导致编辑或显示非常卡顿。于是专门为这种情况提供该方式。如果打开文件时，检查到文件是“巨长行”文件，则默认会使用该该模式打开。

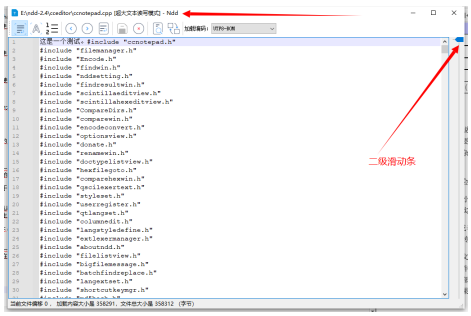
几种打开方式的界面如下：



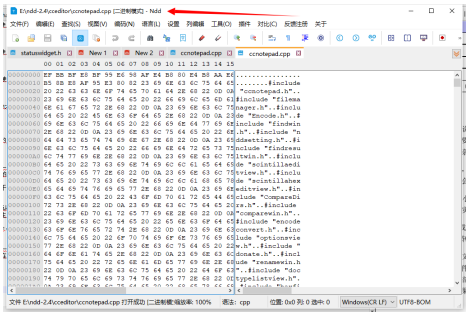
(a) 普通方式



(b) 大文本方式



(c) 超大文本方式



(d) 二进制方式

图 5.3: 几种打开方式展示

说明： [1] 打开方式会在窗口的标题栏进行说明。[2] 大文件和超大文件均有两级进度条，内侧进度条控制当前页面，外侧进度条控制全文页面 [3] 二进制文件小于 1M 是全文打开，大于 1M 则是分页打开。二进制模式下是可以随意修改文件的，需要选中内容后，右键弹出修改功能。

5.2 大文件修改编码

大文件由于太大，比如 5G 大小文件，一次性修改编码时，需要等待时间较长，容易导致界面卡住较长时间。所以 Ndd 不直接在编辑框中实时对大文件进行编码转换。**菜单-工具-大文本处理器**来完成。如下图所示：

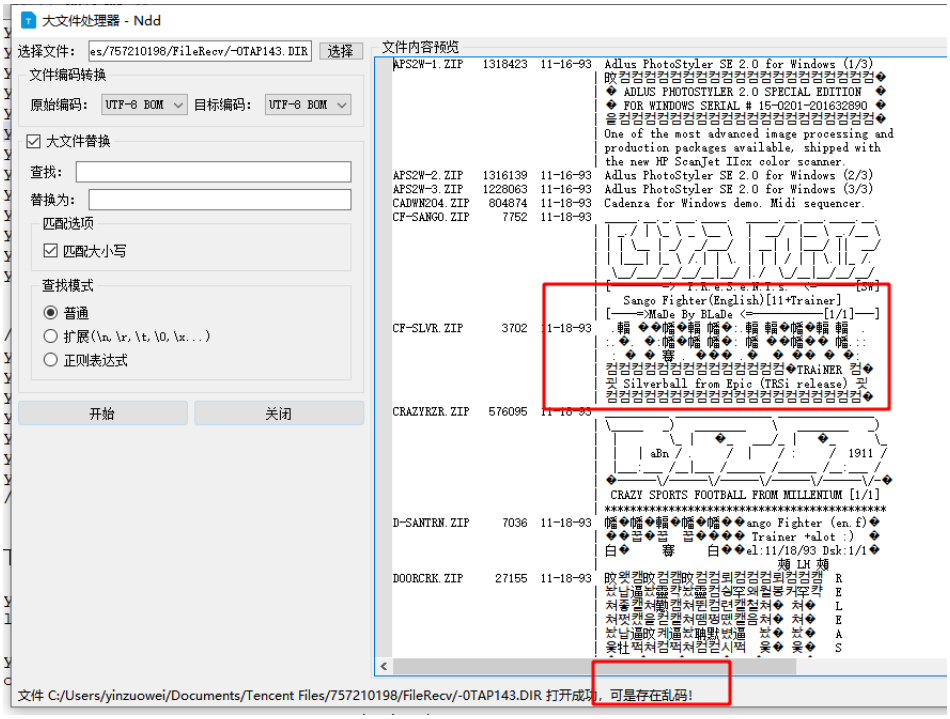


图 5.4: 大文本处理器：编码乱码

对大文件进行编码转换，具体操作过程如下：

1. 打开文件

   选中文件打开后，右边文本框会自动显示该文件的头部部分内容。也会自动检测原始文件的编码。如果编码检测错误，右边文档可能会看到乱码。如果有乱码，说明原始文件编码不明确，需要人工手动观测一下。上图 5.4 大文本处理器：编码乱码 中，显示即乱码了，需要接下来调整。

2. 调整编码

如果发现右边的文件内容预览中存在乱码，说明当前探测该文件的编码类型错误。此时手动调整一下原始编码的类型，看看调整后右边预览文件中乱码是否消失？如果消失，则说明当前手动选择的原始编码是对的。下图 5.4 中手动尝试选择原始编码为 GBK 后，文件编码恢复正常，说明文件原始编码就是 GBK 编码。

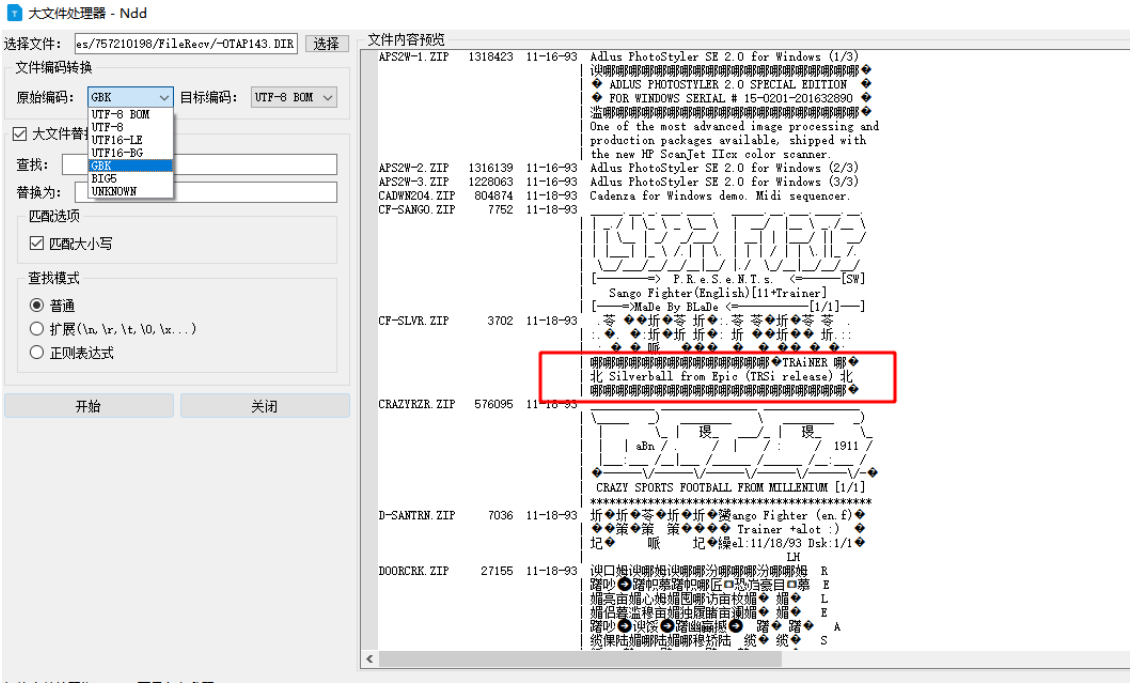


图 5.5: 大文本处理器：选中 GBK 编码后正常

备注：如果右边预留文件框存在乱码，说明编码自动探测失败，此时千万不能强制转换编码。因为原始编码未知，转换必将错误！

3. 编码转换。

   上例中原始编码是 GBK，如果需要把大文件转换为 UTF8 编码，则选择目标编码为 UTF8 即可，点击开始选择保持路径，即开始转换编码 (不勾选下面的文件替换)。此时处理完成后的新文件，其编码是目标编码指定的格式。

备注：编码探测是可能失败的，因为编码理论上有成百上千种，每种都探测则速度很慢，需要精确探测则需要更多文件信息。Ndd 在有限的速度和文本长度下，对大部分常见编码进行了探测。

5.3 大文件文本替换

上例中如果要进行文本的替换，(如果不需要转换编码，则保持原始与目标编码一致即可，但是**原始编码务必要探测或选择正确才行！**）在界面种填入相关查找与替换字段后，进行替换即可。可以支持正则表达式替换。需要说明的是，点开始后，结果文件需要另存为一个文件，没有直接在原始文件上进行修改。这是因为文件较大，为防止操作破坏原始文件，则总是新生成一个处理后的新文件。

在下图例子中，进行了编码转换，把原始 gbk 编码转换为 utf8，把文字“哪”替换为“我”，图例如下：

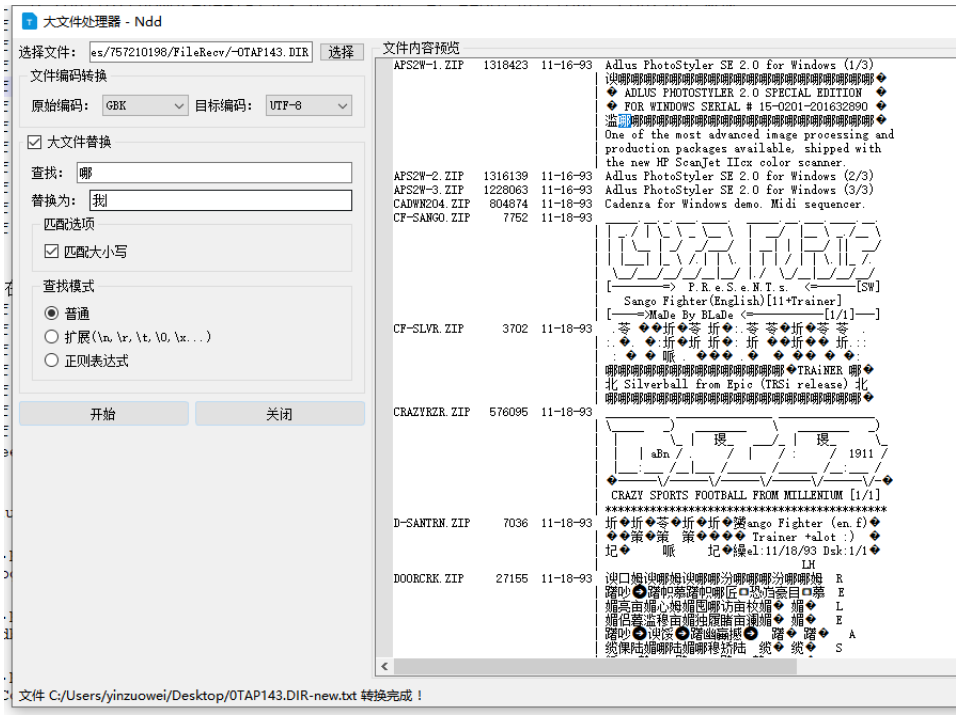


图 5.6: 大文本的替换与编码转换

5.4 万能文本处理器

在 Nddv2.19 版本中，提供了一个功能更加丰富的“万能文本处理器”，支持约 30 种常见的文本处理功能。可支持大文件，在 **菜单-工具-万能文本处器**如下图：

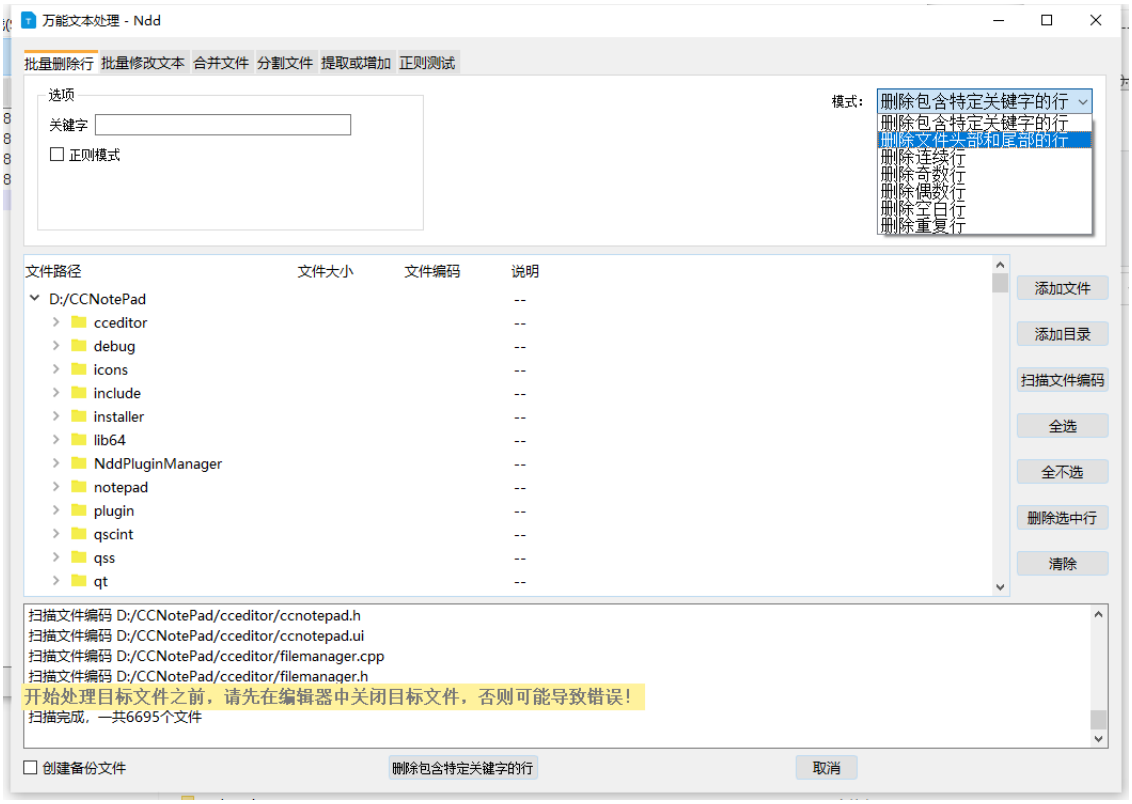


图 5.7: 万能文本处理器

该工具可以对文本进行批量处理。提供了如下 5 大类功能：

功能类	简单说明
批量删除行	提供删除包含关键字行、删除奇偶行、空白行、重复行
批量修改文本	提供删除、替换文本，行前行尾数字或空白，换行转换
合并文件	提供合并文件，文件之间加新内容标记
分割文件	提供按个数分割，或按正则查找关键字分割文件
提取或增加（文本）	提供行前行尾增加内容，提取邮件地址、电话号码等

表 5.2: 万能文本处理 5 大类功能

该万能文本处理中没有包含“编码批量转换功能”。编码批量转换功能在 [**菜单-编码-批量转换编码**]。

Chapter 6

用户相关问题汇总

本章收集 gitee 或 github 上用户提问的相关典型问题，供用户参考。

6.1 打开文件显示乱码

见3.4.1 文件乱码处理

6.2 打开一个 38M 的 sql 文件，查找很卡

这类 sql 文件，因为是机器生成的，里面没有正常的换行符号，人工读取不友好。导致 Ndd 加载的时候只能按照一个巨长的行进行加载，所以导致比较卡顿。

解决方法：在 [**文件-以指定方式打开文件中**] 选择如下方式打开：



图 6.1: 长行文件打开方式

对于 3.7 及以后版本，请选择“快速大文本模式”打开，该模块对于 4G 以下文件，可流畅编辑，连续翻动不卡顿。

6.3 大屏幕下面，界面字体非常小

见2.4 界面字体或图标太小的调整方法

6.4 断电后文件还能找回吗?

见3.10 崩溃后恢复文档和定时保存

6.5 tab 如何使用空格代替

在写 python 等代码中，语法需要对齐缩进，在有 tab 和空格混用的情况下，非常容易出错。此时便有需求，如何将 tab 使用空格代替。

1. 开启“使用空格替换 tab”和确认 tab 的宽度。

   在设置中，如下图所示。勾选上“使用空格替换 tab”，默认 tab 字符的显示长度是 4 个空格。

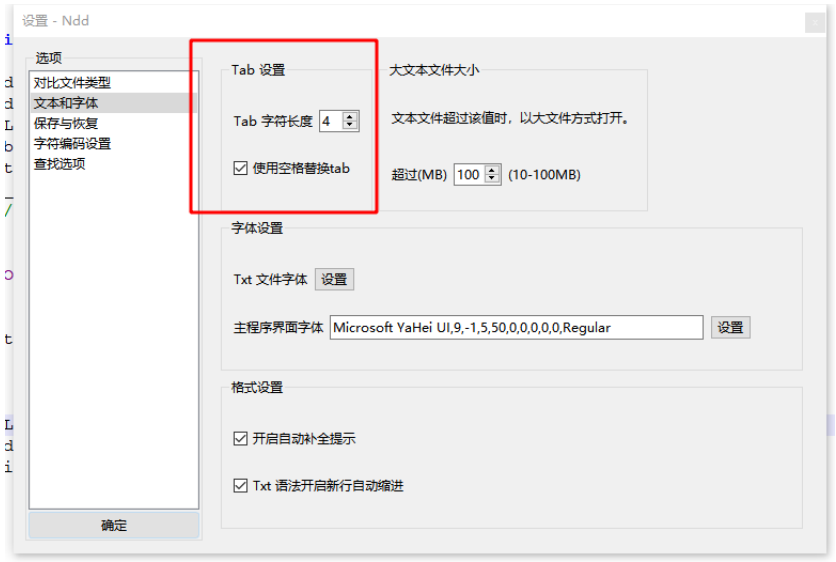


图 6.2: 开启空格替换 tab 设置

注意：这样设置后，后续您在编辑框中输入 tab，都会使用 4 个空格来代替。但是有个问题，**编辑框中之前已经存在的 tab，是不会自动修改为空格的。**如何操作？请看下文 2 节。

2. 如何把文档中的 tab 修改为空格?

   可以先打开空白字符看看，看看现在文档中有哪些是 tab 字符，如下图所示。

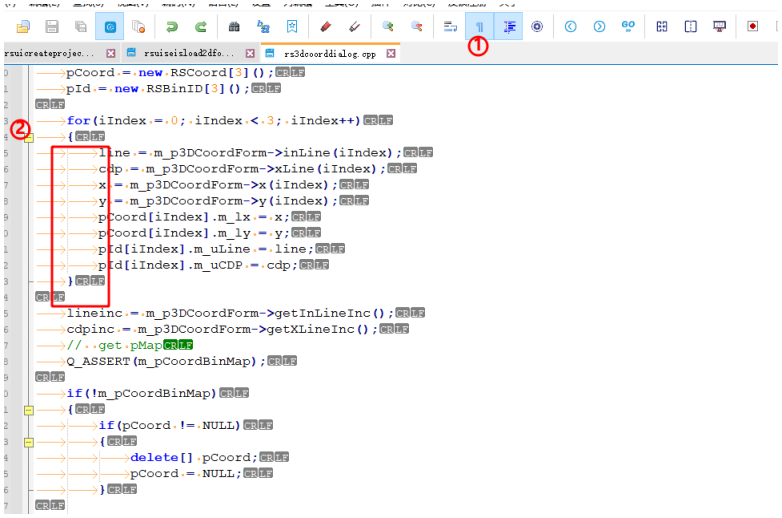


图 6.3: 查看文档的空白字符

点击上图按钮①，可以看到 → 这种类似的字符，就是 tab 字符，见图中红框部分。再选择 [**编辑-空白字符操作-TAB 转空格**] 即可，如下图所示：

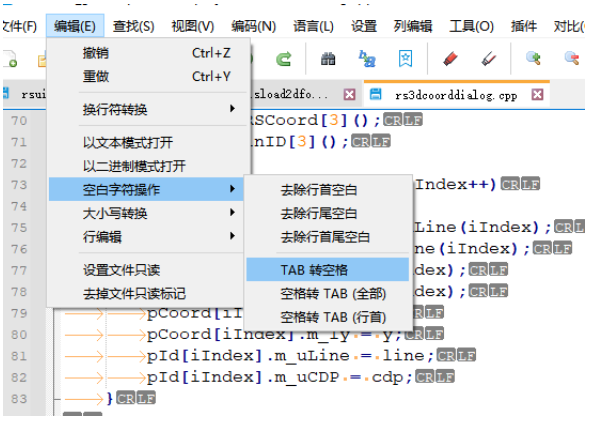


图 6.4: 文档中既有的 tab 转空格

如上操作后，文档中既有的 TAB 都变成空格了。记得保存一下文档。这样既有 tab 已经变为空格，后续敲入 tab 时，也会使用 4 个空格进行替代。

3. 不想修改全局 tab 变空格设置，只想在当前文档中临时生效，如何设置？

1 中的设置是全局的，会对所有文档进行影响。如果只想在当前文档中，使用空格替代 tab 键盘，则在文档中右键，有如下选项，把下图中的选项① 勾选即可。② 处在写 python 的时候也比较方便，勾选一下，就可以看到不显示的空格符号，避免语法对齐错误。



图 6.5: 当前文档开启显示空格和 tab 转空格处理

6.6 注册版和非注册的区别

目前软件主体永久免费使用。windows 版本会永久免费使用。macOS 如果上架到 mac 商场，后续会进行收费；但价格肯定是同类产品的一半或三分之一。主要区别如下：

1）免费版会在标题栏显示免费版；注册版不会。2）非注册版在使用时，会弹出请注册的提示，注册版则不会有该提示。或者当用户使用一段时间后，偶尔会弹出提示用户注册的提示，该提示可手动关闭，用户可以继续使用软件。

我们明白让用户强制注册，在当前中国现状下不太适合。作者认真维护软件产品，付出的劳动价值，需要软件收入去做持续开发。只能在不影响用户实际大部分功能的情况下，在显示或提示上做一些区分。大约 5% 左右的高级版功能，需要注册才不提示或可使用，这部分高级功能比较少。

6.7 赞赏多少金额会发注册码？

6.7.1 对赞赏 Ndd 的说明

首先声明，Ndd 把小额赞赏作为一种用户快速获取软件注册码的渠道，本质是一种等价交换行为，即通过赞赏渠道来获取 ndd 正版注册码。如果您需要 ndd 注册码，完全可以通过拼多多商店来购买正版注册码。有时候在国内一般情况下，捐赠和赞赏的含义可能类似，但有本质区别。

笔者再次重申，请确保您对 ndd 的赞赏支持是因为 Ndd 软件对您工作中有作用，您赞赏软件是为了让作者有更多时间去改进软件，或换取正版注册码，而不是单纯出于同情作者而无偿捐赠。

赞赏是随您个人的意愿，只要赞赏 2 元及以上，附加留言邮箱地址，都会发送注册码，不同金额对应不同类型的注册码。有些用户故意赞赏 0.01 元或者 0.5 元，这种不会发送注册码。因为维护注册云服务器，还有发送邮件、收集邮件都是需要时间及人力成本的，笔者没有精力去对付这种 0.01 类似的赞赏赠。**请不要赞赏 2 元以下金额，2 元以下不发注册码。** 需要说明的是，不同的赞赏金额，对应的注册码类型是不一样的，其能绑定的电脑数量和有效年限是有区别的。

有些用户说，赞赏后不知道注册码可以使用多久。现在统一说明一下，10 元套餐是 3 台电脑可使用 2 年。简单换算下来，费用为一台电脑使用一年的费用不到 2 元，基本等于一瓶矿泉水的价格。用户赞赏 15 元，可以获取一个支持 2 台电脑的永久注册码，永久是指注册过的电脑可以永久使用，直到电脑退休报废，但是不能反注册更换电脑，即支持 2 台电脑注册并用到电脑报废，第 3 台电脑则不行。

赞赏 10 元及以上的用户，建议直接上拼多多购买特定的注册码套餐，因为拼多多不能支持永久套餐，为符合拼多多合规要求，拼多多店铺售卖的 5 年有效期注册码，服务支持有效期为 5 年，本质还是可永久使用，注册过的电脑可一直用到退休，不支持反解绑换电脑。

拼多多店铺如下图所示：



图 6.6: Ndd 拼多多购买注册码途径

同时支持支付宝扫码在线无人自动售卖小额注册码：



图 6.7: Ndd 支付宝扫描支付购买小额注册码

6.7.2 不能联网的用户，如何进行本地注册？

1 如果可以访问网页，则在注册界面，输入注册码，点击获取离线本地码（需要先输入已有的注册码才行）。在弹出的网页中自助获取离线码。获取离线注册码后，在注册界面输入离线注册码，完成注册。图示如下：



图 6.8: Ndd 注册界面点击获取离线本地码



图 6.9: Ndd 通过弹出的网页获取离线码

2 如果连网页也访问不了，则通过拼多多或微信公众号，把机器码和注册码发给作者，作者会给您生成离线本地码。作者的微信公众号二维码如下：



图 6.10: Ndd 通过微信公众号获取离线码

6.7.3 为什么没有收到注册码？

一般有如下几种原因：

① 赞赏额度小于 2 元，如 0.1 0.5 1.0 这种。

② 有些用户留的邮箱存在字母错误，邮件发送失败。

③ 有些用户留的 QQ 邮箱，设置了拒绝接收邮件。

④ 邮件服务器发生错误，有一次阿里云事件，有些阿里云邮箱无法接收邮箱，此种情况很少。

在②③④情况下，笔者其实已经发送了邮件，但是邮件发送失败。一般如果 1 个小时内，邮箱没收到注册码，则可以加 QQ 群，找群主反馈。或者直接给群主发送邮件（邮件贴上赞赏记录），群主邮箱 757210198@qq.com 。**备注：少于 2 元的赞赏，请自动补齐 2 元额度后，才会发注册码。**

6.8 注册失败

在线注册是通过网络进行的，需要注意如下情况：

① 电脑能否连接外网。

② 是否有防火墙对网络有拦截。

③ 是不是还用的比较旧的 Ndd 版本，建议至少升级到 v2.14 之后的版本。

修改相关设置，确保以上条件满足后，重启 Ndd 后重新注册即可。无法在线注册的用户，把本地码和注册码邮件发送给作者邮箱 757210198@qq.com ，会给您发送本地注册码。也可通过上述获取本地离线码 6.8 的方式进行离线本地注册。

6.9 注册码是如何绑定电脑的，重装系统算多个注册吗？

1. 注册码是如何绑定电脑的？

   在电脑上注册时，会通过网络在线注册，一旦注册完成，电脑的机器码会写入到后台注册服务器中。对于不能联网注册的机器，需要把机器码发送给作者，会发送本地注册码给用户，从而完成本地注册。在 Ndd 3.2 版本后，可通过网页自助获取本地离线注册码。

2. 重装系统会影响注册，算占用多个绑定吗？

   只要电脑不更换，随便重置系统，都是可以再次使用注册码来激活的。同一台电脑注册多次，都只算绑定一台电脑，不会重复计算。(如果遇到重复占用问题，可以升级到 3.0 版本一下。或者邮件发给作者，附上注册码)。套餐 345 的注册码是永久有效的，只要不换电脑，可以永久使用下去。套餐 12 的注册码有使用年限。

3. 注册码可以反注册吗？

   不支持。毕竟软件售价和收入已经低到“笔者甚至不再想打理这门生意”的地步。(一顿饭 15 块钱的事，2 台电脑可终身使用，实在谈不上肉疼)。9.9 元特价套餐是 3 台电脑可使用 2 年。简单换算下来，费用为一台电脑使用一年的费用不到 2 元，基本等于一瓶矿泉水的价格。不愿意付费的用户，欢迎使用免费版即可。笔者没有花费大量精力去设计复杂的注册机制，更愿意用户花少量的钱，就能使用几乎免费的软件。请不要来信或发帖讨论降价的事情，为了 3 块 5 块和笔者讨价还价，笔者懒于应付这些几块钱的“生意”，请直接使用免费版即可。

   笔者愿意走中医的思路去做软件，经济条件不允许的用户，可以免费使用软件；经济条件好的用户，您多赞赏笔者也欢迎，毕竟作者也需要生存。即“作者获利于经济条件好的用户，从而让经济条件不好的用户，也能免费使用软件。”

Ndd 作者承诺：绝不引入广告！绝不驻留用户的电脑！关闭就是退出，不做霸占用户电脑资源的流氓软件！还请从 Ndd 界面提供的地址下载软件，避免从其它网站下载到二次修改后的流氓软件。

6.10 为什么不走开源免费的道路呢？

实在是一言难尽。见 1.4 后续还会继续开源吗？ 。总的说来，国内开源就是赔本赚吆喝，用工环境和国情在此摆着，不再赘述。

6.11 如何一键修改所有语言的全部字体和大小

有用户说，需要把所有语言对应的字体、字体大小，统一设置为一种，一个一个语言去设置太麻烦，有没有快速方法？是可以的，在 [**设置-主题语法样式**] 中，操作如下图：

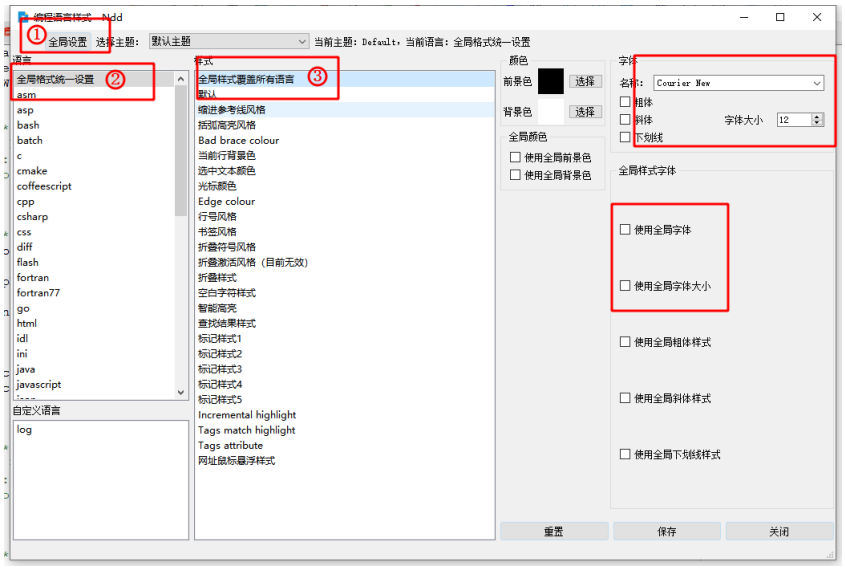


图 6.11: 一键设置主题下所有语言的字体

上图中，依次选择 ① 全局设置 ② 全局格式统一设置 ③ 全局样式覆盖所有语言。

然后勾选图中红框的“使用全局字体”和“使用全局字体大小“两个选项。此时再修改字体和字体大小，就会一键修改全部语法的字体和字体大小了。

如果修改后觉得不满意，又想恢复回去，还是在这个界面，点击下面的重置按钮即可。重置后设置会恢复到系统默认的风格。

6.12 中英文字体为什么没有对齐？

现象如下图所示：

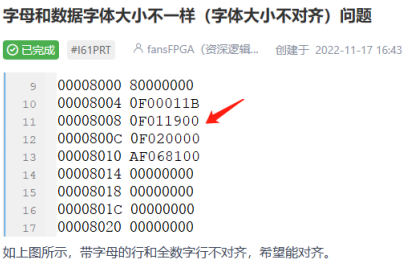


图 6.12: 字体宽度不对齐

有些时候，需要让中文和英文对齐，这样方便列编辑。总的来说，与当前主题设置的字体有关。每种主题下面的每种语法，都是可以自定义字体的。**只有使用等宽字体，才能让编辑器中的文本看起来是对齐的。**

具体来说，如果有中文，需要支持中文字体编辑下面的对齐，建议使用宋体或新宋体字体，此两种字体是等宽字体。这样在有中文和英文混合的编辑情况下，才能让文本对齐。

如果是国外用户，不需要中文，全部是英文，则推荐使用英文等宽字体 coriew new 字体，这种字体对英文是等宽的，但是不支持中文。

后续如果发现字体重叠，字体对不齐问题，请务必通过全局设置修改一下字体, 操作见 6.11 如何一键修改所有语言的全部字体和大小 。推荐使用等宽字体。但是目前还没有发现一种字体，即支持中文等宽，也支持西文等宽，同时还能比较美观。虽然宋体是可以的，但是宋体下面的西文标点符号，看起来怪怪的。修改 txt 文件语法的字体，见 3.2.1 设置 txt 文件语

法对应的字体

6.13 英文操作系统下，字体和光标未能对应

现象如下图所示

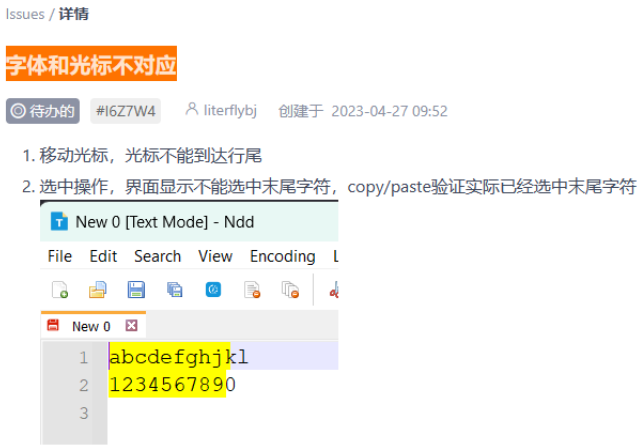


图 6.13: 字体宽度不对齐

解决方案：

1）安装字体或修改字体。英文系统下没有默认字体宋体导致，建议安装宋体字体。

2）或者统一修改 txt 语法字体 coriew new 即可。英文目前默认 txt 字体是宋体，而英文字体没有该字体，修改为系统有的英文字体 coriew new 即可。或者直接在全局里面修改，统一修改全局字体为 coriew new。修改 txt 文件语法的字体，见 3.2.1 设置 txt 文件语法对应的字体

6.14 日文 shift-jis 编码的文件会出现文字覆盖的现象

前文已经已提过，这种多半是字体缺少，或者字体不是等宽导致。统一设置为宋体或 Courier New 字体。

6.15 Macos 安装不上，报错包损坏？

见 2.3.2 Macos 安装包损坏报错解决

6.16 文字发生了重叠和拥挤，高 dpi 双屏切换后，文字错乱等问题。

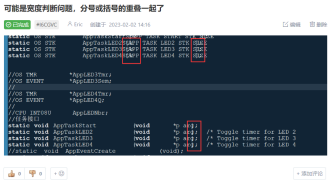
现象类似如下三类：



(a) windows 远程显示错乱



(b) 文字显示错乱



(c) 英文字符错乱

图 6.14: 几种显示错乱现象

问题（a) 往往发生在多个屏幕分辨率不相等的场景下。比如两个显示器，分辨率不一样；还有使用投影仪，投影显示；还有高清屏幕，使用了一些屏幕缩放的配置等待。此类问题在 2.14 之后版本已经解决。

问题（b)(c) 可能是系统缺少对应的字体，或者使用了过于小众的字体。此时最简单的方法，是一键修改全局所有字体，为系统中存在的一种等宽字体即可解决。操作方法见 6.11 如

何一键修改所有语言的全部字体和大小

如果以上方法未能简单解决问题，则可尝试如下方法：

建议尝试使用如下方式解决：在 ndd 的运行目录下面，新建一个 qt.conf 文件，其中文件内容为：

[Platforms][Platforms]

WindowsArguments = dpiawareness=1

然后重启 ndd 即可。这样即可解决因为多个屏幕分辨率不同，或者投影高清缩放等带来的文字重叠问题。

或者也可以“按住 Ctrl，鼠标滚轮随意滚动”，也就是随意缩放一下，文字渲染就不会再重叠了。(此方法的环境：win10 企业版 1809，其他环境下可以尝试使用此方法）另外 Ndd 提供了 qt6 的版本，具体可以在 123 云盘下载见下载与更新软件，已经解决了这部分问题。

6.17 英文单词换行时从中间断开

在开启自动换行模式下，有 2 种换行模式，一种是按中文，另外一种是按单词。备注：只有开启自动换行，才能看到该菜单选项。右键菜单，做如下切换即可。

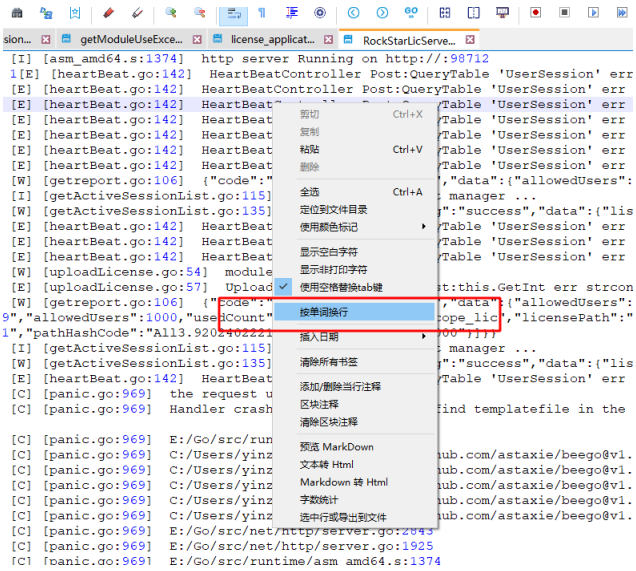


图 6.15: 切换到按英文单词换行

如果您的编辑框中全是英文，日常也大部分在英文环境下面，则切换为按单词换行即可。

6.18 在目录中查找不到结果字段文件

有用户反馈，在目录中查找时，文件中明明有查找字段，但是却查询不到结果。

首先如果您的版本低于 2.20 版本，还请升级到 2.20 版本及以后一下，在 2.18 之前对编码错误的文件会自动跳过。在 2.18 中对编码错误的文件进行了二次编码探测，如果实在编码错误也会显示在跳过文件个数中。2.20 解决了只读文件跳过的问题。

此外，还请注意查找的默认选项，文件目录查找的选项有如下几个，见下图中①②③④：

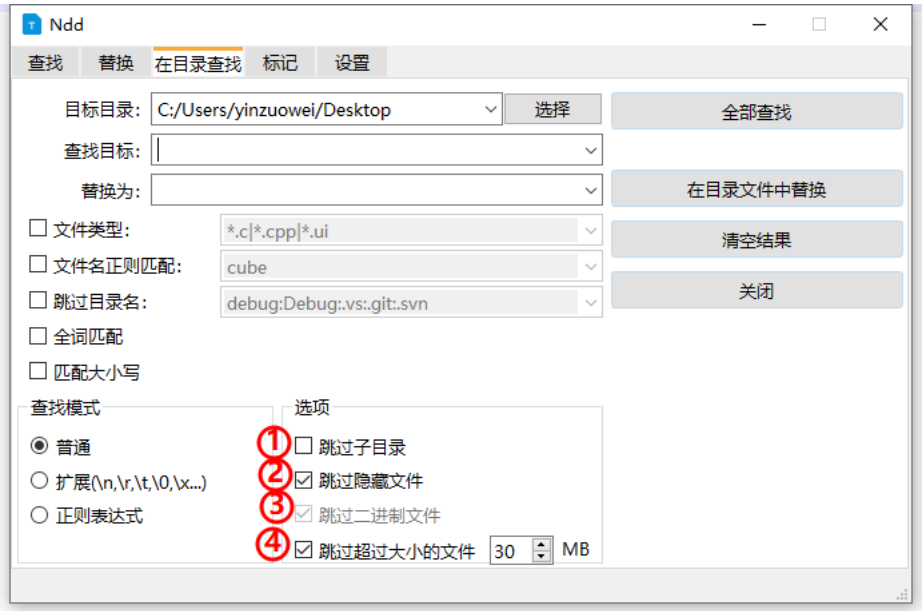


图 6.16: 目录查找选项

① 跳过子目录，默认不勾选。如果勾选，则只查找当前目录根目录下的文件，其余目录则跳过。

② 跳过隐藏文件，默认勾选。隐藏文件在 linux 下面一般是以“.”开头的文件，这部分文件往往是一些系统配置或保护文件，不建议随意修改。Ndd 对这些文件默认不查找，这样可能更加安全合理。如果需要放开，可去掉勾选该选项即可。

③ 跳过二进制文件，默认勾选，禁止放开。二进制文件是不能以文本方式查找的。

④ 跳过超过 30M 大小的文件，默认勾选。如果需要调整大小，可能需要注册才行。一般来说 5M 的文本已经是巨大文件，人工手写不可能有这么大的文件。超过 1M 的纯文本文件往往都是机器生成。这样是为了加快查找速度。如果需要放开，可去掉勾选该选项即可。

总而言之，如果在目录中没有查询到文件，还请注意版本是否为 2.18 之后，再检查是否是因为跳过了“隐藏文件”和“超过一定大小的文件”。

6.19 文字背景出现颜色异常

有用户反馈，在某些暗色背景下，或者某些语言语法下，文字出现“黑一块，白一块”的颜色异常，如下所示：

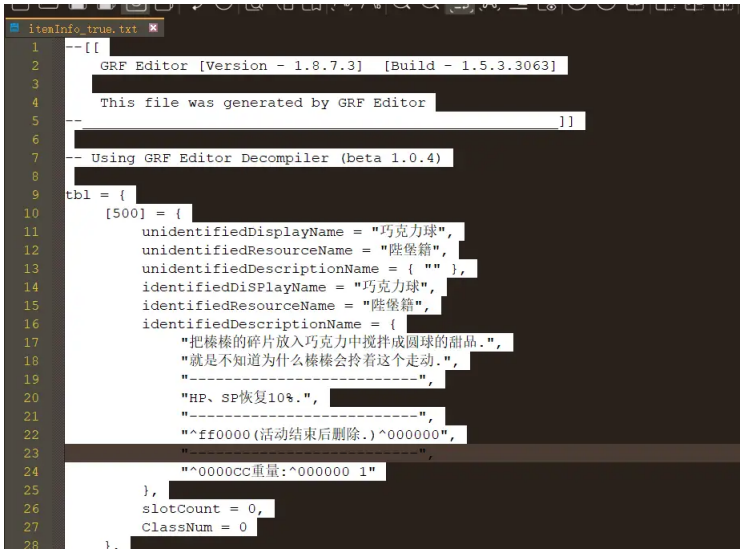


图 6.17: 文字背景异常

解决方法如下：

重置对应语法的颜色设置

有时候升级软件后，因为前后版本的兼容性或颜色设置发生变化，需要手动把对应语法的设置重置一下。比如上图中是 txt 语法，则在 **设置-主题与语法样式**中，把对应语法的高亮设置重置一下。操作如下图：

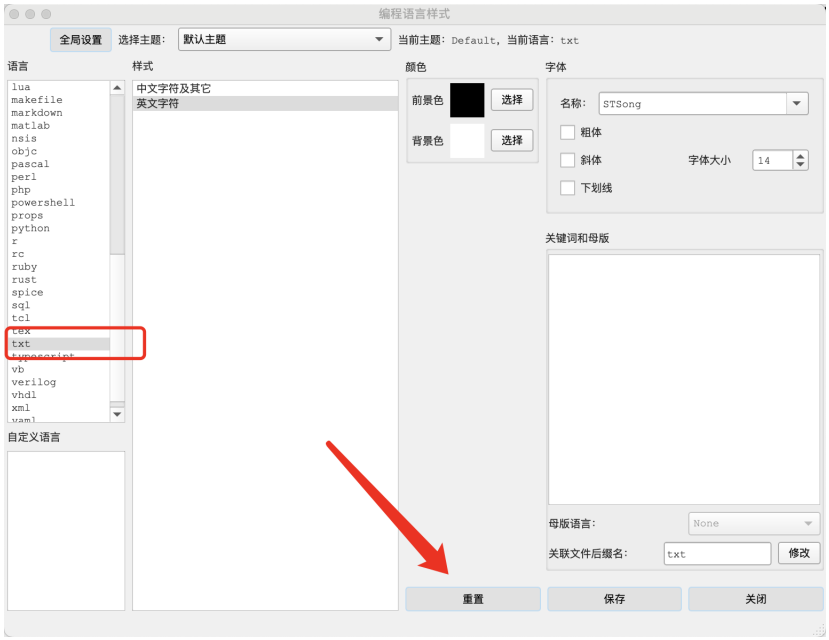


图 6.18: 重置对应语法设置

如果以上操作后，还是没有恢复正常，则说明就是当前语法字段缺少正确默认设置。只需要在图6.18中自行修改各样式的背景或前景色即可。详细的操作说明可参考3.2.2 设置 txt 文件显示的背景色 (例子中是以 txt 举例，其它语言语法类似修改)

6.20 升级软件后再次打开程序崩溃

这种情况可能出现在升级版本跨度太大导致。如果是安装版，则彻底卸载软件后，重新安装。

如果是非安装版 Ndd，则找到缓存目录，关闭软件后，把整个缓存目录删除后重新打开软件，缓存目录位置见 3.10 崩溃后恢复文档和定时保存。

Chapter 7

在 Notepad-- 中使用 pinlang

在 Ndd 3.3 版本开始，内部集成了 pinlang 插件。可以直接把 Ndd 做一个微型的 pinlang 开发运行环境。另外编辑器提供部分操作编辑器的方法函数，供希望进行编辑器脚本开发的用户使用。

如果您是在 Ndd 中使用 pinlang，本章对相关使用方法和 Ndd 特有 api 做一些介绍。反之如果您是直接在命令行中使用单程序 pinlang，则无需阅读此节。需要说明的是，只有集成在 Ndd 中的 pinlang 才有编辑器专有 api 接口。

7.1 Ndd 中 Pin 插件

在 Ndd 3.3 版本中，加入了 Pin 插件。可以使用 2 种方式在 Ndd 中执行 pinlang 脚本。

7.1.1 Ndd 执行当前编辑框的 Pin 脚本

使用 pinlang 语法在编辑框中写入脚本代码，然后在编辑框中，右键菜单-显示拼语言执行控制台，调出控制台。见下图所示：

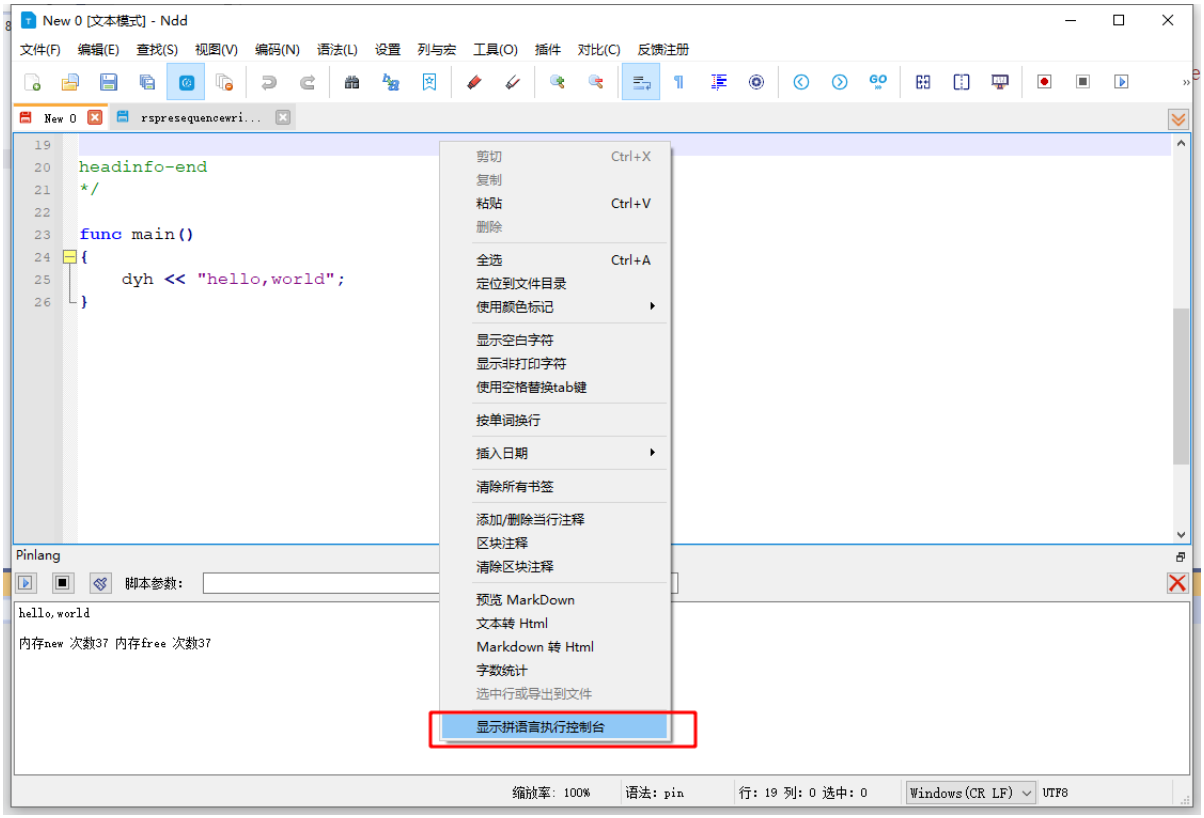


图 7.1: Pin 拼语言执行控制台

图中下半部分窗口，是执行当前 pin 脚本的窗口，有三个按钮，功能分别是：执行脚本、中断执行、清空日志。点击第一个按钮，可执行当前编辑框中的 Pin 脚本。图中例子，写了一个 main 函数，然后打印 helloworld。会把 Pin 的执行结果显示在控制台中。

脚本参数：如果该脚本有命令行参数，可在后面的输入框中填入参数字符串，控制台会自动读取参数列表，然后传递给脚本的 main 函数。关于 Pin 的语法，可以参考本手册的前面部分。

7.1.2 Ndd 执行 Pin 脚本插件

开发 Pin 语言的一个目的，就是为了在 Ndd 中编写插件。最开始 Ndd 的插件使用 cpp 编写，门槛过高，所以作者考虑引入一个简单的脚本语言，来帮助用户降低写脚本的门槛。依

次点击插件-pin Script，弹出如下插件界面:

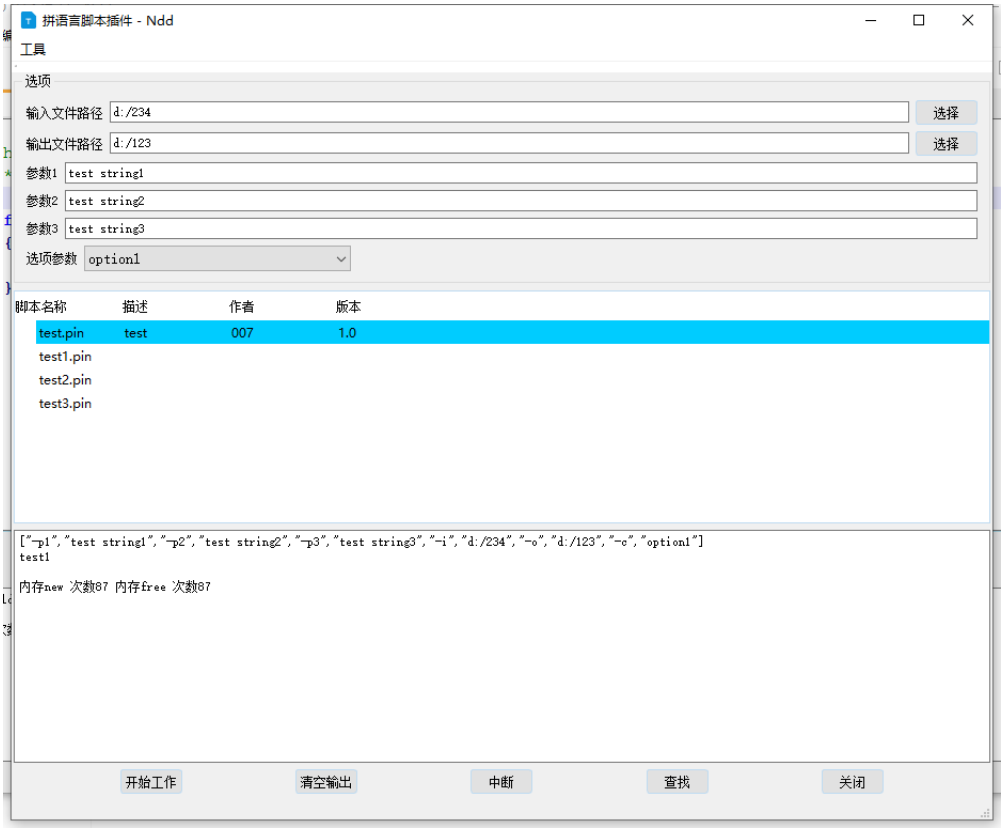


图 7.2: Pin 拼语言脚本插件

对该模块的使用说明：

界面中间有个列表，会列出当前安装的 Pin 脚本。点击上图菜单中的“工具-打开脚本目录”，会弹出当前 Ndd 的 Pin 脚本目录。可以从 Ndd 官网下载，或者拿到其他人写好的 Pin 插件脚本，手动加入到该目录下，然后重启插件，即可在列表中看到新加入的 Pin 脚本。脚本特别适合做一些字符串和文本相关的工作，供喜欢精确控制文本格式或有灵活需求的用户进一步做二次开发。

选项说明：是一个通用的传参格式，和脚本中的预定义字段一一对应。当点击下面的开始工作时，插件会自动把选项中的参数，传递到当前要执行脚本的入口函数。

下面详细介绍脚本头部的参数字段。

```
/*
headinfo-begin

[info]
ScriptName=test-for-ndd
Desc=test
Author=007
Version=1.0
Comment=这是一个例子。

[para]
need=1
inputFilePath=d:/234
outputFilePath=d:/123
stringPara1=test string1
stringPara2=test string2
stringPara3=test string3
comboxPara1=option1,option2,option3

headinfo-end
*/

func main(lb canshu)(zs)
{
  dyh << canshu;
  dyh << "test1";

  return 0;
}

程序执行结果如下:
["-p1","test string1","-p2","test string2","-p3","test string3","-i","d
   :/234","-o","d:/123","-c","option1"]
```

```
33 test1
```

脚本参数说明

在每个插件 Pin 脚本中，都必须要预先在头部留下脚本的参数特定字段，见上例代码行 1 到行 21 之间的字段信息，其本质是一个 ini 文件。其中 info 字段包含脚本的名称、描述、作者、版本和详细解释。这些字段会显示在插件脚本对应列表中的描述位置。

para 字段就是该脚本需要的参数，其中 need 表明该脚本是否需要参数，need 值如果 1 则表示需要参数；反之 need 值 0 则表示不需要参数。

comboxPara1 是一个下拉框中的可选列表选项，该字段是一个列表形式，多个参数字符串以逗号分隔开。

headinfo-begin 和 headinfo-end 是固定格式字段，务必保留。前面的/* 和后面的 */也是必须保留的，表示对 Pin 而言是注释，Pin 解析器将忽略执行头部参数信息。

当 need 值为 1 时，表示该脚本需要参数，在上图7.2 Pin 拼语言脚本插件 中，脚本头部中预留的参数，会以默认值参数的形式填充到插件界面的选项中。用户可以修改界面中的选项，然后把符合自己需要的参数修改好，脚本执行时插件框架会自动把选项中的参数传递下来。上例行 25 直接打印出参数列表，结果在行 32 所示。用户可以自行灵活获取参数后，根据参数来动态决定脚本要做的事情。

举例个例子，假设某个脚本提供了三种工作模式，则可以使用参数 comboxPara1，以列表的方式把选项暴露出来。脚本头部参数信息和代码例子如下：

```
1 /*
2 headinfo-begin
3
4 [info]
5 ScriptName=test-for-ndd
6 Desc=test
7 Author=007
8 Version=1.0
9 Comment=这是一个例子。
10
```

```
[para]
need=1
inputFilePath=
outputFilePath=
stringPara1=
stringPara2=
stringPara3=
comboxPara1=方式1,方式2,方式3

headinfo-end
*/

func main(lb canshu)(zs)
{
  dyh << canshu.size();

  zs index = canshu.indexOf('-c');

  if(index != -1 && (index+1) < canshu.size())
  {
    dyh << "当前选择工作方式是: " << canshu[index+1];
  }
  return 0;
}

程序执行结果如下:
2
当前选择工作方式是:  方式2
```

脚本带参数的例子

上例中脚本带一个选择列表参数，有三种选择方式，界面会显示如下：

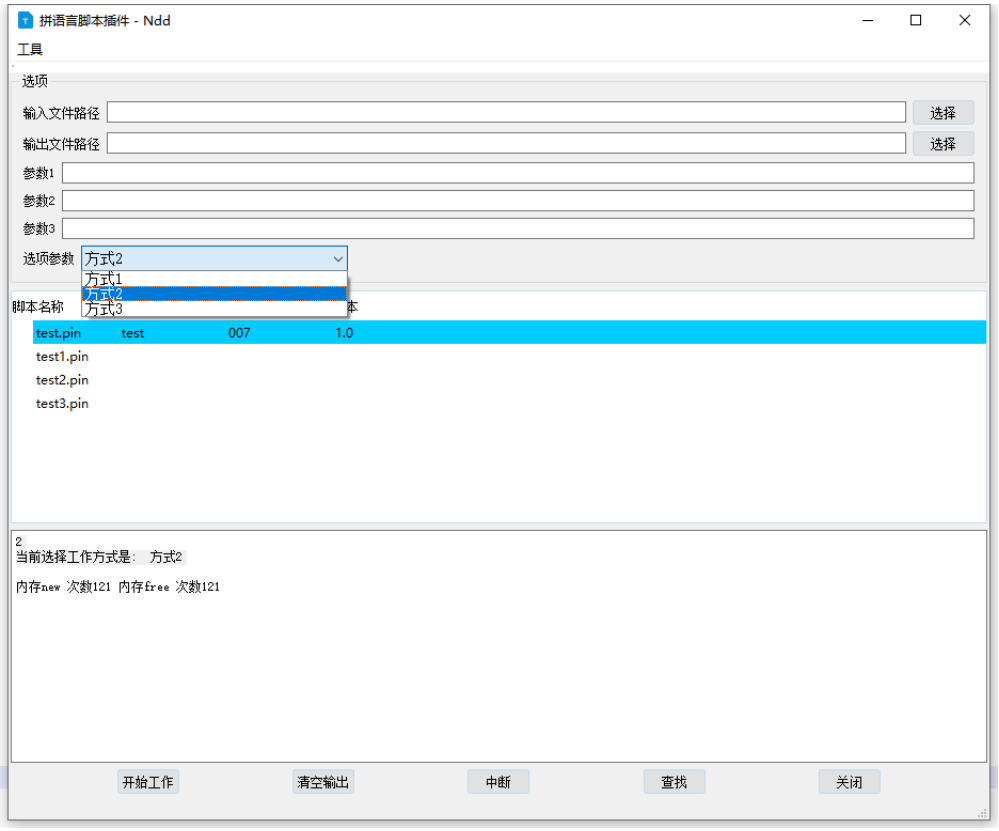


图 7.3: Pin 拼语言脚本提供三种工作方式

该脚本选择时，会默认提供选项参数，有三种方式供用户选择。用户可以在界面中动态选择当前执行脚本的工作方式。在上例代码行 31 中打印出用户选择的工作方式，打印执行结果在行 38。

下面详细说下选项参数字段和界面的对应关系。

界面参数	对应选项
界面输入文件路径	-i
界面输出文件路径	-d
界面参数 1	-p1
界面参数 2	-p2
界面参数 3	-p3
界面选项参数下拉框	-c

表 7.1: UI 参数和选项字段对应关系

所有的参数默认都是以字符串的形式表示。用户可灵活使用字符串的形式，动态传递参数到脚本中，以执行脚本中不同的功能。

当 need 值为 0 时，表示该脚本不需要参数，会自动忽略脚本中的参数，而且界面 UI 选项字段会留空。此时脚本头部只需要保持下面的 para 字段都是空即可，如下所示：

```
/*
headinfo-begin

[info]
ScriptName=test-for-ndd
Desc=test
Author=007
Version=1.0
Comment=这是一个例子。

[para]
need=0
inputFilePath=
outputFilePath=
stringPara1=
stringPara2=
stringPara3=
comboxPara1=
```

```
19
20 headinfo-end
21 */
```

脚本不需要参数的例子

如果不需要参数，则只需要保证头部按如上信息即可。脚本执行时，无需参数，插件也不会读取选项中的参数。

在实际插件开发中，可以灵活选择上面的参数配置。比如笔者要开发一个一键高亮 CPP 代码中注释的脚本插件，提供 2 个功能，功能 1 是高亮注释，功能 2 是删除注释，额外有个附加选择功能，就是可以把处理后的文本保存为一个新的文件。那么这个功能中，可以选择功能 1 和功能 2 做 comboxPara1 的参数选项。而文件输入有 2 种方式，其 1 是使用 inputFilePath 从外部输入，如果有该值，则处理该值的文件路径；反之如果没有该值，则处理当前编辑框中的内容。最终的输出文件则以 outputFilePath 参数来指定，如果有该参数，则保存处理后结果为该参数指定的文件；如果没该参数，则不保存新文件，或者直接修改当前编辑框内容。具体命令行的使用说明可以填写在 Comment 字段中。

7.2 Ndd 支持操作编辑器的 Pin 方法

下面介绍 Ndd 特有的内建方法函数。还是以例子入手，假设有这么一个需求，获取 Ndd 当前编辑框的句柄，然后读取当前编辑框中的内容，后续用户要对这些内容做一些字符串处理后，再回写到原始编辑框中，该如何实现？

下面给出代码如下。

```
1 hanshu hs_rukou()
2 {
3   //先获取当前编辑框的主窗口
4   PNddMain mainWin = ndd.getMainWin();
5
6
7   //再获取主窗口中当前的编辑框窗口
```

```
8   PEdit editor = mainWin.getCurEdit();
9
10  dy << editor.lineCount();
11
12  xun(zs i=0, s = editor.lineCount(); i<s; i++)
13  {
14    dy << editor.getLine(i);
15  }
16 }
```

使用编辑器 api 完成需求的例子

对例子的详细解释：

行 4 中使用 ndd 包全局方法 getMainWin 获取当前编辑器的主窗口，其类型是 PNddMain 。说明：ndd 目前可以打开多个窗口，但是只有第一个是主窗口。目前 Pin v0.6.3 阶段，只能在主窗口中运行 Pin 脚本，其余窗口暂时不放开。

行 8 主窗口中有多个编辑框标签，使用主窗口变量 mainWin, 调用函数 getCurEdit 获取当前编辑框的操作对象，其类型是 PEdit。

行 10 打印出当前编辑框中的行数量。

行 12 使用循环，遍历当前编辑框中的行，使用 getLine 函数获取每行内容，最后打印出来。

有了这么一个直观的全局例子，后续操作其它方法也是在这个架子上面加减 api 即可。

7.3 Ndd 内部方法函数

7.3.1 主程序窗口相关函数

说明：主程序窗口，即看到的整个 Ndd 的程序界面。编辑框、编辑框页面或编辑器 tab 标签，三个说法是一个概念，均是表示主编辑框窗口中的一个编辑框 tab 标签页面。为了不

混淆二者，后面使用 Ndd 主程序和编辑框两个概念来区分二者。目前主程序窗口可以有多个，但是可以理解为只有一个，因为只有第一个主程序窗口才能使用 pinlang 插件。

下面列出主程序窗口结构 PNddMain 相关的函数。

1. **ndd.getMainWin()(PNddMain)**

   返回值：类型 PNddMain，返回主程序窗口的操作句柄。

   作用：ndd 包全局方法，获取当前主程序窗口的操作句柄。

2. **getCurEdit()(PEdit)**

   返回值：返回主程序窗口中当前的编辑框句柄。

   作用：PNddMain 的内部方法，获取主程序窗口中当前的编辑框操作句柄。

3. **getEdit(zs index)(PEdit)**

   参数：index 需要获取编辑框页面的序号。

   返回值：类型 PEdit 结构，返回主程序窗口中指定序号的编辑框句柄。

   作用：PNddMain 的内部方法，获取主程序窗口中指定序号的编辑框句柄。比如主程序窗口中有 5 个编辑框页面，则序号是 0-4，可指定序号后返回指定的编辑框页面。该函数和 getCurEdit 功能类似。

4. **newEditor()**

   作用：在主程序窗口中新建一个编辑框标签页面。没有返回值，后续可通过 getEdit 或 getCurEdit 获取其操作句柄。注意：调用该函数后，主程序窗口中当前编辑框会自动设置为新建的编辑框页面。

7.3.2 编辑框页面相关函数

说明：编辑框页面，或者编辑框，或编辑框标签页，三个说法是同样概念，均是表示主程序窗口中的一个编辑框。

特别备注：编辑器中的所有字符串，都是使用 utf8 编码。即不管原始文件是什么字符编码，通过编辑框打开后，编辑框内部显示的文字都是 utf8 编码，后续下面一切操作的 api 函数都是以 utf8 编码来表示。

下面列出编辑框结构 PEdit 的操作方法。

1. **getText()(zfc)**

   返回值：编辑框页面中的所有文字内容，是一个字符串。再啰嗦一遍，该字符串是以 utf8 字符编码来表示。

   获取该字符串内容后，用户可自由做二次处理。

2. **setText(zfc text)**

   参数：text 给编辑框设置的字符文本内容。

   返回值：无

   作用：和 getText 相对。使用字符串 text 的内容，填充编辑框。要注意默认 text 是 utf8 编码，如果用户有手动修改 text 的字符编码，则使用该函数时，务必保证 text 是 utf8 编码。这个函数不支持撤销，填充时会把编辑框中原有的旧内容清空，然后使用新字符填入编辑框，操作是不能撤销的。

3. **getTextRange(zs start, zs len)(zfc)**

   作用：获取编辑框中从 start 位置开始，长度为 len 字节的字符串内容，返回值是该获取的字符串。注意该内容本质是 utf8 编码字节流。

4. **getLine(zs num)(zfc)**

   参数：num 整数，待获取的行号码。行号是从 0 开始的。即如果要获取编辑框中第一行的内容，则需要使用 num 0。

   返回值：字符串，编辑框中第 num 行的文本内容，包含行尾换行符。如果是空号，则返回空字符串。

5. **lineCount()(zs)**

   作用：返回编辑框中的行数量。

6. **setSel(zs startpos, zs endpos)**

   参数 1：startpos 开始高亮的位置，单位是字节。请注意开始位置，会被认定为锚点。在鼠标点击选择文字的开始，鼠标点下位置会被认定为锚点，鼠标选择文字结束后释放键起来的位置被认定为光标位置。startpos 对应的位置是锚点位置。

   参数 2：endpos 高亮的结束位置 (不包含该位置本身)，单位是字节。endpos 对应的位置是光标位置。

   作用：选择编辑框中从 startpos 开始，到 endpos 结束字节的字符内容。选择的内容会被高亮显示。

   说明：如果文档中有中文，这个函数很可能高亮效果无效。这是因为中文的 utf8 是多字节，用户如果随意给 startpos，或 endpos，则很可能位置是在中文字符的中间。此时会导致失败。

   示例：假设文本内容为”你好”（UTF-8 编码占 6 字节，“你”字三个字节，“好”字三个字节），若想选择第 1 个字符，对应的字节位置为 0-3；第二个字符对应字节位置是 3-6。如果给定位置是 (0,3) 则会高亮“你”字；如果不小心给了（0,4）则会高亮失败，因为 4 正好在“好”字的中间，这个位置是非法的。

   解决办法有两种： 1 自己编程精确控制给定位置。2 使用 posBefore 或 posAfter 函数来做矫正，分别返回位置前一个字符的开始位置，和位置下一个字符的结束位置。这两个函数会把给定的 pos 值，按照字符编码的字符切割位置对齐。比如上例位置 4 是非法的，使用 posBefore(4) 或 posAfter(4) 会得到 3 和 6，即避免给定字节位置在字符的中间。

7. **gotoPos(zs pos)**

   作用：移动编辑器的光标到字节位置 pos。如果 pos 在汉字中间，则会做自动调整到字符后面的位置，因为光标只能出现在字符位置，而不能出现在字符中间的字节位置。该函数会取消一切选择，把锚点和光标位置都设置在 pos 字节处。另外会滚动编辑器滚动视图，让 pos 位置呈现在视图区域。

8. **gotoLine(zs line)**

   作用：移动编辑器的光标到指定行首位置，该行会被认定为当前行。该函数会取消一切选择，把锚点和光标位置都设置在 line 行首处。另外会滚动编辑器滚动视图，让行 Line 位置呈现在视图区域。

9. **setCurPos(zs pos)**

   作用：移动编辑器的光标到指定位置。锚点和光标之间的内容，会被选中高亮。该函数不会自动滚动视图，位置可能不在视图中。

10. **setAnchor(zs pos)**

    作用：设置编辑器的锚点到指定位置 pos。锚点和光标之间的内容，会被选中高亮。该函数不会自动滚动视图，位置可能不在视图中。

11. **getCurPos()(zs)**

    作用：获取编辑器的光标位置并返回。

12. **getAnchor()(zs)**

    作用：获取编辑器的锚点位置并返回。

13. **selectAll()**

    作用：全选编辑框中内容。

14. **posBefore(zs pos)**

    作用：返回位置 pos 的上一个字符的开始位置。在有汉字等多字节编码字符时，可以使用该函数把位置 pos 对齐到汉字开始的边界字节处，防止出现字节位置在多字节字符的中间非法位置。如果 pos 已经在某个汉字的开头位置，则会返回前一个字符的开头位置；反之如果 pos 正好在某个汉字的中间字节，则会返回当前汉字的开头位置。

15. **posAfter(zs pos)**

    作用：返回位置 pos 的下一个字符的结束位置。在有汉字等多字节编码字符时，可以使用该函数把位置 pos 对齐到汉字编码结束的边界字节处，防止出现字节位置在多字节字符的中间非法位置。如果 pos 已经在某个汉字的编码结束位置，则会返回下一个字符的编码结束位置；反之如果 pos 正好在某个汉字的中间字节，则会返回当前汉字的编码结束位置。

16. **setSelStart(zs pos)**

    作用: 给定开始选择的位置。

17. **setSelEnd(zs pos)**

    作用: 给定结束选择的位置。setSelStart 和 setSelEnd 效果同 setSel。

18. **setEmptySel(zs pos)**

    作用：取消当前选择，同时把光标和锚点移动到 pos 位置。

19. **lineFromPos(zs pos)(zs)**

    作用：返回位置 pos 所在的行号。说明：编辑器行号是从 0 开始的，而编辑框行号显示是从 1 开始。

20. **posFromLine(zs line)(zs)**

    作用：返回行 line 行首位置的 pos。

21. **getLineEndPos(zs line)(zs)**

    作用：返回行 line 行尾位置的 pos。

22. **lineLength(zs line)(zs)**

    作用：返回行 line 的长度，目前是按 utf8 的字节长度。

23. **getSelText()(zs)**

    作用：返回编辑框中当前选择的字符串。

24. **removeSelText()**

    作用：删除编辑框中当前选择的字符串内容。注意该函数是操作编辑框，把选择的文字删除。

25. **replaceSelText(zfc text)**

    作用：使用字符串 text 替换编辑框中当前选择的内容。

26. **deleteRange(zs start, zs len)**

    作用：从 start 位置开始，删除 len 长字节的内容。要注意不能在汉字中间位置进行删除，否则会导致乱码。

27. **insertAtPos(zs pos, zfc text)**

    作用：在编辑框中的 pos 位置插入字符串 text。要注意不能在汉字中间位置进行插入，否则会导致乱码。

28. **highLight(zs startpos, zs len, zs colorid)**

    参数 1：startpos 高亮的开始位置

    参数 2：len 高亮的字节长度

    参数 3：colorid 高亮的颜色，目前只能 0-4，是 Ndd 中预设置的 5 种高亮颜色。

    作用：在编辑器中从 startpos 位置开始，使用颜色 colorid 高亮 len 长的字节字符串。

29. **clearHighLight(zs startpos, zs len, zs colorid)**

    作用：取消高亮，是 highLight 的反作用函数。注意颜色 colorid 要和高亮时一样，否则无效。

30. **clearAllHighLight()**

    作用：取消编辑框中的一切高亮。注意：该函数只对 highLight 函数的结果起反高亮作用，对编辑器中手动高亮的内容作用无效。

31. **beginUndo()**

    作用：以上函数如果是操作编辑框，则均支持在编辑器中撤销操作。每个函数操作后的效果均可以撤销。如果想把多个操作的效果，一并撤销操作。比如删除了 5 次字符串，希望这 5 次的撤销效果通过一次撤销即可完成，则需把这 5 次操作包含在函数 beginUndo 和 endUndo 之间。

32. **endUndo()**

    作用：见 beginUndo。注意该函数必须和 beginUndo 成对出现。被 beginUndo 和 endUndo 包围的操作代码，其可以在编辑器中撤销一次即全部撤销效果。

33. **replace(zfc src, zfc dst, zs mode, zs cs)(zs)**

    参数 1：src 待查找替换的字符串

    参数 2：dst 替换的目标字符串

参数 3：mode 替换模式。0 普通模式；1 扩展模式; 2 正则模式。

参数 4：cs 可选参数，是否区分大小写。0 不区分 1 区分大小写。默认为 1。

返回值：zs 返回替换的次数。

作用：使用字符串 dst 替换编辑框中的 src 字符串。支持正则或扩展模式，可选择区分大小写。