|  |  |
| --- | --- |
| 快捷键 | 用途 |
| open {filename} | 终端上打开某个文件 |
| control+command+Q | 锁屏 |
| option+command+esc | 强制杀应用 |

**HTTP/1.0**

除了GET命令，还引入了POST命令和HEAD命令，丰富了浏览器与服务器的互动手段.

GET /mypage.html HTTP/1.0 User-Agent: NCSA\_Mosaic/2.0 (Windows 3.1) 200 OK Date: Tue, 15 Nov 1994 08:12:31 GMT Server: CERN/3.0 libwww/2.17 Content-Type: text/html <HTML> 一个包含图片的页面 <IMG SRC="/myimage.gif"> </HTML>

头信息第一行是”协议版本 + 状态码（status code） + 状态描述”。

**Content-Type 字段**

text/html text/css image/png image/gif application/javascript application/octet-stream

**Content-Encoding字段**

content-encoding: gzip Content-Encoding: compress Content-Encoding: deflate

**HTTP/1.0的缺点**

每个TCP连接只能发送一个请求。发送数据完毕，连接就关闭，如果还要请求其他资源，就必须再新建一个连接。TCP连接的新建成本很高，因为需要客户端和服务器三次握手，并且开始时发送速率较慢（slow start）。

**引入非标准的Connection字段**

Connection: keep-alive

该字段要求服务器不要关闭TCP连接，以便其他请求复用，直到客户端或服务端主动关闭连接。但是，这不是标准字段，不同实现的行为可能不一致，因此不是根本的解决办法。

**HTTP/1.1**

**持久连接**

默认TCP连接不关闭，可以被多个请求复用。

关闭：规范的做法，服务端在最后一个请求时，发送 **Connection: close**，明确要求服务器关闭TCP连接。

同一个域名，一般可以同时建立 **6** 个持久连接。

**管道机制（**pipelining**）**

在同一个TCP连接里面，客户端可以同时发送多个请求。这样就进一步改进了HTTP协议的效率。

**Content-Length 字段**

一个TCP连接现在可以传送多个回应，势必就要有一种机制，区分数据包是属于哪一个回应的。这就是Content-length字段的作用，声明本次回应的数据长度。

**分块传输编码 （Transfer-Encoding）**

使用 Content-Length 字段的前提条件是，服务器发送回应之前，必须知道回应的数据长度。

对于一些很耗时的动态操作来说，这意味着，服务器要等到所有操作完成，才能发送数据，显然这样的效率不高。

Transfer-Encoding: chunked

1.1版规定可以不使用 Content-Length 字段，而使用”分块传输编码”（chunked transfer encoding）。只要请求或回应的头信息有Transfer-Encoding字段，就表明回应将由数量未定的数据块组成.

每个非空的数据块之前，会有一个16进制的数值，表示这个块的长度。最后是一个大小为0的块，就表示本次回应的数据发送完了。

**其他变化**

增了许多动词方法 PUT、PATCH、HEAD、 OPTIONS、DELETE 。另外，客户端请求的头信息新增了 HOST 字段，用来指定服务器的域名。

**缺点**

虽然1.1版允许复用TCP连接，但是同一个TCP连接里面，所有的数据通信是按次序进行的。服务器只有处理完一个回应，才会进行下一个回应。要是前面的回应特别慢，后面就会有许多请求排队等着。这称为”队头堵塞”（Head-of-line blocking）。

为了避免这个问题，只有两种方法：一是减少请求数，二是同时多开持久连接。这导致了很多的网页优化技巧，比如合并脚本和样式表、将图片嵌入CSS代码、域名分片（domain sharding）等等。如果HTTP协议设计得更好一些，这些额外的工作是可以避免的。