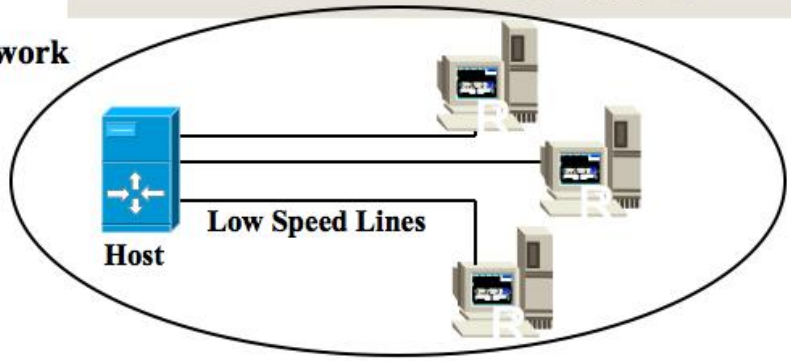
第三章 因特网汉字信息交换技术

内容

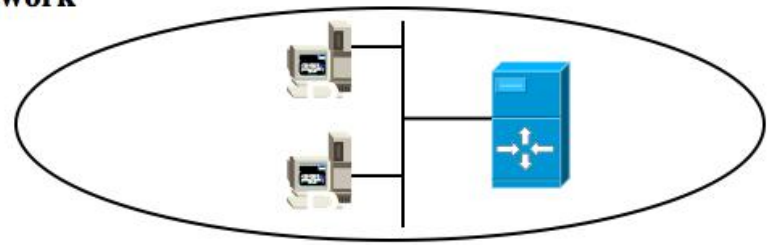
- **概述**
- 二进制文件编码方法
- 常用Internet编码方法
- Unicode实现方式

网络的演进

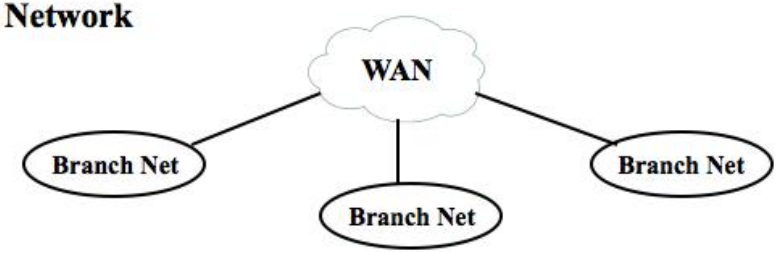
简单的联接 1960-1970



网络化联接 1970-1980



网络间互连 1980-1990



早期主要应用：
E-mail

3.1 概述

- 早期
  - ASCII码：网络传输文字编码，7-bit
  - 网络：7-bit数据通路
- 问题
  - 汉字信息通常用8-bit表示
  - 如何在7-bit数据通路上传输8-bit数据

3.1 概述

- 解决方法：
  - 对汉字机内码或交换码*进行编码*
  - 8-bit -> 7bit
- 缺点
  - 浪费空间
- 各种方法的差异
  - 如何编码和解码

内 容

- 概述
- 二进制文件编码方法
- 常用Internet编码方法
- Unicode实现方式

3.2 ROT 13

- 最简单的编码

- 根据字母顺序用在13位之后的对应字母替代。

3.2.1 Uuencode

- 将二进制文件以文本文件方式编码
  - 方便基于文本传输环境中传输。
  - 邮件系统/二进制新闻组中使用频率较高。
  - 常用于 Attach 二进制文件 。

3.2.2 Uuencode 编码特点

- 特征：
  - 第一行为：begin xxx（数字） 文件名
  - 每一行开头用“M”标志
  - 最后一行为：end
- 例子

```
* 除了记忆什么都不带走，除了足迹什么都不留下*
**********************************
```

3.2.3 转换方法

- 把它单独存成一个文件：test.uue
- 然后用Winzip打开解压
- 得到Test.txt文件

3.2.4 编码算法

- 将3个字符顺序放入一个24 位的缓冲区，
  - 缺字符的地方补零
- 将缓冲区截断成为4个部分
  - 高位在先，每个部分 6 位(bit)
- 用下面的64个字符重新表示：

  `!”#$%&‘()*+,-./0123456789:;<=>?@

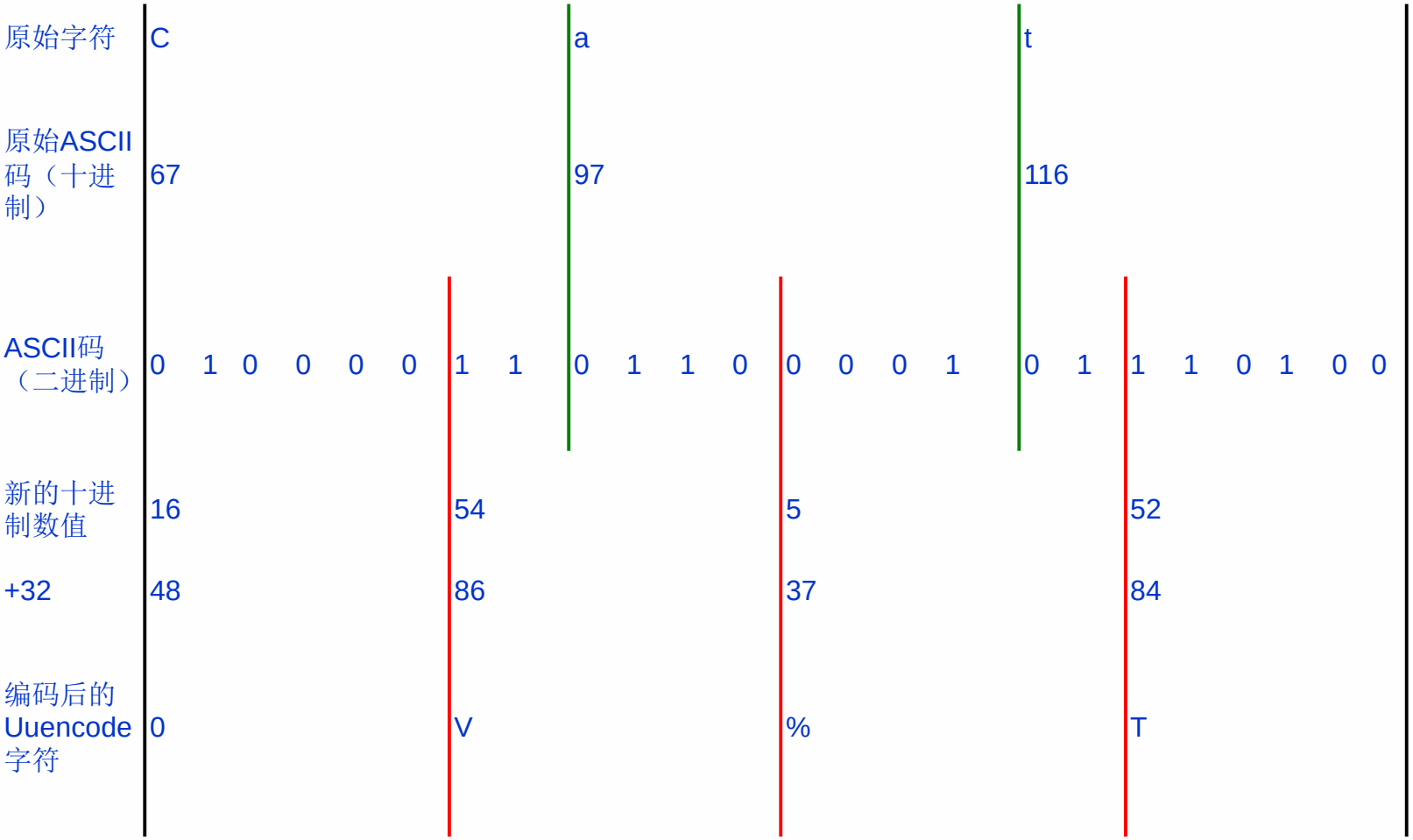
  ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_

3.2.5 编码对照表

可打印的字符	十进制ASCII值	uuencode二进制表示	uuencode十进制表示
(space)	32	000 000	0
!	33	000 001	1
"	34	000 010	2
#	35	000 011	3
$	36	000 100	4
%	37	000 101	5
&	38	000 110	6
'	39	000 111	7
(	40	001 000	8
)	41	001 001	9
*	42	001 010	10
+	43	001 011	11
,	44	001 100	12
-	45	001 101	13
.	46	001 110	14
/	47	001 111	15
0	48	010 000	16
1	49	010 001	17
2	50	010 010	18
3	51	010 011	19
4	52	010 100	20
5	53	010 101	21
6	54	010 110	22
7	55	010 111	23
8	56	011 000	24
9	57	011 001	25
:	58	011 010	26
;	59	011 011	27
<	60	011 100	28
=	61	011 101	29
>	62	011 110	30
?	63	011 111	31

可打印的字符	十进制ASCII值	uuencode二进制表示	uuencode十进制表示
@	64	100 000	32
A	65	100 001	33
B	66	100 010	34
C	67	100 011	35
D	68	100 100	36
E	69	100 101	37
F	70	100 110	38
G	71	100 111	39
H	72	101 000	40
I	73	101 001	41
J	74	101 010	42
K	75	101 011	43
L	76	101 100	44
M	77	101 101	45
N	78	101 110	46
O	79	101 111	47
P	80	110 000	48
Q	81	110 001	49
R	82	110 010	50
S	83	110 011	51
T	84	110 100	52
U	85	110 101	53
V	86	110 110	54
W	87	110 111	55
X	88	111 000	56
Y	89	111 001	57
Z	90	111 010	58
[	91	111 011	59
\	92	111 100	60
]	93	111 101	61
^	94	111 110	62
_	95	111 111	63

3.2.6 例子


原始字符
C
a
t
原始ASCII码（十进制）
67
97
116
ASCII码（二进制）
0 1 0 0 0 0 1 1
0 1 1 0 0 0 0 1
0 1 1 1 0 1 0 0
新的十进制数值
16
54
5
52
+32
48
86
37
84
编码后的Uuencode字符
0
V
%
T

3.2.7 编码实现

- 每次读取源文件的45个字符
  - 不足45用“NULL”补足3的整数倍
  - 如：23补为24
- 输入目标文件一个ASCII为：
  - “32+实际读取字符数” 作为每一行开始
    =77 (M)
- 读取的字符编码后输入目标文件，再输入一个“换行符”。
- 编码完成，输入“`（ASCII为96）”和一个“换行符”表示结束。

3.2.8 Xxencode

- 和Uuencode相似
- 编码字符不同：
  - +-0123456789
  - ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
- 以‘h’为一行的首字符。

内 容

- 概述
- 二进制文件编码方法
- **常用Internet编码方法**
- Unicode实现方式

3.3 从SMTP到 MIME

- SMTP：简单的邮件传输协议（文本信息）
- MIME：多用途互联网邮件扩展 (Multipurpose Internet Mail Extensions)，并没有改变SMTP或取代，增加了定义传输非ASCII码的编码规则和邮件主题的结构等。
- 定义在RFC 2045、RFC 2046、RFC 2047、RFC 2048、RFC 2049等RFC中。

3.3.1 Base 64

- Base64属于MIME
  - 最常见传输8-bit字节编码方式。
  - 几乎所有的电子邮件软件头把它作为默认的二进制编码。
- MIME格式：
  - MIME-Version: 1.0
  - Content-Type: text/plain;
  - charset="gb2312"
  - Content-Transfer-Encoding: base64

3.3.2 例子

- MIME-Version: 1.0
- Content-Type: text/plain;
- charset="gb2312"
- Content-Transfer-Encoding: base64

我国不仅是世界上最大的水稻生产国和消费国，更是稻作历史最悠久、水稻遗传资源最丰富的国家之一。考古证明，早在7000多年前，我国就开始种植水稻。

- ztK5+rK7vfbKx8rAvefJz9futPO1xMuutb7J+rL6ufq6zc/7t9G5+qOsuPzKx7W+1/fA+sq31+7T
- xr7DoaLLrrW+0sW0q9fK1LTX7rfhuLu1xLn6vNLWrtK7oaO/vLnF1qTD96Os1OfU2jcwMDC24MTq
- x7CjrM7Sufq+zb+qyrzW1tayy661vqGjDQo=
- 把它单独存成一个文件，可以取名为：XXX.eml
  双击可以用OutLook打开

3.3.3 编码算法

- 顺序放入 24 位缓冲区
  - 缺字符的地方补零
- 将缓冲区截断成为 4 个部分
  - 高位在先，每个部分 6 位(bit)
- 用下面的64个字符重新表示：
  - ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz0123456789+/”。

3.3.3 编码算法

- 如果输入不足3个字符
  - 输出将用等号“=”补足
  - 可以隔断附加的信息造成编码的混乱
  - 每行一般为76个字符。

3.3.4 索引表

索引	对应字符	索引	对应字符	索引	对应字符	索引	对应字符
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w		
15	P	32	g	49	x		
16	Q	33	h	50	y		

3.3.5 例子

转换前	1010 1101 *1011 1010* 0111 0110(24bits)			
转换后	00101011	00011011	00101001	00110110
十进制	43	27	41	54
对应码表中的值	r	b	p	2

3.4 Quoted-Printable

- 简称QP
  - 一般用在mail系统
  - 专门为了处理8位字符制定
  - 用于少量文本的8位字符编码
  - 例如：Foxmail对主题和信体编码
  - 特点：大量“=”
- 算法最简单，编码效率最低
  - 编码率1:3

3.4.1 算法

- 读一个字符
  - 如果ASCII码大于127，进行编码(分成3个字符，1个等于号和1个16进制的数字）
  - 否则忽略

例子：

- 文字：美　　国　　在　　线　A O L
- 内码：C3C0 B9FA　D4DA CFDF 41 4F 4C
- QP：=C3=C0 =B9=FA =D4=DA =CF=DF A O L
- 内码：3D 43 33 3D 43 30 ……

3.4.2 例子

日经BP社报道】美国在线时代华纳的美国在线(AOL)公司于当地时间9月12日公布了新的董事任命。AOL将废除COO(首席运营官)及总经理职务，建立由该公司的董事长兼首席行政官(CEO)Jon Miller更直接监管运营Interactive Marketing集团及AOL宽带服务集团等主要业务的机制。

3.4.2 例子

- MIME-Version: 1.0
- Content-Type: text/plain;
- charset="gb2312"
- Content-Transfer-Encoding: quoted-printable
- X-Priority: 3
- X-MSMail-Priority: Normal
- X-Unsent: 1
- X-MimeOLE: Produced By Microsoft MimeOLE V5.00.2615.200

- =C8=D5=BE=ADBP=C9=E7=B1=A8=B5=C0=A1=BF=C3=C0=B9=FA=D4=DA=CF=DF=CA=B1=B4=FA=
- =BB=AA=C4=C9=B5=C4=C3=C0=B9=FA=D4=DA=CF=DF(AOL)=B9=AB=CB=BE=D3=DA=B5=B1=B5=
- =D8=CA=B1=BC=E49=D4=C212=C8=D5=B9=AB=B2=BC=C1=CB=D0=C2=B5=C4=B6=AD=CA=C2=C8=
- =CE=C3=FC=A1=A3AOL=BD=AB=B7=CF=B3=FDCOO(=CA=D7=CF=AF=D4=CB=D3=AA=B9=D9)=BC=
- =B0=D7=DC=BE=AD=C0=ED=D6=B0=CE=F1=A3=AC=BD=A8=C1=A2=D3=C9=B8=C3=B9=AB=CB=BE
  =
- =B5=C4=B6=AD=CA=C2=B3=A4=BC=E6=CA=D7=CF=AF=D0=D0=D5=FE=B9=D9(CEO)Jon =
- Miller=B8=FC=D6=B1=BD=D3=BC=E0=B9=DC=D4=CB=D3=AAInteractive =
- Marketing=BC=AF=CD=C5=BC=B0AOL=BF=ED=B4=F8=B7=FE=CE=F1=BC=AF=CD=C5=B5=C8=D6=
- =F7=D2=AA=D2=B5=CE=F1=B5=C4=BB=FA=D6=C6=A1=A3
- =20

3.5 HZ编码

- 中文常用编码
- 和QP一样，只能对文本进行编码
- 编码时忽略控制字符。
- 特点：有许多“~{”和“~}”，总是成对出现 。

3.5.1 编码算法

- 编码：
  - 读一个字符，如果是8位字符，把最高位清零
  - 把连续的高位清零后字符用“~{”和“~}”括起来。
- 解码：
  - 把用“~{”和“~}”括起来每个字符第8位置“1”

3.5.2 HZ编码举例

- …
- MIME-Version: 1.0
- Content-Type: text/plain;
- charset="hz-gb-2312"
- Content-Transfer-Encoding: 7bit
- …
- X-MimeOLE: Produced By Microsoft

- ~{5gWSSJ<~:MPBNEWi~}
- ~{6`8vUJ;':M1jJ6~}
- HTML ~{SJ<~V'3V~}
- ~{M(Q62>:MD?B<7~Nq~}
- ~{MQ;zM,2=~}
- ~{8D=x5D!0JU<~Od!19fTr~}

电子邮件和新闻组
多个帐户和标识
HTML 邮件支持
通讯簿和目录服务
脱机同步
改进的“收件箱”规则

1	**0**	**1**	**1**	**0**	**1**	**0**	**1**	**“电”高位字节**
0	0	1	1	0	1	0	1	35H “5”
1	**1**	**1**	**0**	**0**	**1**	**1**	**1**	**“电”低位字节**
0	1	1	0	0	1	1	1	67H g

内容

- 概述
- 二进制文件编码方法
- 常用Internet编码方法
- **Unicode实现方式**

3.4 Unicode实现方式

- 不同于编码方式。
- 一个字符的Unicode编码是确定的。
- 但是在实际传输过程中：
  - 不同系统平台的设计不一定一致
  - 出于节省空间的目的
  - 导致Unicode编码的实现方式不同
- Unicode的实现方式
  - UTF- Unicode Transformation Format

3.4.1 常用UTF

- UTF-7
  - RFC 1642，A Mail-Safe Transformation Format of Unicode
- UTF-8
  - RFC 3629， 8-bit Unicode Transformation Format
- UTF-16
  - ISO/IEC 10646-1的附录C，RFC2781
- UTF-32

3.4.2 UTF-7

- 7位交换码
- A Mail-Safe Transformation Format of Unicode
- 背景：
  - Unicode 并无法在Internet上传输
  - 为ISO10646(Unicode)在Internet上传输而设计
- 用ASCII码对Unicode文本进行转换。

3.4.3 编码

- 类似Base 64
- 定义不同字符集
- 针对不同字符集，采用不同的编码方法

3.4.4 字符集定义

- 直接编码字符集(Set D)：
  - ‘ ( ) , - . / : ?
  - ABCDEFGHIJKLMNOPQRSTUVWXYZ
  - abcdefghijklmnopqrstuvwxyz
  - 0123456789
- 可选择直接编码字符集(Set O)：
  - ! ” # $ % & * ; < = > @ [ ] ^ { | }
  - Set O中的字符可以根据系统需要选择
- 把ASCII字符中的 “+”作为特殊字符

3.4.5 编码方法

- 对Unicode文本中
  - 属于Set D和Set O内的字符不编码
  - 直接用其ASCII码表示（即去除其高8位00H）
- 字符“+”，用“+-”的ASCII码来表示
- 其它字符：用BASE64来编码
  - 在被编码的字符串首加上“+”作标记
  - 在字符串末加上“-”作标记
  - UTF-7编码文本中，正文每一行均由“+”开头，并由“-”结尾

3.4.6 例子

- MIME-Version: 1.0
- Content-Type: text/plain;
- charset="utf-7"
- Content-Transfer-Encoding: 7bit
- X-Priority: 3
- X-MSMail-Priority: Normal
- X-Unsent: 1
- X-MimeOLE:

- +VBtODYhMUW5ZN3M2/wyOR4qwdVlRbk4tbTL/Hw-
- +f46JgXcHUW5bnE/u/wxsm1Q+TlhRbmhCgh8wAg-

君不行兮夷猶，蹇誰留兮中洲？
美要眇兮宜修，沛吾乘兮桂舟。

3.4.7 解码方法

- 括在“+”和“-”之间部分。
  - 采用与BASE64同样的解码方法
- “+-”：用“+”的Unicode编码表示
- 其它字符（即Set D和Set O内的字符）：
  - 在其前面加上00h字节，从而形成其相应的Unicode编码。

3.4.8 UTF-8

- 专门为单字节处理系统设计
  - Unix
  - 解决UNIX对ASCII字符支持
  - 避免直接使用Unicode
    - 特殊字符的混淆
    - 如“/”、NULL(00h)等在UNIX内具有特殊含义
  - 在网络上，UTF-8常用于UNIX邮件服务器
- Unix，Linux，IOS，Andriod均采用UTF-8作为内码。

3.4.9 编码

- Unicode二进制流
  - 按照8位代码进行编码
  - 变长编码
  - 编码长度为1—6字节
- 码值为：
  - 0000h—007Fh： 单字节
  - 0080h — 007FFh: 双字节
  - 0800h — FFFFh: 三字节
  - 010000h — 1FFFFFh: 四字节
  - 00200000h — 03FFFFFFh: 五字节
  - 04000000h — 7FFFFFFFh: 六字节

3.4.10 UTF-8 编码方法

- 00000000h – 0000007Fh: 0xxxxxxx
- 00000080h – 000007FFh: 110xxxxx 10xxxxxx
- 00000800h – 0000FFFFh: 1110xxxx 10xxxxxx 10xxxxxx
- 00010000h – 001FFFFFh: 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
- 00200000h – 03FFFFFFh: 111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx
- 04000000h – 7FFFFFFFh: 1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx

- UTF-8 编码示例
- Unicode 字符版权标记字符 00A9h = 0000 0000 1010 1001 用 UTF-8 编码如下所示：
- 11000010 10101001 = C2h A9h

- “不等于”符号字符 2260h = 0010 0010 0110 0000 编码如下所示：
- 11100010 10001001 10100000 = E2h 89h A0h

3.4.11 UTF-8的特征

- UTF-8码：
  - ASCII码：单字节ASCII码（最高位为“0”）
  - 非ASCII符(大于7Fh者)：多字节不等长编码
    - 首字节为C0h－FDh
    - 首字节高位所含“1”的个数，就是该码的码长(字节数)
    - 首字节外的其它字节为80h－BFh；
- 如果仅使用16位的Unicode(UCS-2)，则UTF-8码的码长为1-3个字节
  - 汉字（3400h－9FFFh）码长均为3个字节。

3.4.12 例子

```
EF BB BF 33 E6 9C 88 31 35 E6 97 A5 EF BC 8C E5
BE 90 E5 85 88 E7 94 9F E5 9B A0 E6 90 BA E5 B8
A6 33 E7 BD 90 E5 A5 B6 E7 B1 B3 E7 B2 89 E8 BF
87 E5 85 B3 EF BC 8C E8 A2 AB E9 A6 99 E6 B8 AF
E6 B5 B7 E5 85 B3 E8 AF AF E5 88 A4 E8 80 8C E9
81 AD E5 88 B0 E6 9F A5 E6 89 A3 EF BC 8C E5 B9
B6 E8 A2 AB E8 A6 81 E6 B1 82 E4 BA A4 E5 87 BA
33 30 30 30 E5 85 83 E4 BF 9D E9 87 8A E9 87 91
E6 89 8D E8 83 BD E8 BF 87 E5 85 B3 E8 BF 94 E5
9B 9E E6 B7 B1 E5 9C B3 E3 80 82 E6 8D AE E6 82
89 EF BC 8C E9 A6 99 E6 B8 AF E6 B5 B7 E5 85 B3
E5 BD 93 E6 97 B6 E8 BF 98 E8 A6 81 E6 B1 82 E4
BB 96 E5 9C A8 35 E6 9C 88 37 E6 97 A5 E5 88 B0
E9 A6 99 E6 B8 AF E8 AD A6 E7 BD B2 E6 8A A5 E5
88 B0 EF BC 8C 35 E6 9C 88 39 E6 97 A5 E5 87 BA
E5 BA AD E5 BA 94 E8 AE AF E3 80 82
```

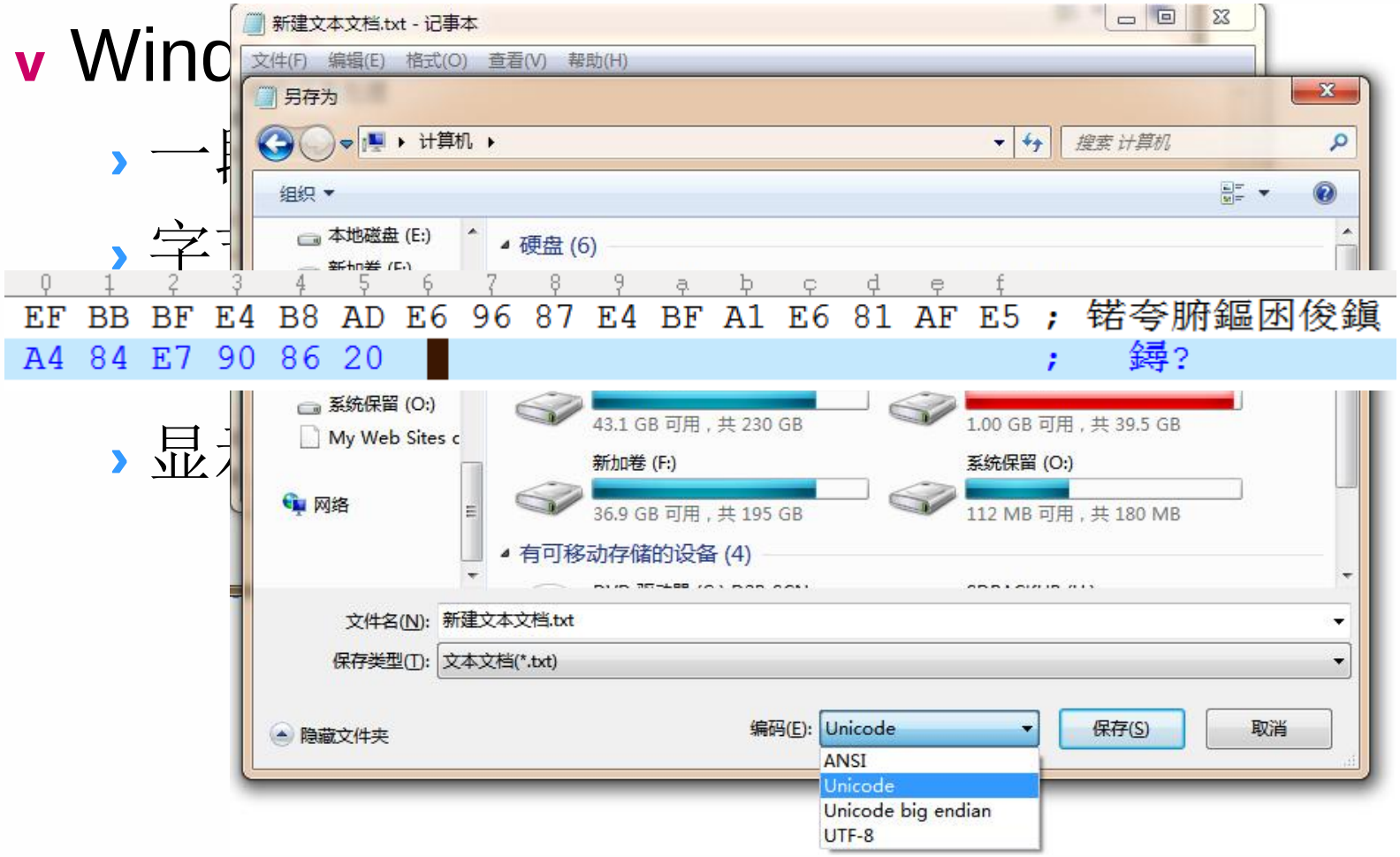
3月15日，徐先生因携带3罐奶米粉过关，被香港海关误判而遭到查扣，并被要求交出3000元保释金才能过关返回深圳。据悉，香港海关当时还要求他在5月7日到香港警署报到，5月9日出庭应讯。

3.4.13 优点和缺点

- 优点
  - ASCII是UTF-8的一个子集
  - 任何面向字节的字符串搜索算法都可以用于UTF-8的数据
  - UTF-8字符串可由一个简单算法可靠地识别出来
- 缺点
  - 不利于正则表达式检索

3.4.14 应用-Windows

- Wind
  - 一
  - 字
  - 显



3.4.15 UTF-16

- 汉字
  - BMP内：近3万个
  - BMP外：CJK表意文字扩充B区，共计42807个汉字
- ISO10646使用20位编码
- 32位编码？
  - 若仅使用BMP，则字符可用16位UCS-2编码
  - 若超出BMP，则字符要用32位UCS-4编码

UTF-16

- 解决方法：
- UTF-16在使用超出BMP一定范围（即BMP之外的前16个字面内）时：
  - BMP内字符仍用16位编码
  - BMP外的字符采用32位编码
  - 提高编码效率

3.4.17编码原理

- 利用BMP中的代理区
  - 编码扩充到100多万个
  - 扩充码位与00组字面内的1－16号字面的码位相对应
  - 通过代理区共可扩充
    - 917504个辅助码位
    - 131072个专用码位
- BMP内的UTF-16编码与UCS-2编码一样
- 超出BMP的前16个字面内的UFT-16编码则使用

3.4.18 替代对机制

- 替代对 （Surrogate Pair）
  - 采用两个16位编码来表示一个字符
  - 共32位
  - 替代对的两个编码的取值范围：
    - D800h—DBFFh
    - DC00h—DFFFh
  - 编码空间为$2^{10}\times 2^{10}=2^{20}$
  - 相当于16个字面

3.4.19 编码方法

- 把编码当作二进制数
  - 小于10000h编码：不变
  - 10000h到10FFFFh间的编码（20位）：
    - 一个范围在D800h到DBFFh的16位整数（高位字）
    - 一个范围在DC00h到DFFFh的整数（低位字）
    - 32位编码
  - 数值超过10FFFFh，则不能编码

3.4.20 转换方法

- 范围为10000h－10FFFFh编码：
  - 设V为此范围内的编码
  - W_1和W_2分别为UTF-16编码高位字(16位)和低位字(16位)
  - V’=V-10000h=yyyyyyyyyyxxxxxxxxxx(二进制形式)
  - W_1=110110yyyyyyyyyy(二进制形式)
  - W_2=110111xxxxxxxxxx(二进制形式)

3.4.21 例子

```
V  = 0x64321
Vx = V - 0x10000
   = 0x54321
   = 0101 0100 0011 0010 0001

Vh = 01 0101 0000 // Vx 的高位部份的 10 bits
Vl = 11 0010 0001 // Vx 的低位部份的 10 bits
w1 = 0xD800 //結果的前16位元初始值
w2 = 0xDC00 //結果的後16位元初始值

w1 = w1 | Vh
   = 1101 1000 0000 0000
   |         01 0101 0000
   = 1101 1001 0101 0000
   = 0xD950

w2 = w2 | Vl
   = 1101 1100 0000 0000
   |         11 0010 0001
   = 1101 1111 0010 0001
   = 0xDF21
```

3.4.22 UCS-4和UTF-16

字面号	UCS-4编码	UTF-16编码
1	00010000-0001FFFF	D800DC00-D83FDFFF
2	00020000-0002FFFF	D840DC00-D87FDFFF
3	00030000-0003FFFF	D880DC00-D8BFDFFF
4	00040000-0004FFFF	D8C0DC00-D8FFDFFF
5	00050000-0005FFFF	D900DC00-D93FDFFF
6	00060000-0006FFFF	D940DC00-D97FDFFF
7	00070000-0007FFFF	D980DC00-D9BFDFFF
8	00080000-0008FFFF	D9C0DC00-D9FFDFFF
9	00090000-0009FFFF	DA00DC00-DA3FDFFF
10	000A0000-000AFFFF	DA40DC00-DA7FDFFF
11	000B0000-000BFFFF	DA80DC00-DABFDFFF
12	000C0000-000CFFFF	DAC0DC00-DAFFDFFF
13	000D0000-000DFFFF	DB00DC00-DB3FDFFF
14	000E0000-000EFFFF	DB40DC00-DB7FDFFF
15	000F0000-000FFFFF	DB80DC00-DBBFDFFF
16	00100000-0010FFFF	DBC0DC00-DBFFDFFF

3.4.23 UTF-32

- 编码特点
  - 4字节等长编码
  - 是UCS-4编码的子集
  - 取值范围为：00000000h—0010FFFFh
  - 取值与相应的UCS-4编码相同
  - UTF-32的编码空间与UTF-16相同
  - UCS-4编码空间的前17个字面（0~16号字面）
- UTF-32编码适用于编码空间超出BMP字面一定范围，并且需要等长编码的情况。

3.4.23 UTF-32

- UTF-32与UCS-4区别:
  - 前者只对0字面组中的0~16号字面编码
  - 后者要对128个字面组所有平面编码
- Unicode和WG2表明:
  - 只会使用0号字面组中的0~16号字面
  - 就是UTF-32的编码空间
  - 可以认为现在UTF-32编码等同于UCS-4编码

作业

- P50. 3-6