目录

说明

1. linux目录记录个人学习linux时用到的一些命令
2. notes目录记录个人感兴趣的一些笔记

dpkg命令

dpkg命令的英文全称是“Debian package”，dpkg是Debian的一个底层包管理工具，主要用于对已下载到本地和已安装的软件包进行管理。

dpkg这个机制最早由Debian Linux社区所开发出来的，通过dpkg的机制，Debian提供的软件就能够简单的安装起来，同时能提供安装后的软件信息，实在非常不错。只要派生于Debian的其它Linux distributions大多使用dpkg这个机制来管理，包括B2D，Ubuntu等。

dpkg软件包相关文件介绍

`/etc/dpkg/dpkg.cfg` dpkg包管理软件的配置文件【Configuration file with default options】

`/var/log/dpkg.log` dpkg包管理软件的日志文件【Default log file (see /etc/dpkg/dpkg.cfg(5) and option --log)】

`/var/lib/dpkg/available` 存放系统所有安装过的软件包信息【List of available packages.】

`/var/lib/dpkg/status` 存放系统现在所有安装软件的状态信息

`/var/lib/dpkg/info` 记安装软件包控制目录的控制信息文件

参数	说明
-i	安装软件包
-r	删除软件包
-l	显示已安装软件包列表
-L	显示于软件包关联的文件
-c	显示软件包内文件列表

参考实例

安装包：

```
dpkg -i package.deb
```

删除包：

```
dpkg -r package.deb
```

`dpkg -r package-name` # --remove， 移除软件包，但保留其配置文件

`dpkg -P package-name` # --purge， 清除软件包的所有文件（removes everything, including conffiles）

列出当前已安装的包：

```
dpkg -l
```

每条记录对应一个软件包，注意每条记录的第一、二、三个字符，这就是软件包的状态标识，后边依此是软件包名称、版本号和简单描述。

1. 第一字符为期望值(Desired=Unknown/Install/Remove/Purge/Hold)，它包括：
   u Unknown状态未知,这意味着软件包未安装,并且用户也未发出安装请求.
   i Install用户请求安装软件包.
   r Remove用户请求卸载软件包.
   p Purge用户请求清除软件包.
   h Hold用户请求保持软件包版本锁定.

2. 第二列,是软件包的当前状态(Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend)
   n Not软件包未安装.
   i Inst软件包安装并完成配置.
   c Conf-files软件包以前安装过,现在删除了,但是它的配置文件还留在系统中.
   u Unpacked软件包被解包,但还未配置.
   f halF-conf试图配置软件包,但是失败了.
   h Half-inst软件包安装,但是但是没有成功.
   w trig-aWait触发器等待
   t Trig-pend触发器未决

3. 第三列标识错误状态,第一种状态标识没有问题,为空. 其它符号则标识相应问题（Err?=(none)/Reinst-required (Status,Err: uppercase=bad)）
   h 软件包被强制保持,因为有其它软件包依赖需求,无法升级.
   r Reinst-required，软件包被破坏,可能需要重新安装才能正常使用(包括删除).
   x 软包件被破坏,并且被强制保持.

4. 案例说明： ii —— 表示系统正常安装了该软件
   pn —— 表示安装了该软件，后来又清除了
   un —— 表示从未安装过该软件
   iu —— 表示安装了该软件，但是未配置
   rc —— 该软件已被删除，但配置文件仍在

5. dpkg子命令 为了方便用户使用，dpkg不仅提供了大量的参数选项, 同时也提供了许多子命令。比如：
   `dpkg-deb、dpkg-divert、dpkg-query、dpkg-split、dpkg-statoverride、start-stop-daemon`

列出deb包的内容：

```
dpkg -c package.deb
```

配置

```
dpkg --configure package
```

查询

`dpkg -l package-name-pattern` # --list, 查看系统中软件包名符合pattern模式的软件包

`dpkg -L package-name` # --listfiles, 查看package-name对应的软件包安装的文件及目录

`dpkg -p package-name` # --print-avail, 显示包的具体信息

`dpkg -s package-name` # --status, 查看package-name（已安装）对应的软件包信息

`dpkg -S filename-search-pattern` # --search, 从已经安装的软件包中查找包含filename的软件包名称

更多dpkg的使用方法可在命令行里使用 `man dpkg` 来查阅 或直接使用 `dpkg --help` 。

useradd命令详解

原文链接

语法

```
useradd [-mMnr][-c <备注>][-d <登入目录>][-e <有效期限>][-f <缓冲天数>][-g <群组>][-G <群
```

或

```
useradd -D [-b][-e <有效期限>][-f <缓冲天数>][-g <群组>][-G <群组>][-s <shell>]
```

参数说明

- -c<备注>　加上备注文字。备注文字会保存在passwd的备注栏位中。
- -d<登入目录>　指定用户登入时的起始目录。
- -D　变更预设值.
- -e<有效期限>　指定帐号的有效期限。
- -f<缓冲天数>　指定在密码过期后多少天即关闭该帐号。
- -g<群组>　指定用户所属的群组。
- -G<群组>　指定用户所属的附加群组。
- -m　自动建立用户的登入目录。
- -M　不要自动建立用户的登入目录。
- -n　取消建立以用户名称为名的群组.
- -r　建立系统帐号。
- -s　　指定用户登入后所使用的shell。
- -u　指定用户ID。

实例

添加一般用户

```
useradd tt
```

为添加的用户指定相应的用户组

```
useradd -g root tt
```

创建一个系统用户

```
useradd -r tt
```

为新添加的用户指定home目录

```
useradd -d /home/myd tt
```

建立用户且制定ID

```
useradd caojh -u 544
```

禁止用户wsj0051登录

```
 usermod -s /sbin/nologin wsj0051
```

恢复用户wsj0051登录

```
usermod -s /bin/bash wsj0051
```

其他

```
#添加一个不能登录的用户
useradd -d /usr/local/apache -g apache -s /bin/false apache
```

要拒绝系统用户登录，可以将其 shell 设置为 /usr/sbin/nologin 或者 /bin/false。

```
usermod -s | --shell /usr/sbin/nologin username
```

或者

```
usermod -s | -shell /bin/false username
```

说明及比较：

```
/bin/false
```

/bin/false 什么也不做只是返回一个错误状态，然后立即退出。将用户的 shell 设置为 /bin/false，用户会无法登录，并且不会有任何提示。

```
/usr/sbin/nologin
```

nologin 会礼貌的向用户显示一条信息，并拒绝用户登录：

```
This account is currently not available.
```

有一些软件，比如一些 ftp 服务器软件，对于本地非虚拟账户，只有用户有有效的 shell 才能使用 ftp 服务。这时候就可以使用 nologin 使用户即不能登录系统，还能使用一些系统服务，比如 ftp 服务。/bin/false 则不行，这是二者的重要区别之一。

```
/etc/nologin
```

如果存在 /etc/nologin 文件，则系统只允许 root 用户登录，其他用户全部被拒绝登录，并向他们显示 /etc/nologin 文件的内容。

图 - 笔记

cat命令

参数说明

- -n 或 --number：由 1 开始对所有输出的行数编号。
- -b 或 --number-nonblank：和 -n 相似，只不过对于空白行不编号。
- -s 或 --squeeze-blank：当遇到有连续两行以上的空白行，就代换为一行的空白行。
- -v 或 --show-nonprinting：使用 ^ 和 M- 符号，除了 LFD 和 TAB 之外。
- -E 或 --show-ends : 在每行结束处显示 $。
- -T 或 --show-tabs: 将 TAB 字符显示为 ^I。
- -A, --show-all：等价于 -vET。
- -e：等价于"-vE"选项；
- -t：等价于"-vT"选项；

cat命令是linux下的一个文本输出命令，通常是用于观看某个文件的内容的；

cat主要有三大功能

1. 一次显示整个文件。

```
$ cat   filename
```

2. 从键盘创建一个文件。只能创建新文件,不能编辑已有文件.

```
$ cat  >  filename
```

`cat > test.sh << EOF` 以EOF作为输入结束，之前存在的内容会被覆盖掉。
`cat << EOF >> test.sh` 将内容追加到 test.sh 的后面，不会覆盖掉原有的内容
EOF只是标识，不是固定的

```
cat << HHH > iii.txt
```

输入内容：

```
> sdlkfjksl
> sdkjflk
> asdlfj
> HHH
```

这里的“HHH”就代替了“EOF”的功能。结果是相同的。 引用 `cat iii.txt` 可以看到结果：

```
sdlkfjksl
sdkjflk
asdlfj
```

3. 将几个文件合并为一个文件。 将file1 file合并到file

```
$ cat   file1   file2  > file
```

把 textfile1 的文档内容加上行号后输入 textfile2 这个文档里：

```
cat -n textfile1 > textfile2
```

把 textfile1 和 textfile2 的文档内容加上行号（空白行不加）之后将内容附加到 textfile3 文档里：

```
cat -b textfile1 textfile2 >> textfile3
```

实例

清空 /etc/test.txt 文档内容：

```
cat /dev/null > /etc/test.txt
```

cat 也可以用来制作镜像文件。例如要制作软盘的镜像文件，将软盘放好后输入：

```
cat /dev/fd0 > OUTFILE
```

相反的，如果想把 image file 写到软盘，输入：

```
cat IMG_FILE > /dev/fd0
```

注：

1. OUTFILE 指输出的镜像文件名。
2. IMG_FILE 指镜像文件。
3. 若从镜像文件写回 device 时，device 容量需与相当。
4. 通常用制作开机磁片。

sed命令

sed是一种流编辑器，它是文本处理中非常好的工具，能够完美的配合正则表达式使用，功能不同凡响。 处理时，把当前处理的行存储在临时缓冲区中，称为“模式空间”（pattern space），接着用sed命令处 理缓冲区中的内容，处理完成后，把缓冲区的内容送往屏幕。接着处理下一行，这样不断重复，直到文 件末尾。文件内容并没有改变，除非你使用重定向存储输出。Sed主要用来自动编辑一个或多个文件， 可以将数据行进行替换、删除、新增、选取等特定工作，简化对文件的反复操作，编写转换程序等。

命令格式

sed的命令格式：sed [options] 'command' file(s);
sed的脚本格式：sed [options] -f scriptfile file(s);

选项

-e ： 直接在命令行模式上进行sed动作编辑，此为默认选项;

-f ： 将sed的动作写在一个文件内，用–f filename 执行filename内的sed动作;

-i ： 直接修改文件内容;

-n ： 只打印模式匹配的行；

-r ： 支持扩展表达式;

-h或--help：显示帮助；

-V或--version：显示版本信息。

sed常用命令

a\ 在当前行下面插入文本;

i\ 在当前行上面插入文本;

c\ 把选定的行改为新的文本;

d 删除，删除选择的行;

D 删除模板块的第一行;

s 替换指定字符;

h 拷贝模板块的内容到内存中的缓冲区;

H 追加模板块的内容到内存中的缓冲区;

g 获得内存缓冲区的内容，并替代当前模板块中的文本;

G 获得内存缓冲区的内容，并追加到当前模板块文本的后面;

l 列表不能打印字符的清单;

n 读取下一个输入行，用下一个命令处理新的行而不是用第一个命令;

N 追加下一个输入行到模板块后面并在二者间嵌入一个新行，改变当前行号码;

p 打印模板块的行。 P(大写) 打印模板块的第一行;

q 退出Sed;

b lable 分支到脚本中带有标记的地方，如果分支不存在则分支到脚本的末尾;

r file 从file中读行;

t label if分支，从最后一行开始，条件一旦满足或者T，t命令，将导致分支到带有标号的命令处，或者到脚本的末尾;

T label 错误分支，从最后一行开始，一旦发生错误或者T，t命令，将导致分支到带有标号的命令处，或者到脚本的末尾;

w file 写并追加模板块到file末尾;

W file 写并追加模板块的第一行到file末尾;

! 表示后面的命令对所有没有被选定的行发生作用;

= 打印当前行号;

sed替换标记

g 表示行内全面替换;

p 表示打印行;

w 表示把行写入一个文件;

x 表示互换模板块中的文本和缓冲区中的文本;

y 表示把一个字符翻译为另外的字符（但是不用于正则表达式）;

\1 子串匹配标记;

& 已匹配字符串标记;

实例

替换文本中的字符串：

```
sed 's/book/books/' file
```

`-n` 选项和 `p` 命令一起使用表示只打印那些发生替换的行：

```
sed -n 's/test/TEST/p' file
```

直接编辑文件选项-i，会匹配file文件中每一行的第一个book替换为books

```
sed -i 's/book/books/g' file
```

全面替换标记g

使用后缀 /g 标记会替换每一行中的所有匹配：

```
sed 's/book/books/g' file
```

当需要从第N处匹配开始替换时，可以使用 /Ng：

```
echo sksksksksksk | sed 's/sk/SK/2g'
skSKSKSKSKSK
echo sksksksksksk | sed 's/sk/SK/3g'
skskSKSKSKSK
echo sksksksksksk | sed 's/sk/SK/4g'
skskskSKSKSK
```

定界符

以上命令中字符 / 在sed中作为定界符使用，也可以使用任意的定界符

```
sed 's:test:TEXT:g'
sed 's|test|TEXT|g'
```

定界符出现在样式内部时，需要进行转义：

```
sed 's/\/bin/\/usr\/local\/bin/g'
```

删除操作：d命令

删除空白行：

```
sed '/^$/d' file
```

删除文件的第2行：

```
sed '2d' file
```

删除文件的第2行到末尾所有行：

```
sed '2,$d' file
```

删除文件最后一行：

```
sed '$d' file
```

删除文件中所有开头是test的行：

```
sed '/^test/'d file
```

已匹配字符串标记& 正则表达式 \w+ 匹配每一个单词，使用 [&] 替换它，& 对应于之前所匹配到的单词：

```
echo this is a test line | sed 's/\w\+/[&]/g'
[this] [is] [a] [test] [line]
```

所有以192.168.0.1开头的行都会被替换成它自己加localhost：

```
sed 's/^192.168.0.1/&localhost/' file
192.168.0.1localhost
```

子串匹配标记\1

匹配给定样式的其中一部分：

```
echo this is digit 7 in a number | sed 's/digit \([0-9]\)/\1/'
this is 7 in a number
```

命令中 digit 7，被替换成了 7。样式匹配到的子串是 7，(..) 用于匹配子串，对于匹配到的第一个子串就标记为 \1，依此类推匹配到的第二个结果就是 \2，例如：

```
echo aaa BBB | sed 's/\([a-z]\+\) \([A-Z]\+\)/\2 \1/'
BBB aaa
```

love被标记为1，所有loveable会被替换成lovers，并打印出来：

```
sed -n 's/\(love\)able/\1rs/p' file
```

选定行的范围：,（逗号）

所有在模板test和check所确定的范围内的行都被打印：

```
sed -n '/test/,/check/p' file
```

打印从第5行开始到第一个包含以test开始的行之间的所有行：

```
sed -n '5,/^test/p' file
```

对于模板test和west之间的行，每行的末尾用字符串aaa bbb替换：

```
sed '/test/,/west/s/$/aaa bbb/' file
```

多点编辑：e命令

`-e` 选项允许在同一行里执行多条命令：

```
sed -e '1,5d' -e 's/test/check/' file
```

上面sed表达式的第一条命令删除1至5行，第二条命令用check替换test。命令的执行顺序对结果有影响。如果两个命令都是替换命令，那么第一个替换命令将影响第二个替换命令的结果。

和 -e 等价的命令是 --expression：

```
sed --expression='s/test/check/' --expression='/love/d' file
```

更多详情

HTML5基础

了解HTML5

HTML5属于上一代HTML的新迭代语言，设计HTML5最主要的目的是为了在移动设备上支持多媒体！！！ 例如： video 标签和 audio 及 canvas 标记 新特性：

1. 取消了过时的显示效果标记 `<font></font>` 和 `<center></center>` 等
2. 新表单元素引入
3. 新语义标签的引入
4. canvas标签（图形设计）
5. 本地数据库（本地存储）
6. 一些API 好处： 跨平台 例如：比如你开发了一款HTML5的游戏，你可以很轻易地移植到UC的开放平台、Opera的游戏中心、Facebook应用平台，甚至可以通过封装的技术发放到App Store或Google Play上，所以它的跨平台性非常强大，这也是大多数人对HTML5有兴趣的主要原因。

   缺点：

pc端浏览器支持不是特别友好，造成用户体验不佳

新语义标签

网页布局结构标签及兼容处理

HTML5 语义元素

```
<header></header>
<footer></footer>
<article></article>
<aside></aside>
<nav></nav>
<section></section>
```

多媒体标签及属性介绍

HTML 视频

`<video></video>` 视频 属性：controls 显示控制栏 属性：autoplay 自动播放 属性：loop 设置循环播放 `<audio></audio>` 音频 属性：controls 显示控制栏 属性：autoplay 自动播放
属性：loop 设置循环播放 video标签支持的格式
多媒体标签在网页中的兼容效果方式

```
<video>
   <source src="code/多媒体标签/trailer.mp4">
   <source src="trailer.ogg">
   <source src="trailer.WebM">
</video>
```

新表单元素及属性

智能表单控件

```
<input  type="email">
```

email: 输入合法的邮箱地址 url： 输入合法的网址 number： 只能输入数字 range： 滑块 color： 拾色器 date： 显示日期 month： 显示月份 week ： 显示第几周 time： 显示时间

表单属性

form属性：
autocomplete=on | off 自动完成 novalidate=true | false 是否关闭校验

input属性： *autofocus ： 自动获取焦点 form：
list：

```
<input type="text" list="abc"/>
<datalist id="abc">
    <option value="123">12312</option>
    <option value="123">12312</option>
    <option value="123">12312</option>
    <option value="123">12312</option>
</datalist>
```

multiple： 实现多选效果 *placeholder： 占位符（提示信息）* required： 必填项

API

获取页面元素及类名操作和自定义属性

```
//选择器： 可以是css中的任意一种选择器
//通过该选择器只能选中第一个元素。
document.querySelector("选择器");

//与document.querySelector区别： querySelectorAll 可以选中所有符合选择器规则的元素，返回的
document.querySelectorAll("选择器");

Dom.classList.add("类名"); //给当前dom元素添加类样式

Dom.classList.remove("类名"); //给当前dom元素移除类样式

classList.contains("类名"); //检测是否包含类样式

classList.toggle("active");  //切换类样式（有就删除，没有就添加）
```

自定义属性

data-自定义属性名： 在标签中，以data-自定义名称

1. 获取自定义属性 Dom.dataset 返回的是一个对象 Dom.dataset.属性名 或者 Dom.dataset[属性名]

   注意：属性名是不包含**data-**

2. 设置自定义属性 Dom.dataset.自定义属性名=值 或者 Dom.dataset[自定义属性名]=值；

文件读取

- FileReader

```
FileReader              接口有3个用来读取文件方法返回结果在result中
  readAsBinaryString     ---将文件读取为二进制编码
  readAsText            ---将文件读取为文本
  readAsDataURL            ---将文件读取为DataURL
```

- FileReader 提供的事件模型

```
onabort         中断时触发
onerror         出错时触发
onload          文件读取成功完成时触发
onloadend     读取完成触发，无论成功或失败
onloadstart     读取开始时触发
onprogress     读取中
```

获取网络状态

- 获取当前网络状态

```
    window.navigator.onLine 返回一个布尔值
```

- 网络状态事件

```
    1. window.ononline
          2. window.onoffline
```

获取地理定位

1. 获取一次当前位置

```
window.navigator.geolocation.getCurrentPosition(success,error);
1. coords.latitude    维度
2. coords.longitude    经度
```

2. 实时获取当前位置

```
 window.navigator.geolocation.watchPosition(success,error);
```

本地存储

随着互联网的快速发展，基于网页的应用越来越普遍，同时也变的越来越复杂，为了满足各种各样的需求，会经常性在本地存储大量的数据，传统方式我们以document.cookie来进行存储的，但是由于其存储大小只有4k左右，并且解析也相当的复杂，给开发带来诸多不便，HTML5规范则提出解决方案，使用sessionStorage和localStorage存储数据。

- localStorage：
  - 永久生效
  - 多窗口共享
  - 容量大约为20M

```
window.localStorage.setItem(key,value)  //设置存储内容
window.localStorage.getItem(key)          //获取内容
window.localStorage.removeItem(key)     //删除内容
window.localStorage.clear()            //清空内容
```

- sessionStorage：
  - 生命周期为关闭当前浏览器窗口
  - 可以在同一个窗口下访问

- 数据大小为5M左右

```
window.sessionStorage.setItem(key,value)
window.sessionStorage.getItem(key)
window.sessionStorage.removeItem(key)
window.sessionStorage.clear()
```

操作多媒体

HTML 5 视频/音频参考手册

Canvas

绘图方法

```
<canvas id="myCanvas" width="200" height="100"></canvas>
<script type="text/javascript">
    var ctx=document.getElementById("myCanvas");
    ctx.moveTo(x,y)    //落笔
    ctx.lineTo(x,y)    //连线
    ctx.stroke();       //描边
    ctx.beginPath();  //开启新的图层
</script>
```

- 样式： strokeStyle="值"
- 线宽： linewidth="值" 备注：不需要带单位
- 线连接方式： lineJoin: round | bevel | miter (默认)
- 线帽（线两端的结束方式）： lineCap: butt(默认值) | round | square

```
ctx.closePath(); //闭合路径
```

渐变方案

线性渐变

```
var grd=ctx.createLinearGradient(x0,y0,x1,y1);
```

- x0-->渐变开始的x坐标
- y0-->渐变开始的y坐标
- x1-->渐变结束的x坐标
- y1-->渐变结束的y坐标

```
grd.addColorStop(0,"black");       //设置渐变的开始颜色
grd.addColorStop(0.1,"yellow");    //设置渐变的中间颜色
grd.addColorStop(1,"red");         //设置渐变的结束颜色
ctx.strokeStyle=grd;
ctx.stroke();
```

- 备注：渐变的开始位置和结束位置介于0-1之间，0代表开始，1代表结束。中间可以设置任何小数

径向渐变

```
ctx.createradialGradient(x0,y0,r0,x1,y1,r1);
```

- (x0,y0)：渐变的开始圆的 x,y 坐标
- r0：开始圆的半径
- (x1,y1)：渐变的结束圆的 x,y 坐标
- r1：结束圆的半径

填充效果

```
ctx.fill();          //设置填充效果
ctx.fillstyle="值"; //设置填充颜色
```

非零环绕原则

绘制一个如下图形

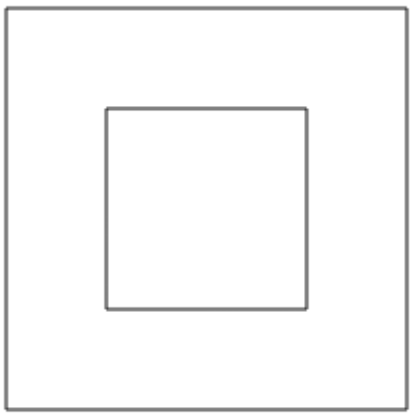


图 - 矩形

- 非零环绕原则：
  - 任意找一点，越简单越好
  - 以点为圆心，绘制一条射线，越简单越好（相交的边越少越好）
  - 以射线为半径顺时针旋转，相交的边同向记为+1，反方向记为-1，如果相加的区域等于0，则不填充。
  - 非零区域填充

绘制虚线

1. 原理：

   设置虚线其实就是设置实线与空白部分直接的距离,利用数组描述其中的关系

例如： [10,10] 实线部分10px 空白部分10px

例如： [10,5] 实线部分10px 空白部分5px

例如： [10,5,20] 实线部分10px 空白5px 实线20px 空白部分10px 实线5px 空白20px....

2. 绘制：

```
ctx.setLineDash(数组);
ctx.stroke();
ctx.moveTo(100, 100);
ctx.lineTo(300, 100);
ctx.setLineDash([2,4]);
ctx.stroke();
```

注意： 如果要将虚线改为实线，只要将数组改为空数组即可。

绘制动画效果

绘制一个描边矩形： `content.strokeRect(x,y,width,height)`

绘制一个填充矩形： `content.fillRect(x,y,width,height)`

清除： `content.clearRect(x,y,width,height)`

实现动画效果：

1. 先清屏
2. 绘制图形
3. 处理变量

绘制文本

绘制填充文本

```
content.fillText(文本的内容,x,y)
```

绘制镂空文本

```
content.strokeText();
```

设置文字大小：

```
 content.font="20px 微软雅黑"
```

文字水平对齐方式【文字在圆心点位置的对齐方式】

```
 content.textalign="left | right | center"
```

文字垂直对齐方式

```
content.textBaseline="top | middle | bottom | alphabetic(默认)"
```

文字阴影效果

```
ctx.shadowColor="red";  设置文字阴影的颜色
ctx.ShadowOffsetX=值;   设置文字阴影的水平偏移量

ctx.shadowOffsetY=值;   设置文字阴影的垂直偏移量

ctx.shadowBlur=值;      设置文字阴影的模糊度
```

绘制图片

```
//将图片绘制到画布的指定位置
content.drawImage(图片对象,x,y);
//将图片绘制到指定区域大小的位置  x,y指的是矩形区域的位置，width和height指的是矩形区域的大
content.drawImage(图片对象,x,y,width,height);
content.drawImage(图片对象,sx,sy,swidth,sheight,dx,dy,dwidth,dheight);
```

- sx,sy 指的是要从图片哪块区域开始绘制，
- swidth，sheight 是值 截取图片区域的大小
- dx,dy 是指矩形区域的位置，dwidth,dheight是值矩形区域的大小

解决图片绘制到某一个区域的按原比例缩放绘制：绘制宽：绘制高==原始宽：原始高

绘制圆弧

```
content.arc(x,y,radius,startradian,endradian[,direct]);
```

1. x,y 圆心的坐标
2. radius 半径
3. startradian 开始弧度
4. endradian 结束弧度
5. direct 方向（默认顺时针 false） true 代表逆时针

角度 和 弧度的关系： 角度:弧度= 180:pi

- 0度角在哪？ 以圆心为中心向右为0角 顺时针为正，逆时针为负
- 特殊值

  0度 = 0弧度

  30度 = π/6 (180度的六分之一)

  45度 = π/4

  60度 = π/3

  90度 = π/2

  180度 = π

  360度 = 2π

  绘制圆上任意点：

- 公式：
  - x=ox+r*cos(弧度)*
  - y=oy+r*sin(弧度)
  - ox: 圆心的横坐标
  - oy: 圆心的纵坐标
  - r： 圆的半径

平移【坐标系圆点的平移】

```
ctx.translate(x,y);
```

- 通过该方法可以将原点的位置进行重新设置。
- translate(x,y) 中不能设置一个值
- 与moveTo(x,y) 的区别：
  - moveTo(x,y) 指的是将画笔的落笔点的位置改变，而坐标系中的原点位置并没有发生改变
  - translate(x,y) 是将坐标系中的原点位置发生改变

旋转【坐标系旋转】

```
ctx.rotate(弧度)
```

伸缩

```
 ctx.scale(x,y)
```

备注： 沿着x轴和y轴缩放 x,y 为倍数 例如： 0.5 1

CSS3

样式

背景

规定背景图片的定位区域

```
background-origin:
    padding-box     背景图像相对内边距定位（默认值）
    border-box        背景图像相对边框定位【以边框左上角为参照进行位置设置】
    content-box     背景图像相对内容区域定位【以内容区域左上角为参照进行位置设置】
```

备注：1. 默认盒子的背景图片是在盒子的内边距左上角对齐设置。

规定背景的绘制区域

```
background-clip:
     border-box          背景被裁切到边框盒子位置 【将背景图片在整个容器中显示】
     padding-box     背景被裁切到内边距区域【将背景图片在内边距区域（包含内容区域）显示】
     content-box     背景被裁切到内容区域【将背景图片在内容区域显示】
```

规定背景图片的尺寸

```
background-size:
     cover
     contain
```

边框

- box-shadow： 盒子阴影
- border-radius： 边框圆角
- border-image: 边框图片

```
/* 设置边框图片 */
border-image-source: url("2.png");

/* 边框图片裁切 : 不需要带单位*/
border-image-slice: 20;

/* 设置边框图片的平铺方式 */
/* border-image-repeat: stretch; */
border-image-repeat: round;
/*  border-image-repeat: repeat; */

border-image-width: 20px;
```

文本

```
text-shadow:  设置文本阴影
```

选择器

1. 属性选择器： [属性名=值] {} [属性名] {} 匹配对应的属性即可 [属性名^=值] {} 以值开头 [属性名*=值] {} 包含 [属性名$=值] {} 以值结束
2. 结构伪类选择器：

   ```
   `:first-child {}`      选中父元素中第一个子元素
   ```

   `:last-child {}` 选中父元素中最后一个子元素 `:nth-child(n) {}` 选中父元素中正数第n个子元素 `:nth-last-child(n) {}` 选中父元素中倒数第n个子元素

   - 备注：
     - n 的取值大于等于0
     - n 可以设置预定义的值
     - odd[选中奇数位置的元素]
     - even【选中偶数位置的元素】
     - n 可以是一个表达式： an+b的格式

   i. 其他选择器： `:target` 被锚链接指向的时候会触发该选择器 `::selection` 当被鼠标选中的时候的样式 `::first-line` 选中第一行 `::first-letter` 选中第一个字符

颜色渐变

- 线性渐变：

  1. 开始颜色和结束颜色
  2. 渐变的方向
  3. 渐变的范围

```
background-image:  linear-gradient(
            to right,
            red,
            blue
     );
```

备注：

4. to + right | top | bottom | left
5. 通过角度表示一个方向 0deg [从下向上渐变] 90deg【从左向右】

- 径向渐变：

```
/* 径向渐变 */
background-image: radial-gradient(
    100px at center,
    red,
    blue
);
```

2D转换

- 位移

```
`transform: translate(100px,100px);`
```

备注： 位移是相对元素自身的位置发生位置改变

- 旋转

```
`transform: rotate(60deg);`
```

备注： 取值为角度

- 缩放

```
`transform: scale(0.5,1);`
```

备注：取值为倍数关系，缩小大于0小于1，放大设置大于1

- 倾斜

```
`transform: skew(30deg,30deg);`
```

备注：

```
第一个值代表沿着x轴方向倾斜
第二个值代表沿着y轴方向倾斜
```

3D转换

- 位移 `transform: translateX() translateY() translateZ()`
- 旋转 `transform: rotateX(60deg) rotateY(60deg) rotateZ(60deg);`
- 缩放 `transform: scaleX(0.5) scaleY(1) scaleZ(1);`
- 倾斜 `transform: skewX(30deg) skewY();`
- 将平面图形转换为立体图形 `transform-style: preserve-3d;`

动画

过渡

```
/* 设置哪些属性要参与到过渡动画效果中： all */
transition-property: all;

/* 设置过渡执行时间 */

transition-duration: 1s;

/* 设置过渡延时执行时间 */
transition-delay: 1s;

/* 设置过渡的速度类型 */

transition-timing-function: linear;
```

自定义动画

```
/* 1定义动画集 */
@keyframes  rotate {

    /* 定义开始状态  0%*/
    from {
        transform: rotateZ(0deg);
    }

    /* 结束状态 100%*/
    to {
        transform: rotateZ(360deg);
    }
}
```

注意： 如果设置动画集使用的是百分比，那么记住百分比是相对整个动画执行时间的。

伸缩布局或者弹性布局

- 设置父元素为伸缩盒子【直接父元素】

```
display:  flex
```

为什么在伸缩盒子中，子元素会在一行上显示？

1. 子元素是按照伸缩盒子中主轴方向显示
2. 只有伸缩盒子才有主轴和侧轴
3. 主轴： 默认水平从左向右显示
4. 侧轴： 始终要垂直于主轴

- 设置伸缩盒子主轴方向（flex-direction）

```
flex-direction: row;
flex-direction: row-reverse;
flex-direction: column;
flex-direction: column-reverse;
```

- 设置元素在主轴的对齐方式(justify-content)

```
/* 设置子元素在主轴方向的对齐方式 */
justify-content: flex-start;
justify-content: flex-end;
justify-content: center;
justify-content: space-between;
justify-content: space-around;
```

- 设置元素在侧轴的对齐方式 （align-items）

```
align-items: flex-start;
align-items: flex-end;
align-items: center;
/* 默认值 */
align-items: stretch;
```

- 设置元素是否换行显示（flex-wrap）

1. 在伸缩盒子中所有的元素默认都会在一行上显示
2. 如果希望换行： `flex-wrap: wrap | nowrap;`

- 设置元素换行后的对齐方式（ align-content）

```
align-content: flex-start;
align-content: flex-end;
align-content: center;
align-content: space-around;
align-content: space-between;
/* 换行后的默认值 */
align-content: stretch;
```

JS学习

JS基本介绍

- JS的用途：Javascript可以实现浏览器端、服务器端(nodejs)。。。
- 浏览器端JS由以下三个部分组成：
  - ECMAScript：基础语法(数据类型、运算符、函数。。。)
  - BOM(浏览器对象模型)：window、location、history、navigator。。。
  - DOM(文档对象模型)：div、p、span。。。
- ECMAScript又名es，有以下重大版本：
  - 旧时代：
    - es1.0。。。es3.1
  - 新时代：
    - es5
    - es6(es2015)

- es7(es2016)、es8(es2017)

数据类型

- 基本数据类型——值类型：(数字、字符串、布尔值、null、undefined)
  - undefined类型？
- 复杂数据类型——引用类型：(对象)
  - 数组
  - 函数
  - 正则表达式
  - Date

对象的基本使用

创建一个对象

```
var student={
    name:"李白" , //student有一个name属性，值为"李白"
    grade:"初一" ,
    //a、student有一个say属性，值为一个函数
    //b、student有一个say方法
    say:function(){
        console.log("你好");
    },
    run:function(speed){
        console.log("正在以"+speed+"米/秒的速度奔跑");
    }
}
```

对象是键值对的集合：对象是由属性和方法构成的 (ps：也有说法为：对象里面皆属性，认为方法也是一个属性)

- name是属性 grade是属性
- say是方法 run是方法

对象属性操作

获取属性：

第一种方式：.语法

- student.name 获取到name属性的值，为："李白"
- student.say 获取到一个函数

第二种方式：[]语法

- student["name"] 等价于student.name
- student["say"] 等价于student.say

号外：2种方式的差异：

- .语法更方便，但是坑比较多(有局限性)，比如：
  - .后面不能使用js中的关键字、保留字(class、this、function。。。)
  - .后面不能使用数字

```
var obj={};
obj.this=5; //语法错误
obj.0=10;   //语法错误
```

- []使用更广泛
  - o1[变量name]
  - ["class"]、["this"]都可以随意使用 `obj["this"]=10`
  - [0]、[1]、[2]也可以使用
    - `obj[3]=50 = obj["3"]=50`
    - 思考：为什么obj[3]=obj["3"]
  - 甚至还可以这样用：["[object Array]"]
    - jquery里面就有这样的实现
  - 也可以这样用："{abc}"]
    - 给对象添加了{abc}属性

设置属性

- `student["gender"]="男"` 等价于： `student.gender="男"`
  - 含义：如果student对象中没有gender属性，就添加一个gender属性，值为"男"
  - 如果student对象中有gender属性，就修改gender属性的值为"男"
- 案例1： `student.isFemale=true`
- 案例2： `student["children"]=[1,2,5]`
- 案例3：

```
student.toShanghai=function(){
    console.log("正在去往上海的路上")
}
```

删除属性

- delete student["gender"]
- delete student.gender

通过构造函数创建对象

构造函数创建对象的例子：

- var xiaoming = new Object() --> var xiaoming = {};
- var now = new Date()
- var rooms = new Array(1,3,5) --> var rooms = [1,3,5]
- `var isMale=/123/;` ==> `var isMale=new RegExp("123")`
  - isMale是通过RegExp构造函数创建出来的对象

    - isMale是RegExp构造函数的实例
- 以上例子中，Object、Date、Array都是内置的构造函数

自定义一个构造函数来创建对象

- 构造函数

```
function Person(name,age){
    this.name=name;
    this.age=age;
}
var p1=new Person("赵云",18)
```

- 说明：`p1就是根据【Person构造函数】创建出来的对象`

构造函数的概念

- 任何函数都可以当成构造函数 `function CreateFunc(){ }`
- 只要把一个函数通过new的方式来进行调用，我们就把这一次函数的调用方式称之为：构造函数的调用
    - new CreateFunc(); 此时CreateFunc就是一个构造函数
    - CreateFunc(); 此时的CreateFunc并不是构造函数

关于new Object()

- new Object()等同于对象字面量{}

构造函数的执行过程

`var p1=new Person();`

- 1、创建一个对象 (我们把这个对象称之为Person构造函数的实例)- `_p1`
- 2、创建一个内部对象，`this`，将this指向该实例(_p1)
- 3、执行函数内部的代码，其中，操作this的部分就是操作了该实例(_p1)
- 4、返回值：
    - a、如果函数没有返回值(没有return语句)，那么就会返回构造函数的实例(p1)
    - b、如果函数返回了一个基本数据类型的值，那么本次构造函数的返回值是该实例(_p1)

```
function fn(){
}
var f1=new fn();    //f1就是fn的实例

function fn2(){
    return "abc";
}
var f2=new fn2();   //f2是fn2构造函数的实例
```

    - c、如果函数返回了一个复杂数据类型的值，那么本次函数的返回值就是该值

```
function fn3(){
    return [1,3,5];
    //数组是一个对象类型的值，
    //所以数组是一个复杂数据类型的值
    //本次构造函数的真正返回值就是该数组
    //不再是fn3构造函数的实例
}
var f3=new fn3();   //f3还是fn3的实例吗？错
//f3值为[1,3,5]
```

- 如何判断一个数据是否是复杂数据类型？

  使用排除法：

  a、看它的值是不是数字、字符串、布尔值、null、undefined

  b、如果不是以上5种值，那就是复杂数据类型

- 为什么要理解构造函数的返回值？

  String是一个内置函数：a、String() b、new String()

  一个函数通过new调用，或者不通过new调用，很多时候会有截然不同的返回值

- 我们如何分辨出一个对象到底是不是某个构造函数的实例？

  a、 var isTrue=xxx instanceof Person

- 如何识别xxx对象是哪个构造函数的实例？

  xxx.**proto**属性，也是对象，该对象中一般会有一个constructor属性，这个值指向PPP，那么xxx就是PPP的构造函数

- typeof运算符不能用来判断对象的构造函数

继承

JS中继承的概念：

- 通过【某种方式】让一个对象可以访问到另一个对象中的属性和方法，我们把这种方式称之为继承 `并不是所谓的xxx extends yyy`

为什么要使用继承？

- 有些对象会有方法(动作、行为)，而这些方法都是函数，如果把这些方法和函数都放在构造函数中声明就会导致内存的浪费

```
function Person(){
    this.say=function(){
        console.log("你好")
    }
}
var p1=new Person();
var p2=new Person();
console.log(p1.say === p2.say);   //false
```

继承的第一种方式：原型链继承1

```
Person.prototype.say=function(){
    console.log("你好")
}
```

- 缺点：添加1、2个方法无所谓，但是如果方法很多会导致过多的代码冗余

继承的第二种方式：原型链继承2

```
Person.prototype={
    constructor:Person,
    say:function(){
        console.log("你好");
    },
    run:function(){
        console.log("正在进行百米冲刺");
    }
}
```

- 注意点：
- a、一般情况下，应该先改变原型对象，再创建对象
- b、一般情况下，对于新原型，会添加一个constructor属性，从而不破坏原有的原型对象的结构

继承的第三种方式：拷贝继承(混入继承)

- 场景：有时候想使用某个对象中的属性，但是又不能直接修改它，于是就可以创建一个该对象的拷贝
- 实现1：

```
var source={name:"李白",age:15}
var target={};
target.name=source.name
target.age=source.age;
```

- 上面的方式很明显无法重用，实际代码编写过程中，很多时候都会使用拷贝继承的方式，所以为了重用，可以编写一个函数把他们封装起来：

```
function extend(target,source){
    for(key in source){
        target[key]=source[key];
    }
    return target;
}
extend(target,source)
```

- 由于拷贝继承在实际开发中使用场景非常多，所以很多库都对此有了实现
  - jquery：$.extend
- es6中有了对象扩展运算符仿佛就是专门为了拷贝继承而生：

```
var source={name:"李白",age:15}
var target={ ...source }
```

继承的第四种方式：原型式继承

- 场景：
  - 创建一个纯洁的对象
  - 创建一个继承自某个父对象的子对象
- 使用方式：
  - 空对象：Object.create(null)

```
var o1={ say:function(){} }
var o2=Object.create(o1);
```

继承的第五种方式：借用构造函数实现继承

- 场景：适用于2种构造函数之间逻辑有相似的情况

function Animal(name){ this.name=name; } function Person(name,age){ this.name=name; this.age=age; }

- 以上代码用借用构造函数实现

```
function Animal(name,age){
  this.name=name;
  this.age=age;
}
function Person(name,age,address){
  Animal.call(this,name);
  //this.name=name;
  //this.age=age;
  this.address=address;
}
```

原型链

闭包

变量作用域

- 变量作用域的概念：就是一个变量可以使用的范围
- JS中首先有一个最外层的作用域：称之为全局作用域
- JS中还可以通过函数创建出一个独立的作用域，其中函数可以嵌套，所以作用域也可以嵌套

作用域链

- 由于作用域是相对于变量而言的，而如果存在多级作用域，这个变量又来自于哪里？这个问题就需要好好地探究一下了，我们把这个变量的查找过程称之为变量的作用域链

- 简单来说，作用域链可以用以下几句话来概括：(或者说：确定一个变量来自于哪个作用域)
    - 查看当前作用域，如果当前作用域声明了这个变量，就确定结果
        - 查找当前作用域的上级作用域，也就是当前函数的上级函数，看看上级函数中有没有声明
            - 再查找上级函数的上级函数，直到全局作用域为止
                - 如果全局作用域中也没有，我们就认为这个变量未声明(xxx is not defined)
- 举例1：

```
var name="张三";
function f1(){
    var name="abc";
    console.log(name);
}
f1();
```

- 举例2：

```
var name="张三";
function f1(){
    console.log(name);
    var name="abc";
}
f1();
```

- 举例3：

```
var name="张三";
function f1(){
    console.log(name);
    var name="abc";
}
f1();
```

- 举例4：

```
var name="张三";
function f1(){
    return function(){
        console.log(name);
    }
    var name="abc";
}
var fn=f1();
fn();
```

- 举例5：

```
var name="张三";
function f1(){
    return {
        say:function(){
            console.log(name);
            var name="abc";
        }
    }
}
var fn=f1();
```

闭包的问题

```
function fn(){
    var a=5;
    return function(){
        a++;
        console.log(a);
    }
}
var f1=fn();
f1();
f1();
f1();
```

闭包问题的产生原因

- 函数执行完毕后，作用域中保留了最新的a变量的值

闭包的应用场景

- 模块化
- 防止变量被破坏

es6内容

- 1、解构赋值
- 2、函数rest参数
- 3、箭头函数
  - 箭头函数和普通函数有哪些不同？(4点)
- 4、对象的Object.assign
- 5、promise
- 6、generator
- 7、async
- 8、class
- 9、module

原型

- 原型很多人开发用不到？
  - 很多人都用es6/7/8开发，确实用的比较少
  - 如果你用es5之前的版本开发代码(IE8、IE7。。。)，可能天天都要写原型

- 理解了原型，才是理解了JS面向对象的核心，没有理解原型，你就没有理解面向对象的核心

C语言学习笔记

C语言中的 32 个关键字

int	float	double	char	short	long	signed
if	else	switch	case	default	for	while
break	continue	return	void	const	sizeof	struct
static	extern	auto	register	enum	goto	union

二进制

二进制加法：1+0=1、1+1=10、11+10=101、111+111=1110

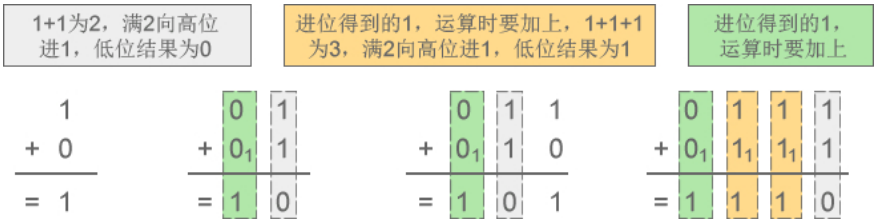


图 - 二进制加法示意图

二进制减法：1-0=1、10-1=1、101-11=10、1100-111=101

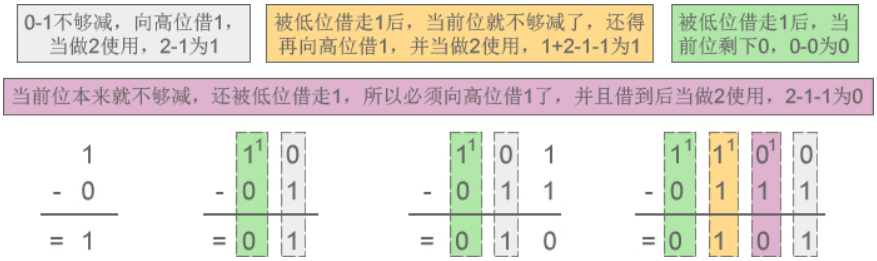


图 - 二进制减法示意图

八进制

八进制加法：3+4=7、5+6=13、75+42=137、2427+567=3216

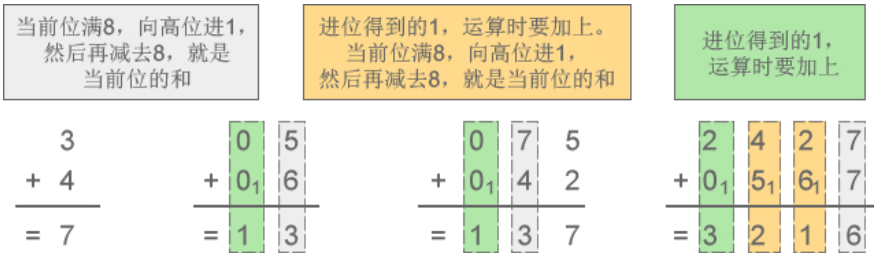
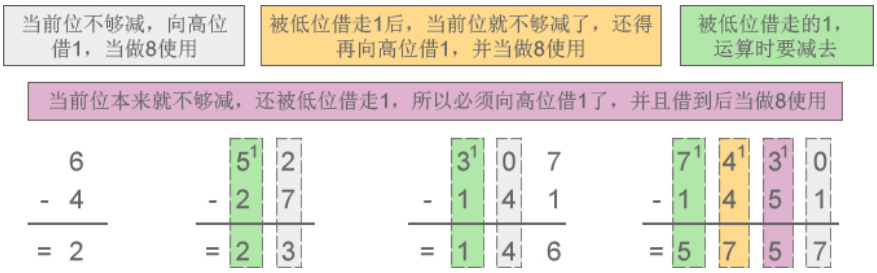


图 - 八进制加法示意图

八进制减法：6-4=2、52-27=23、307-141=146、7430-1451=5757



十六进制

十六进制中，用A来表示10，B表示11，C表示12，D表示13，E表示14，F表示15，因此有 0~F 共16个数字，基数为16， 加法运算时逢16进1，减法运算时借1当16。例如，数字 0、1、6、9、A、D、F、419、EA32、80A3、BC00 都是有效的十六进制。

十六进制加法：6+7=D、18+BA=D2、595+792=D27、2F87+F8A=3F11

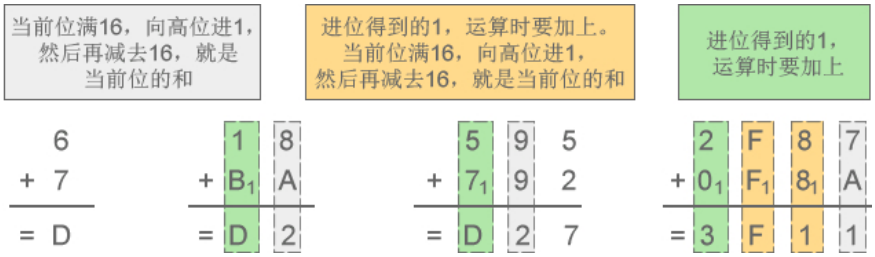


图 - 十六进制加法示意图

十六进制减法：D-3=A、52-2F=23、E07-141=CC6、7CA0-1CB1=5FEF

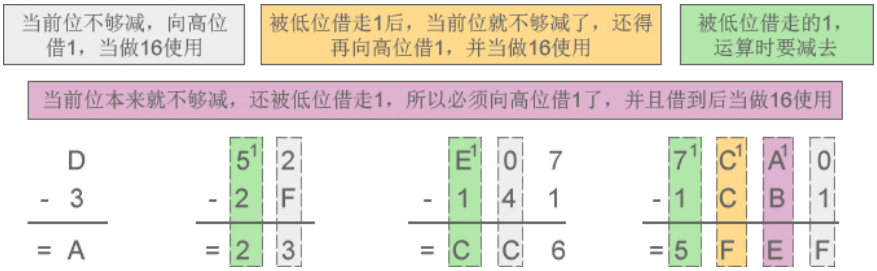


图 - 十六进制减法示意图

将二进制、八进制、十六进制转换为十进制

二进制、八进制和十六进制向十进制转换都非常容易，就是“按权相加”。所谓“权”，也即“位权”。

假设当前数字是 N 进制，那么：

- 对于整数部分，从右往左看，第 i 位的位权等于N^{i-1}
- 对于小数部分，恰好相反，要从左往右看，第 j 位的位权为N^{-j}。

更加通俗的理解是，假设一个多位数（由多个数字组成的数）某位上的数字是 1，那么它所表示的数值大小就是该位的位权。
转换成十进制的例子：

- 二进制：$1001 = 1\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0 = 8 + 0 + 0 + 1 = 9$（十进制）
- 二进制：$101.1001 = 1\times2^2 + 0\times2^1 + 1\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 0\times2^{-3} + 1\times2^{-4} = 4 + 0 + 1 + 0.5 + 0 + 0 + 0.0625 = 5.5625$（十进制）
- 八进制：$302 = 3\times8^2 + 0\times8^1 + 2\times8^0 = 192 + 0 + 2 = 194$（十进制）
- 八进制：$302.46 = 3\times8^2 + 0\times8^1 + 2\times8^0 + 4\times8^{-1} + 6\times8^{-2} = 192 + 0 + 2 + 0.5 + 0.09375= 194.59375$（十进制）
- 十六进制：$EA7 = 14\times16^2 + 10\times16^1 + 7\times16^0 = 3751$（十进制）

将十进制转换为二进制、八进制、十六进制

将十进制转换为其它进制时比较复杂，整数部分和小数部分的算法不一样。

整数部分

十进制整数转换为 N 进制整数采用“除 N 取余，逆序排列”法。具体做法是：

- 将 N 作为除数，用十进制整数除以 N，可以得到一个商和余数；
- 保留余数，用商继续除以 N，又得到一个新的商和余数；
- 仍然保留余数，用商继续除以 N，还会得到一个新的商和余数；
- ……
- 如此反复进行，每次都保留余数，用商接着除以 N，直到商为 0 时为止。

把先得到的余数作为 N 进制数的低位数字，后得到的余数作为 N 进制数的高位数字，依次排列起来，就得到了 N 进制数字。

下图演示了将十进制数字 36926 转换成八进制的过程：

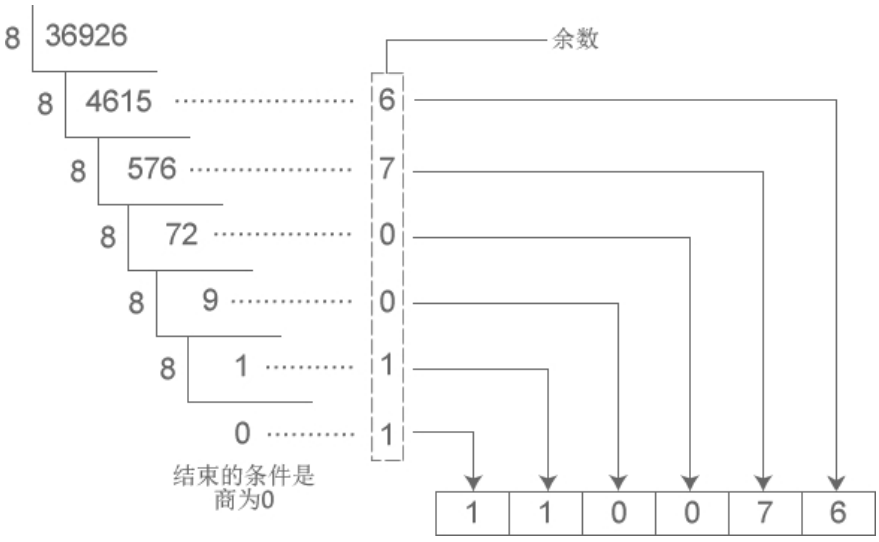


图 - 36926 转换成八进制

从图中得知，十进制数字 36926 转换成八进制的结果为 110076。

下图演示了将十进制数字 42 转换成二进制的过程：

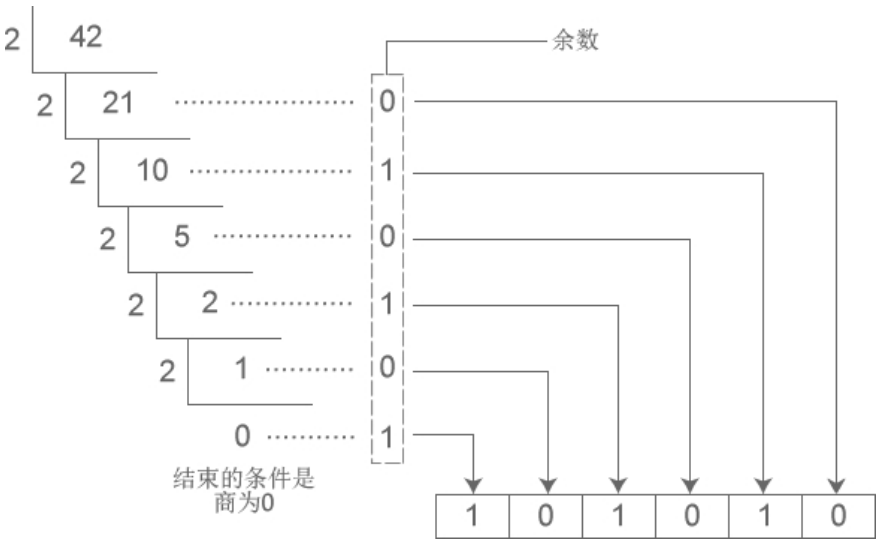


图 - 42 转换成二进制

从图中得知，十进制数字 42 转换成二进制的结果为 101010。

小数部分

十进制小数转换成 N 进制小数采用“乘 N 取整，顺序排列”法。具体做法是：

- 用 N 乘以十进制小数，可以得到一个积，这个积包含了整数部分和小数部分；
- 将积的整数部分取出，再用 N 乘以余下的小数部分，又得到一个新的积；
- 再将积的整数部分取出，继续用 N 乘以余下的小数部分；
- ……
- 如此反复进行，每次都取出整数部分，用 N 接着乘以小数部分，直到积中的小数部分为 0，或者达到所要求的精度为止。

把取出的整数部分按顺序排列起来，先取出的整数作为 N 进制小数的高位数字，后取出的整数作为低位数字，这样就得到了 N 进制小数。

下图演示了将十进制小数 0.930908203125 转换成八进制小数的过程：

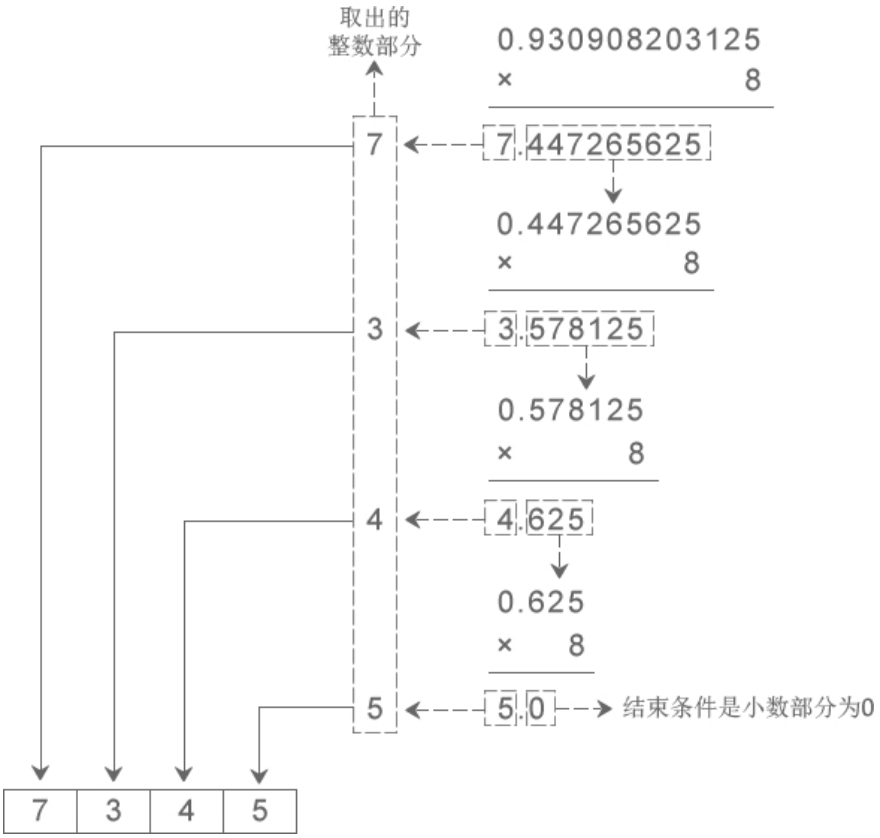


图 - 0.930908203125 转换成八进制

从图中得知，十进制小数 0.930908203125 转换成八进制小数的结果为 0.7345。

下表列出了前 17 个十进制整数与二进制、八进制、十六进制的对应关系：

十进制	0	1	2	3	4	5	6	7	8
二进制	0	1	10	11	100	101	110	111	1000
八进制	0	1	2	3	4	5	6	7	10
十六进制	0	1	2	3	4	5	6	7	8

二进制和八进制、十六进制的转换

其实，任何进制之间的转换都可以使用上面讲到的方法，只不过有时比较麻烦，所以一般针对不同的进制采取不同的方法。将二进制转换为八进制和十六进制时就有非常简洁的方法，反之亦然。

二进制整数和八进制整数之间的转换

二进制整数转换为八进制整数时，每三位二进制数字转换为一位八进制数字，运算的顺序是从低位向高位依次进行，高位不足三位用零补齐。下图演示了如何将二进制整数 1110111100 转换为八进制：

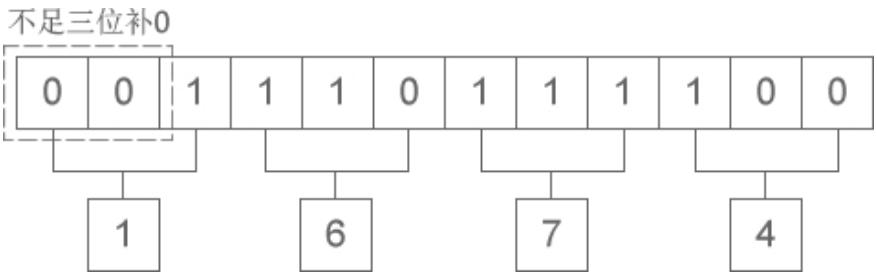


图 - 1110111100 转换为八进制

从图中可以看出，二进制整数 1110111100 转换为八进制的结果为 1674。

八进制整数转换为二进制整数时，思路是相反的，每一位八进制数字转换为三位二进制数字，运算的顺序也是从低位向高位依次进行。下图演示了如何将八进制整数 2743 转换为二进制：

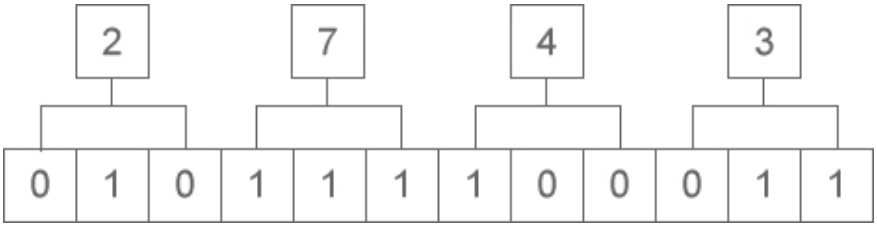


图 - 2743 转换为二进制

从图中可以看出，八进制整数 2743 转换为二进制的结果为 10111100011。

二进制整数和十六进制整数之间的转换

二进制整数转换为十六进制整数时，每四位二进制数字转换为一位十六进制数字，运算的顺序是从低位向高位依次进行，高位不足四位用零补齐。

下图演示了如何将二进制整数 10 1101 0101 1100 转换为十六进制：

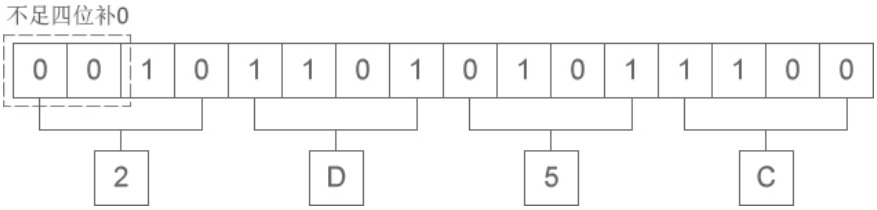


图 - 二进制整数 10 1101 0101 1100 转换为十六进制

从图中可以看出，二进制整数 10 1101 0101 1100 转换为十六进制的结果为 2D5C。

十六进制整数转换为二进制整数时，思路是相反的，每一位十六进制数字转换为四位二进制数字，运算的顺序也是从低位向高位依次进行。下图演示了如何将十六进制整数 A5D6 转换为二进制：

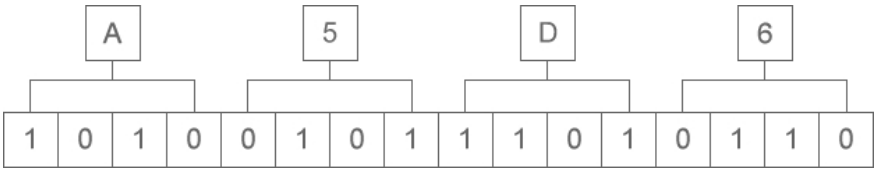


图 - 十六进制整数 A5D6 转换为二进制

从图中可以看出，十六进制整数 A5D6 转换为二进制的结果为 1010 0101 1101 0110。

在 linux 下使用 adb 刷机

使用adb fastboot 线刷小米平板4Plus

1. 安装adb fastboot工具

```
sudo apt install adb fastboot
```

2. 通过USB将您的设备连接到电脑，并验证手机连接

```
adb devices
```

3. 重启到fastboot模式

```
adb reboot-bootloader
```

4. 下载官方线刷包后解压链接

```
tar zxvf clover_images_V10.3.2.0.ODJCNXM_20190515.0000.00_8.1_cn_4388d99e5e.tgz
```

5. 进入解压后的目录

```
cd clover_images_V10.3.2.0.ODJCNXM_20190515.0000.00_8.1_cn
```

6. 开始刷机
   - 修改为可执行 `chmod +x flash_all.sh`
   - fastboot模式下再次验证手机连接 `fastboot devices`
   - 执行刷机脚本[1] `sudo sh flash_all.sh`

通过 fastboot 安装 Recovery

1. 下载 Recovery - 例如 TWRP。
   - 您可以前往 魔趣下载站 的设备页面，点击 **Recovery** 下载。
   - 如果没有找到适用于您设备的 Recovery，请借助互联网搜索设备维护者或资深玩家发布的教程。
2. 通过USB将您的设备连接到电脑。
3. 在电脑上打开命令提示符（Windows）或 终端 （Linux 或 macOS）并输入：

```
adb reboot bootloader
```

   您也可以通过组合键启动 fastboot 模式：

- 关闭设备后，按住音量调低 + 电源键，直到屏幕上方出现“FASTBOOT”字样，然后松开。

4. 一旦设备处于 fastboot 模式，请通过键入以下内容验证您的 PC 是否找到它：

```
fastboot devices
```

5. 将 Recovery 刷入到您的设备。

```
fastboot flash recovery twrp-x.x.x-x-x.img
```

6. 现在进入 Recovery 模式以验证安装：

   - 关闭设备后，按住音量调高 + 电源键，直到进入 Recovery 模式，然后松开。

通过 Recovery 安装魔趣ROM

1. 下载你想要安装的魔趣ROM包。

   - 可选项，下载第三方扩展包，例如 OpenGapps

2. 如果您尚未进入 Recovery 模式，请重启到 Recovery 模式。

   - 关闭设备后，按住音量调高 + 电源键，直到进入 Recovery 模式，然后松开。

> [1]. 该脚本删除所有数据，可以选择其他sh结尾的脚本，根据名字可以猜到大概意思 ↩

树莓派使用记录

下载

选择自己要安装的镜像下载

校验包，解压

```
sha1sum 2013-09-25-wheezy-raspbian.zip
unzip  2013-09-25-wheezy-raspbian.zip
```

查看当前哪些设备已经挂载, `df -h` ，插入u盘或sd卡再执行一次 为了防止在写入镜像的时候有其他读取或写入，我们需要卸载设备。两个分区都要卸载。

```
umount  /dev/sdb1
umount  /dev/sdb2
```

xz烧录命令(该命令尝试失败)

```
sudo xz -cd kali-2017.3-rpi3-nexmon.img.xz> /dev/sdb
```

查看烧录进度 `sudo pkill -USR1 -n -x xz`

img格式镜像烧录命令如下（亲测成功）

```
sudo dd bs=4M if=2013-09-25-wheezy-raspbian.img of=/dev/sdb
```

查看烧录进度 `sudo pkill -USR1 -n -x dd`

卸载软件

1. 卸载但不删除配置

   ```
   apt-get remove packagename
   ```

2. 卸载并删除配置

```
apt-get purge packagename
```

基础命令

原链接

安装软件 `apt-get install softname1 softname2 softname3……`

卸载软件 `apt-get remove softname1 softname2 softname3……`

卸载并清除配置 `apt-get remove –purge softname1`

更新软件信息数据库 `apt-get update`

进行系统升级 `apt-get upgrade`

搜索软件包 `apt-cache search softname1 softname2 softname3……`

安装deb软件包 `dpkg -i xxx.deb`

删除软件包 `dpkg -r xxx.deb`

连同配置文件一起删除 `dpkg -r –purge xxx.deb`

查看软件包信息 `dpkg -info xxx.deb`

查看文件拷贝详情 `dpkg -L xxx.deb`

查看系统中已安装软件包信息 `dpkg -l`

重新配置软件包 `dpkg-reconfigure xxx`

清除所有已删除包的残馀配置文件

```
dpkg -l |grep ^rc|awk '{print $2}' |sudo xargs dpkg -P
```

> dpkg安装的可以用apt卸載，反之亦可。

aptitude

`aptitude update` 更新可用的包列表

`aptitude upgrade` 升级可用的包

`aptitude dist-upgrade` 将系统升级到新的发行版

`aptitude install pkgname` 安装包

`aptitude remove pkgname` 删除包

`aptitude purge pkgname` 删除包及其配置文件

`aptitude search string` 搜索包

`aptitude show pkgname` 显示包的详细信息

`aptitude clean` 删除下载的包文件

`aptitude autoclean` 仅删除过期的包文件

Termux

访问手机存储

```
termux-setup-storage
```

执行上面的命令以后，会跳出一个对话框，询问是否允许 Termux 访问手机存储，点击"允许"。

创建软链

直接跳转到手机内存卡对应目录的快捷方式

```
ln -s /data/data/com.termux/files/home/storage/shared/wsj0051 wsj0051
```

修改为清华源

使用如下命令行替换官方源为 TUNA 镜像源

```
sed -i 's@^\(deb.*stable main\)$@#\1\ndeb https://mirrors.tuna.tsinghua.edu.cn/termux/
sed -i 's@^\(deb.*games stable\)$@#\1\ndeb https://mirrors.tuna.tsinghua.edu.cn/termux
sed -i 's@^\(deb.*science stable\)$@#\1\ndeb https://mirrors.tuna.tsinghua.edu.cn/term
apt update && apt upgrade
```

安装基本工具

```
pkg update
pkg upgrade
pkg install vim curl wget git unzip unrar
```

修改终端配色

```
sh -c "$(curl -fsSL https://github.com/Cabbagec/termux-ohmyzsh/raw/master/install.sh)"
```

脚本运行后会提示选择背景色和字体

```
Enter a number, leave blank to not to change: 14
Enter a number, leave blank to not to change: 6
```

如果需要重新修改配色

```
~/termux-ohmyzsh/install.sh
```

Termux快捷按键

```
mkdir $HOME/.termux;
echo "extra-keys = [ \
    ['ESC','<','>','BACKSLASH','=','^','$','()','{}','[]','ENTER'], \
    ['TAB','&',';','/','~','%','*','HOME','UP','END','PGUP'], \
    ['CTRL','FN','ALT','|','-','+','QUOTE','LEFT','DOWN','RIGHT','PGDN'] \
    ]
" >> $HOME/.termux/termux.properties
```

修改启动问候语

```
vim $PREFIX/etc/motd
```

如果没有安装vim的话会有提示，根据提示安装：`pkg install vim`

修改启动语为sh脚本方式

```
cd $PREFIX/etc
vim motd
```

motd内容修改为脚本文件，内容为：

```
#!$PREFIX/bin/bash
neofetch
```

修改后保存并退出，执行以下命令

```
mv motd profile.d/motd.sh
```

重复执行问题

如果启动后出现触发两次，将sh文件执行语句放进.zshrc下

```
mv $PREFIX/etc/profile.d/motd.sh .
echo "$PREFIX/bin/bash ~/motd.sh" >> ~/.zshrc
```

修改neofetch配置

```
cd .config/neofetch
vim config.conf
```

可以修改展示的信息，颜色，修改 `ascii_distro="linux"` 将默认的安卓换为linux

搭建Hexo 博客

```
pkg install nodejs
pkg install git
npm install hexo-cli -g
mkdir hexoblog
cd hexoblog
hexo init
```

> 更多资料参考官方api:https://hexo.io/zh-cn/docs/index.html

git

配置全局用户名邮箱信息

```
git config --global user.name  "your user name"
git config --global user.email "your email"
```

让Git显示颜色

```
git config --global color.ui true
```

git设置别名

```
git config --global alias.lg "log --color --graph --pretty=format:'%Cred%h%Creset -%C(
```

Git在clone仓库时，有两种URL可以选择，分别为HTTPS和SSH：

1. HTTPS的格式为： `https://github.com/用户名/仓库名.git`

2. SSH的格式为： `git@github.com:用户名/仓库名.git`

git图形化工具sourcetree

ssh生成密钥

```
ssh-keygen -t rsa -C "your email"
```

将公钥配置到github后,验证公钥是否绑定成功

```
ssh -T git@github.com
```

npm安装http-server

```
npm install -g http-server
```

然后，运行 Server。

```
http-server
```

正常情况下，命令行会提示 Server 已经在 8080 端口运行了，并且还会提示外部可以访问的 IP 地址。

使用ecj termux-tools dx编译java文件

1. 更新资源

```
pkg update & pkg upgrade
```

2. 安装所需软件

```
pkg install ecj termux-tools dx
```

3. 创建java文件Hello.java

```
public class Hello{
   public static void main(String[] args){
    System.out.println("Hello World!");
   }
}
```

4. 编译java文件

```
ecj Hello.java
```

5. 生成安卓虚拟机文件

```
dx --dex --output=Hello.dex Hello.class
```

6. 安卓虚拟机运行程序

```
dalvikvm -cp Hello.dex Hello
```

7. 更简单方式，创建shell脚本 `vim ecj.sh`

```
#!/usr/bin/sh
ecj "$1.java"
dx --dex --output="$1.dex" "$1.class"
dalvikvm -cp "$1.dex" "$1"
```

8. 执行shell编译java

```
sh ecj.sh Hello
```

手机通知栏时间打开时分秒

手机通知栏的时间没有精确到秒，手机root后可以打开

1. 使用一键方式安装adb工具
2. 执行以下命令：

```
pkg install tsu
adb shell
settings put secure clock_seconds 1
```

使用atilo安装linux

1. 在Termux安装Linux的bash脚本

```
echo "deb [trusted=yes] https://yadominjinta.github.io/files/ termux extras" >>
pkg in atilo-cn
```

2. 安装debian

```
atilo pull debian
```

3. 运行debian

```
atilo run debian
```

配置apache java环境

1. 在usr/local下创建java文件夹，将下载好的tar文件移动到java目录下，对应手机目录为 `/data/data/com.termux/files/home/.atilo/debian/usr/local/java`
2. 解压jdk和tomcat

```
cd /usr/local/java
tar xzf jdk-8u241-linux-arm64-vfp-hflt.tar.gz
tar xzf apache-tomcat-9.0.31.tar.gz
```

3. 以下内容拷进 `/etc/profile` 文件末尾

```
JAVA_HOME=/usr/local/java/jdk1.8.0_241
PATH=$JAVA_HOME/bin:$PATH
CLASSPATH=$JAVA_HOME/jre/lib/ext:$JAVA_HOME/lib/tools.jar
export PATH JAVA_HOME CLASSPATH
```

4. 保存退出后执行以下命令让配置生效，使用 `java -version` 查看环境变量是否配置成功

```
 source /etc/profile
```

一键脚本

1. 一键安装tmoe-linux

```
 . <(curl -L gitee.com/mo2/linux/raw/2/2)
```

2. 仅美化termux界面

```
. <(curl -L gitee.com/mo2/zsh/raw/2/2)
```

参考链接

1. 国光-Termux
2. 简书
3. tmoe

其他

Jenkins

- 插件下载地址
  - hpi插件地址
  - Theme插件
- 页面自定义 修改jar包 `jenkins/WEB-INF/lib/jenkins-core-1.651.3.jar` 文件中的 `lib/layout/layout.jelly`

手机

火狐插件安装链接

Github加速

- 油猴脚本工具
  - 加速访问脚本地址：https://greasyfork.org/zh-CN/scripts/397419
  - 加速下载脚本脚本地址：https://greasyfork.org/scripts/412245
  - Github 增强 - 高速下载: https://greasyfork.org/zh-CN/scripts/412245-github-%E5%A2%9E%E5%BC%BA-%E9%AB%98%E9%80%9F%E4%B8%8B%E8%BD%BD
- 一键获取Github文件加速下载地址（浏览器扩展）
  - 扩展官网：https://fhefh2015.github.io/Fast-GitHub/
  - Chrome安装地址：https://chrome.google.com/webstore/detail/mfnkflidjnladnkldfonnaicljppahpg
  - 国内可访问Chrome安装地址1：https://chrome.zzzmh.cn/info?token=mfnkflidjnladnkldfonnaicljppahpg
  - 国内可访问Chrome安装地址2：https://www.extfans.com/productivity/mfnkflidjnladnkldfonnaicljppahpg/
  - Edge安装地址：https://microsoftedge.microsoft.com/addons/detail/ljceflkaahacpphaioldeledefadpmdp

网址

- git
  - github

    - gitee
    - gitlab
    - coding
    - github代理加速
- 博客论坛
    - csdn
    - 开源中国
    - 掘金
    - V2EX
    - segmentfault
    - 语雀
- 部署
    - glitch
    - vercel
- 邮箱
    - 联通邮箱
    - 163邮箱
    - 126邮箱
    - qq邮箱
    - 新浪邮箱
    - Outlook邮箱
    - 139邮箱
- 主机域名
    - Freenom域名
    - Vultr
    - 腾讯云
    - Cloudflare
- 论坛类
    - 知乎
    - 简书
    - 微博
    - 领英
    - iteye
    - 豆瓣
    - 少数派
- 云盘
    - 蓝奏云
    - 天翼云盘
    - 百度云
    - sync
- 下载
    - 迅雷
    - 音乐磁场
    - 磁力搜索
- 通知留言
    - Leancloud
    - Server酱
    - WxPush

- 账号登录
  - 360账号
  - 火狐
- 装机
  - MSDN
  - Potplayer播放器
  - 微pe工具箱
  - 下载office
- 其他
  - Chrome Ctx下载
  - Google Play下载
  - 163study
  - 油猴脚本
  - 52破解
  - 假52