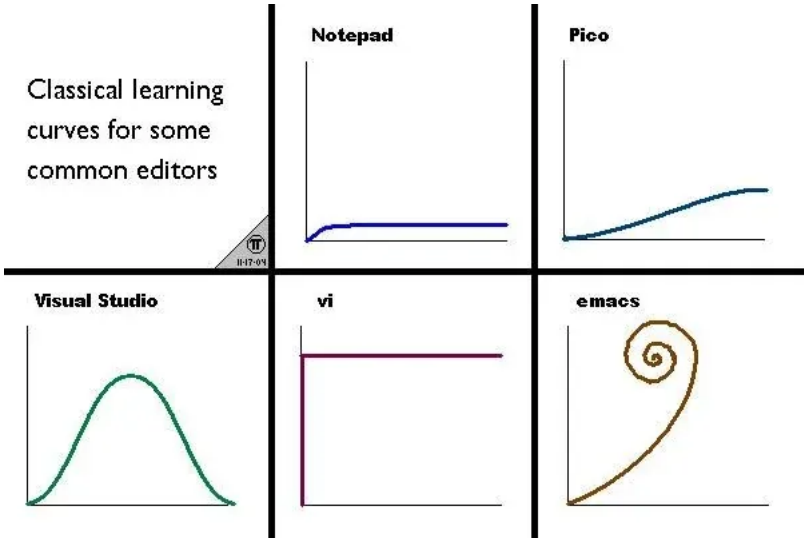
Linux基础应用

Vim 编辑器

Vim 是一种功能强大且灵活的文本编辑器，常用于 Linux 和 Unix 系统上，Vim 默认存在于大部分 Linux 系统中。作为 Vi 编辑器的增强版本，Vim 支持语法高亮、自动补全、多级撤销等多种现代文本编辑功能。它被广泛用于编写代码、编辑配置文件、处理文档等任务。虽然 Vim 初学者可能会觉得学习曲线较高，但一旦熟练掌握，它可以极大地提升工作效率。



vim 格式&参数

命令格式：

```
vim  [options]  [file]
```

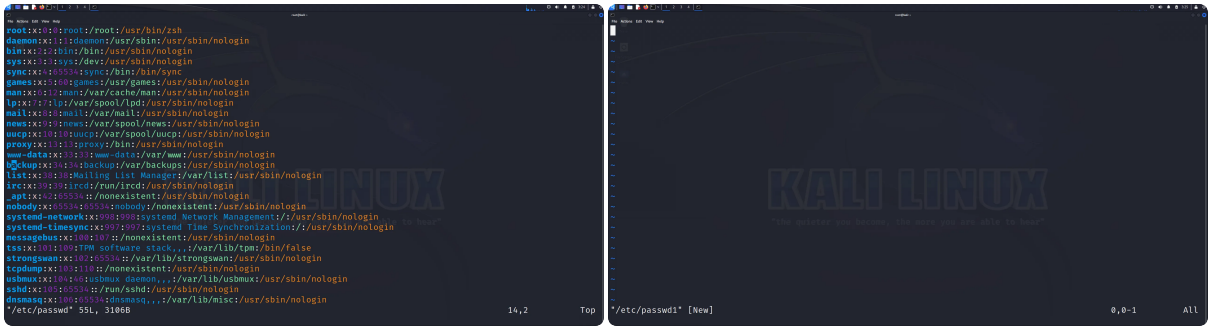
主要参数：

- –c ： 读取文件后会执行 –c 之后的指令；
- –R ： 以只读方式打开，但是可以强制保存；
- –M ： 以只读方式打开，不可以强制保存；
- **–r ： 恢复崩溃的会话；**
- +num ： 从第 num 行开始。

打开文件

如果 filename 文件存在，则会打开文件并显示文件内容；

如果 filename 文件不存在，vim 会在下面提示 [New]，并且会在第一次保存时创建该文件（不保存时会产生备份文件）。



Vim的工作模式

Vim具有三种基本模式，分别用于不同的操作：

普通模式：打开文件后默认进入的模式，用于浏览和选择命令。

插入模式：编辑模式，可以输入文本。

命令模式：用于执行保存、退出、搜索等命令。

按 Esc 键退回到普通模式。

普通模式

vim 普通模式相关命令：

- **移动光标：　　k、j、h、l 键（方向键）；**
- cc：　　修改，即删除当前行并进入插入模式；
- **dd：　　删除当前行，num dd 删除光标所在行及其以下的 num 行；**
- **yy：　　复制当前行，num yy 复制光标所在行及其以下的 num 行；**
- **p、P：　　复制的内容作粘贴（p 粘贴在当前行下面，P粘贴在当前行上面）；**
- **u、U：　　撤销最近的操作（u 最近一次，U 最近操作的那行）；**
- **/keyword：　　搜索关键字；**
- **n、N：　　搜索关键字的时候继续寻找下一个（n 向下找，N 向上找）；**
- **G：　　光标跳到最后一行，num G 光标跳到 num 行；**
- **gg：　　光标跳到第一行。**
- **i：　　进入插入模式，在当前光标处插入文本。**

插入模式

在插入模式下可以直接输入文本。按 `Esc` 键可返回普通模式。

- a：　光标会跳到当前后一格位置；
- A：　光标会停在当前行的最后位置；
- I：　光标会停在当前行的最前面的位置；
- o：　会在光标所在行的下面新增一行空白；
- O：　会在光标所在行的上面新增一行空白；
- r：　替换光标所在位置的字符，但不进入插入模式；
- R：　从光标所在位置开始替换，能替换多个字符。

命令模式

按下 `:` 进入命令模式 ， 一些常用的命令如下：

保存/退出编辑内容：

- :w　　保存；
- **:q　　退出；**
- **:wq　　保存及退出；**
- :w!　　强制保存；

- :q!　　**强制离开；**
- **:wq!**　　**强制保存离开。**

显示/隐藏行号：

- **:set nu**
- :set nonu

查找文件内容：

- **/word**　　在光标之后查找一个字符串
- ?word　　在光标之前查找一个字符串

替换文件内容：

- 1,5s/word1/word2/g　　将文档中 1–5 行的 word1 替换为 word2，不加 g 则只替换每行的第一个 word1。
- %s/word1/word2/gi　　将文档中所有的 word1 替换为 word2，不区分大小写。

Nano 编辑器

Nano 是一款简单易用的文本编辑器，广泛应用于 Linux 和 Unix 系统中。与 Vim 相比，Nano 更加适合初学者，使用直观、操作简单，主要通过快捷键完成各项操作，适合快速编辑文件而无需掌握复杂的命令。Nano 预装在很多 Linux 发行版上，因其轻量和易用，成为了 Linux 用户常用的文本编辑工具之一

nano 的一切操作都很简单。要在命令行下使用 nano 打开文件，可以像下面这样做：

```
1   nano /tmp/demo
```

如果启动 nano 的时候没有指定文件名，或者指定的文件不存在，则 nano 会开辟一段新的缓冲区进行编辑。如果你在命令行中指定了一个已有的文件，则 nano 会将该文件的全部内容读入缓冲区，以备编辑。

> 在 nano 编辑器窗口的底部显示了各种命令以及简要的描述。这些都是 nano 的控制命令。脱字符（^）表示 Ctrl 键。因此，^X 代表组合键 Ctrl+X。

Nano基本操作

在 Nano 编辑器中，操作命令通常通过快捷键完成。以下是一些常用的快捷键：

- **保存文件**：`Ctrl + O`（Write Out）。输入后会显示保存路径，按回车键确认。
- **退出编辑器**：`Ctrl + X`。若文件有更改，Nano 会询问是否保存。
- **剪切整行**：`Ctrl + K`。可以用来快速剪切当前行。
- **粘贴剪切内容**：`Ctrl + U`。将剪切的内容粘贴到当前位置。
- **搜索**：`Ctrl + W`。输入要查找的关键词后按回车进行搜索。
- **行首/行尾移动**：`Ctrl + A` 移动到行首，`Ctrl + E` 移动到行尾。
- **撤销**：`Ctrl + _`。用于撤销之前的更改。

Linux特殊符号

操作符

Linux操作符 能够帮助用户灵活地控制命令的输入输出、组合命令、实现逻辑判断和条件执行等操作。这些符号使得 Linux 命令行具备了更强的功能，适合用于自动化脚本、批量文件处理等复杂任务。

`>`（重定向输出）

`>` 符号用于将命令的标准输出（stdout）重定向到文件中。如果指定的文件不存在，系统会创建一个新的文件；如果文件已存在，则覆盖其内容。适合用于输出记录或将输出保存到文件中。

```
echo "Hello, World!" > output.txt  # 将输出保存到 output.txt 文件中
```

`>>`（追加输出）

与 `>` 类似，但 `>>` 符号会将命令输出追加到文件末尾，而不会覆盖已有内容。适合需要将多个命令的输出累积记录在同一文件中的情况。

```
Plain Text
1  echo "Another line" >> output.txt  # 将输出追加到 output.txt 文件末尾
```

`<`（重定向输入）

`>` 和 `>>` 用于将输出重定向到文件，而 `<` 则用于将文件内容作为命令的输入。此符号用于读取文件的内容并将其传递给命令，通常与 `cat`、`grep` 等命令结合使用。

```
cat < input.txt  # 使用 input.txt 作为 cat 命令的输入
```

`|`（管道符）

`|` 符号用于将一个命令的输出直接作为另一个命令的输入，允许用户将多个命令组合起来处理数据。例如，将 `ls -l` 的输出直接传递给 `grep` 以过滤结果。

```
ls -l | grep ".txt"  # 列出当前目录下所有 .txt 文件
```

`&`（后台运行）

`&` 符号用于将命令放入后台运行，使用户可以继续在前台执行其他命令。这在长时间运行的任务中非常有用，例如启动一个服务器进程或执行一个耗时的计算任务。

```
long_running_command &  # 后台运行该命令
```

`&&`（逻辑与）

`&&` 是条件执行符，用于在前一个命令成功执行（返回状态码 0）后才执行下一个命令。如果前一个命令失败，则跳过后续命令。例如，可用于确保某项操作在成功完成后再执行下一步。

```
mkdir new_folder && cd new_folder  # 如果创建目录成功，则进入该目录
```

`||`（逻辑或）

`||` 也是条件执行符，但它会在前一个命令失败时（返回状态码非 0）执行下一个命令。常用于容错处理，如在文件不存在时输出提示信息。

```
ls nonexistent_file || echo "File not found"  # 文件不存在时输出提示
```

`;`（命令分隔符）

`;` 允许用户在同一行内输入多个命令，系统会按顺序执行这些命令，彼此之间没有条件关系。这有助于节省时间，特别是当多条命令需要按顺序执行时。

```
Plain Text
cd /home; ls; pwd  # 进入 /home 目录，列出文件并显示当前路径
```

文件描述符

Linux 系统会将每个对象当作文件来处理，这包括输入和输出。Linux 用文件描述符来标识每个文件对象。文件描述符是一个非负整数，唯一会标识的是会话中打开的文件。每个进程一次最多可以打开 9 个文件描述符。出于特殊目的，bash shell 保留了前 3 个文件描述符（0、1 和 2）：

文件描述符	缩写	描述	硬件
0	STDIN	标准输入	键盘
1	STDOUT	标准输出	显示器
2	STDERR	标准错误	显示器

重定向标准输出到文件：

```
ls > output.txt
```

这里的 `>` 实际上是 `1>` 的简写，等价于 `ls 1> output.txt` 。

重定向标准错误到文件：

```
ls nonexistentfile 2> error.txt
```

将错误信息（文件描述符 `2` ）重定向到 `error.txt` 。

同时重定向标准输出和标准错误：

```
ls file.txt nonexistentfile > output.txt 2>&1
```

这里的 `2>&1` 表示将标准错误重定向到标准输出， `> output.txt` 则将标准输出（和标准错误）都重定向到 `output.txt` 。

文本处理

Linux中有一系列强大的文本处理命令，可以帮助用户高效地操作、编辑、分析文本文件。这些命令适用于各种情况，从简单的查看文件内容，到复杂的文本处理和数据提取。在我们进行渗透测试的过程中，使用这些命令能大大地提高我们的工作效率，帮助我们更好地实现自动化。

`grep` – 查找匹配文本

`grep` 命令是一个强大的文本搜索工具，用于查找文件中的特定模式（字符串）。它会返回所有包含该模式的行。`grep` 适合用于日志分析或提取文件中符合特定条件的数据。

```
grep "pattern" file.txt
```

支持正则表达式。

```
grep -E "[0-9]+" file.txt
```

`sed` – 文本替换和流编辑

`sed` 是一个流编辑器，常用于在流中对文本进行操作。它可以执行诸如查找并替换、插入、删除文本等任务。`sed` 是处理文本文件时非常有用的工具，尤其是在批量处理文本文件时。

Plain Text

```
sed 's/old_text/new_text/g' file.txt
```

该命令将 `file.txt` 中所有的 `old_text` 替换为 `new_text`。`g` 表示全局替换（即替换文件中的所有匹配项）。

`awk` – 基于模式处理文本

`awk` 是一种功能强大的编程语言，专门用于处理和分析文本数据，尤其适用于按列处理表格数据。`awk` 可以自动将文本行分割为字段，默认的分隔符是 **空格** 或 **制表符**（Tab），允许根据字段进行操作。

Plain Text

```
awk '{print $1, $3}' file.txt
```

该命令会打印 `file.txt` 文件的每一行中的第一列和第三列。

`cut` – 按列切割文本

`cut` 用于根据分隔符切割文本文件。它非常适合从表格格式的数据中提取某些列。`cut` 支持按字符位置或分隔符（如空格或逗号）来提取数据。

示例：

```
Plain Text
cut -d ' ' -f 1,3 file.txt
```

该命令会提取 `file.txt` 文件中由空格分隔的第一列和第三列。

用法扩展： 如果数据是按逗号分隔的（例如 CSV 文件），可以使用 `-d` 参数指定分隔符：

```
Plain Text
cut -d ',' -f 2,4 file.csv
```

`sort` – 排序文本行

`sort` 用于对文件中的文本行进行排序。它可以根据字母、数字、日期等多种方式对文件进行排序。排序结果默认按升序排列。

示例：

```
sort file.txt
```

该命令会对 `file.txt` 文件中的文本行按字母顺序进行排序。

`uniq` – 去重

`uniq` 用于删除重复的行，它只能删除连续的重复行，因此常与 `sort` 一起使用，先对文件进行排序，再使用 `uniq` 去除重复行。

```
sort file.txt | uniq
```

该命令会删除 `file.txt` 中重复的行，显示唯一的行。

`head` – 显示文件开头部分

`head` 用于显示文件的前几行。它非常适用于查看文件的开头内容，或者快速检查大型文件的部分内容。

```
Plain Text
1  head -n 20 file.txt
```

该命令会显示 `file.txt` 的前 20 行。

`tail` – 显示文件末尾部分

`tail` 用于显示文件的最后几行。这个命令非常适合查看日志文件的最新内容，通常与 `-f` 参数结合使用，实时监控文件更新。

示例：

```
Plain Text
1  tail -n 50 file.txt
```

该命令会显示 `file.txt` 的最后 50 行。

```
Plain Text
1  tail -f /var/log/syslog
```

`tr` – 替换字符

`tr` 用于字符替换、删除或转换。它常用于转换字符集、大小写转换等任务。

```
Plain Text
1  echo "hello world" | tr 'a-z' 'A-Z'
```

该命令会将字符串 "hello world" 中的小写字母转换为大写字母，输出 "HELLO WORLD"。

`wc` – 统计文件内容

`wc` 用于统计文件的行数、字数和字符数。它非常适用于检查文本文件的大小。

```
Plain Text
1  wc -l file.txt
```

该命令会返回 `file.txt` 中的行数。

`paste` – 合并文件

`paste` 用于将多个文件的内容按列合并。它非常适用于将多个文件的内容并排显示。

```
Plain Text
1  paste file1.txt file2.txt
```

该命令会将 `file1.txt` 和 `file2.txt` 文件的内容合并成两列输出。

`tee` – 输出重定向到文件和终端

`tee` 用于将命令输出同时发送到终端和文件。它常用于需要查看输出并同时保存结果的情况。

```
Plain Text
1  echo "hello world" | tee output.txt
```

该命令会将 "hello world" 输出到终端，并将其保存到 `output.txt` 文件中。

具体应用

敏感信息

在渗透测试过程中，查找特定类型的文件或敏感文件是常见任务，`find` 命令也可以帮助我们寻找flag。`find` 命令可以根据文件名、权限、大小等条件进行文件搜索。

```
1  find / -name "*.bak" 2>/dev/null
```

这个命令会查找整个系统中所有扩展名为 `.bak` 的文件，并将错误输出（如权限不足的目录）重定向到 `null`，避免显示不必要的错误信息。

获取目标账号

查看当前系统有哪些账号。

```
awk -F ':' '{ print $1}' /etc/passwd
```

日志分析

统计每个 IP 地址的访问次数。

```
awk '{print $1}' access.log | sort | uniq -c
```

制作用户名字典

从Web响应中提取邮箱用户名作为字典。

```
curl http://example.com/index.html | grep -oE '[a-z]+@example.net' | sed
's/@example\.net//' > user.txt
```

拓展

Linux命令闯关挑战：cmdchallenge.com

正则表达式：https://help.aliyun.com/zh/sls/user-guide/getting-started-with-regular-expressions