2.4 各种令人头疼的字符

URL 是*可移植的*（portable）。它要统一地命名因特网上所有的资源，这也就意味着要通过各种不同的协议来传送这些资源。这些协议在传输数据时都会使用不同的机制，所以，设计 URL，使其可以通过任意因特网协议安全地传输是很重要的。

安全传输意味着 URL 的传输不能丢失信息。有些协议，比如传输电子邮件的简单邮件传输协议（Simple Mail Transfer Protocol，SMTP），所使用的传输方法就会剥去一些特定的字符。[4] 为了避开这些问题，URL 只能使用一些相对较小的、通用的安全字母表中的字符。

除了希望 URL 可以被所有因特网协议进行传送之外，设计者们还希望 URL 是*可读的*。因此，即使不可见、不可打印的字符能够穿过邮件程序，从而成为可移植的，也不能在 URL 中使用。[5]

URL 还得是*完整的*，这就使问题变得更加复杂了。URL 的设计者们认识到有时人们可能会希望 URL 中包含除通用的安全字母表之外的二进制数据或字符。因此，需要有一种转义机制，能够将不安全的字符编码为安全字符，再进行传输。

本节总结了 URL 的通用字母表和编码规则。

2.4.1 URL字符集

默认的计算机系统字符集通常都倾向于以英语为中心。从历史上来看，很多计算机应用程序使用的都是 US-ASCII 字符集。US-ASCII 使用 7 位二进制码来表示英文打字机提供的大多数按键和少数用于文本格式和硬件通知的不可打印控制字符。

由于 US-ASCII 的历史悠久，所以其可移植性很好。但是，虽然美国用户使用起来很便捷，它却并不支持在各种欧洲语言或全世界数十亿人使用的数百种非罗马语言
35 中很常见的变体字符。

而且，有些 URL 中还会包含任意的二进制数据。认识到对完整性的需求之后，URL 的设计者就将*转义序列*集成了进去。通过转义序列，就可以用 US-ASCII 字符集的有限子集对任意字符值或数据进行编码了，这样就实现了可移植性和完整性。

2.4.2 编码机制

为了避开安全字符集表示法带来的限制，人们设计了一种编码机制，用来在 URL 中表示各种不安全的字符。这种编码机制就是通过一种“转义”表示法来表示不安全

注 4：这是由报文的 7 位二进制码编码造成的。如果源端是以 8 位或更多位编码的，就会有部分信息被剥离。
注 5：不可打印字符中包括空格符（注意，RFC 2396 建议应用程序忽略空格符）。