U Positivo - UTFPR - PUCPr –
20/05/2024 - 10:31:53.4
Prof Dr P Kantek
(pkantek@gmail.com)
O esquema de codificação Base64
VIVX037a, V: 1.30 76506 CARLOS EDUARDO ANTAO 24CC2202 - 1
SUBSTITUIR e entregar ate
30/maio _____ / _____ / _____

Base64

Base64 é um sistema de numeração posicional que usa 64 como base. Ao produzir um texto que possa ser imprimível em qualquer impressora do mundo ocidental, sem nenhuma chance de erro ou incoerência, apenas podem ser usados os caracteres A..Z, a..z e 0..9. Note que tem-se aqui 26+26+10=62 caracteres. Para completar o conjunto esperado de 64 caracteres, costuma-se usar "+"e "/", nesta ordem. (Note-se que muitos programas que usam este princípio costumam escolher outros 2 caracteres que não sejam "+"e "/", mas nenhum deles usa o rótulo de Base64. São exemplos deste caso: a B64 modificado para URL (usa -"e "_") ou B64 para expressões regulares que usa ("!"e -").

Há ainda um caracter importante, o "="(em Base64) que serve para sinalizar eventuais necessidades de preechimento de *stuffing*.

A idéia desta codificação é poder mandar qualquer conteúdo binário como parte de uma mensagem de e-mail, sem correr o risco de algum conteúdo parcial acabe gerando caracteres de controle do protocolo e ponha a perder a comunicação.

A conversão se dá trocando 3 bytes (8 bits × 3 = 24 bits) em 4 caracteres Base64, cada um deles representado por 6 bits. (então 6 bits × 4 = 24 bits).

Na codificação de 8 bits/byte, usa-se um código qualquer. Atualmente a grande maioria de ambientes usa o padrão ASCII, cuja pequena parte pode ser vista aqui.

uma parte da Tabela ASCII

Binário	Dec	Hex	Car.
0010.1000	40	28	(
0010.1001	41	29	)
0010.1010	42	2A	*
0010.1011	43	2B	+
0010.1100	44	2C	,
0010.1101	45	2D	-
0010.1110	46	2E	.
0010.1111	47	2F	/
0011.0000	48	30	0
0011.0001	49	31	1
0011.0010	50	32	2
0011.0011	51	33	3
0011.0100	52	34	4
0011.0101	53	35	5
0011.0110	54	36	6
0011.0111	55	37	7
0011.1000	56	38	8
0011.1001	57	39	9
0100.0001	65	41	A
0100.0010	66	42	B
0100.0011	67	43	C
0100.0100	68	44	D
0100.0101	69	45	E
0100.0110	70	46	F
0100.0111	71	47	G
0100.1000	72	48	H
0100.1001	73	49	I
0100.1010	74	4A	J
0100.1011	75	4B	K
0100.1100	76	4C	L
0100.1101	77	4D	M
0100.1110	78	4E	N
0100.1111	79	4F	O
0101.0000	80	50	P
0101.0001	81	51	Q
0101.0010	82	52	R
0101.0011	83	53	S
0101.0100	84	54	T
0101.0101	85	55	U
0101.0110	86	56	V
0101.0111	87	57	W
0101.1000	88	58	X
0101.1001	89	59	Y
0101.1010	90	5A	Z
0110.0001	97	61	a
0110.0010	98	62	b
0110.0011	99	63	c
0110.0100	100	64	d
0110.0101	101	65	e
0110.0110	102	66	f
0110.0111	103	67	g
0110.1000	104	68	h
0110.1001	105	69	i
0110.1010	106	6A	j
0110.1011	107	6B	k
0110.1100	108	6C	l
0110.1101	109	6D	m
0110.1110	110	6E	n
0110.1111	111	6F	o
0111.0000	112	70	p
0111.0001	113	71	q
0111.0010	114	72	r
0111.0011	115	73	s
0111.0100	116	74	t
0111.0101	117	75	u
0111.0110	118	76	v
0111.0111	119	77	w
0111.1000	120	78	x
0111.1001	121	79	y
0111.1010	122	7A	z

Para efeito de conversão entre os 2 códigos (ASCII e Base64) deve-se usar o seguinte vetor de conversão:

```
ABCDEFGHIJKLMN
OPQRSTUVWXYZab
cdefghijklmnop
qrstuvwxyz0123
456789+/
```

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

ASCII → Base64

Deve-se pegar 3 bytes a converter (24 bits) e colocá-los lado a lado. Daí, o numerão binário obtido deve ser quebrado em 4 partes, cada uma com 6 bits. Como se sabe, um número binário de 6 bits varia entre 0..63. Este número obtido é que deve indexar o vetor acima estabelecido, dando como resultado 4 caracteres Base64,

Acompanhe no exemplo: Seja passar o ASCII `Aa`) para Base64. Da tabela ASCII, sabe-se que A=0100.0001, a=0110.0001 e)=0010.1001. Juntando tudo, fica: 0100.0001.0110.0001.0010.1001. Rearrumando, fica: 010000.010110.000100.101001, números binários que valem: 16, 22, 4 e 41, respectivamente. Obtendo os elementos 17, 23, 5 e 42 do vetor de elementos do Base64, fica-se com QWEp. Então: $Aa)_{ascii} \equiv QWEp_{Base64}$.

Um cuidado especial deve ser tomado quando se chega ao final do conjunto de caracteres a converter. No fim, podem sobrar nenhum, 1 ou 2 caracteres. Se sobrar nenhum, o problema está resolvido. Se sobrarem 2 caracteres, estes devem ser colocados lado a lado (16 bits) e o conjunto deve ser completado com 2 zeros à direita, formando os 18 bits esperados. Para sinalizar estes 2 *stuffing bits*, após os 3 caracteres Base64 gerados, agrega-se o sinal "=". Por exemplo, seja converter o string ASCII `A1B2C`. O primeiro conjunto `A1B` é convertido normalmente, gerando `QTFC`. Sobra `2C`. Suas configurações em binário são 2=0011.0010 e C=0100.0011. Juntando tudo, fica 0011.0010.0100.0011 cujo comprimento é 16 bits. Agregando 2 zeros à direita e reordenando fica: 0011.0010.0100.0011.00 ou 001100.100100.001100, que correspondem a 12, 36 e 12 respectivamente. Obtendo os elementos 13, 37 e 13 do vetor formador, obtém-se M, k e M respectivamente. A maneira de informar que foram incluídos 2 zeros ao final é colocar um "="como quarto caractere da conversão. Assim: $2C_{ascii} \equiv MkM=_{Base64}$.

Finalmente, se sobrar um único caractere (8 bits), o conversor precisa incluir 4 zeros à direita (*stuffing bits*). Por exemplo, para converter o ASCII "M"para Base64, faz-se: M=0100.1101, com 8 bits. Acrescentando os 4 zeros a direita, fica: 0100.1101.0000 e quebrando em dois números de 6 bits cada, fica: 010011.010000 que são os números 19 e 16. Obtendo o vigésimo e décimo sétimo elementos do vetor formador do Base64 obtem-se "TQ". Para sinalizar que incluiram-se 4 zeros, o conversor acrescenta "==". Com isso, $M_{ascii} \equiv TQ==_{Base64}$.

Base64 → ASCII

A volta segue o mesmo procedimento, agora realizado em sentido inverso. Inicialmente, deve-se verificar que o comprimento da mensagem codificada em Base64 deve ter um número de caracteres múltiplos de 4. Se não for, há um erro. Tudo começa, verificando se o último caractere do grupo (de 4) em análise é igual a "=". Se não for, este grupo não é terminal e deve sofrer o processamento padrão que é juntar os 24 bits (cada letra dá origem a um número de 6 bits, obtido pela localização de cada letra no vetor gerador do Base64) que devem ser reorganizados a fim de obter os 3 bytes em ASCII. Por exemplo, seja o conjunto Base64 "VGU4". As posições destes caracteres (V, G, U e 4) no vetor formador são: 22, 7, 21 e 57. Subtraindo uma unidade (já que agora a origem dos índices é zero) tem-se 21, 6, 20 e 56. Convertendo tais números a binário tem-se 0 1 0 1 0 1, 0 0 0 1 1 0, 0 1 0 1 0 0 e 1 1 1 0 0 0. Juntando e reorganizando 0101.0100 0110.0101 e 0011.1000, que são "Te8". Entao: $VGU4_{Base64} \equiv Te8_{ascii}$.

Se o último caracter é "=", então deve-se olhar o penúltimo. Se o penúltimo é diferente de "=", então tem-se aqui 2 caracteres ASCII, e depois de juntar 3 x 6 bits = 18 bits, deve-se desprezar os últimos 2 bits e ficar com 16 bits que perfazem os 2 caracteres ASCII buscados.

Por exemplo, seja o Base64 igual a "QTY=". Como o último é = e o penúltimo não é =, então este grupo codifica 2 caracteres ASCII.

Q, T e Y são os caracteres 17, 20 e 25 do vetor formador. Subtraindo 1 fica: 16, 19 e 24. Convertendo a binário fica: 0 1 0 0 0 0, 0 1 0 0 1 1 e 0 1 1 0 0 0. Juntando e desprezando os dois últimos zeros: 0100.0001, 0011.0110 e 00. Com isso os dois caracteres ASCII são: A6, que é a codificação original. Ou QTY=$_{Base64} \equiv A6_{ascii}$.

Se o último e o penúltimo forem =, então é apenas um o caracter original. Deve-se converter os 2 caracteres iniciais (os que são diferentes de =) formando 12 bits e desprezar os últimos 4 bits.

Por exemplo, para desconverter "Rg==", descobre-se que R e g são os elementos 18 e 33 no vetor formador. Subtraindo fica 17 e 32. Convertendo: 0 1 0 0 1 0 e 1 0 0 0 0 0. Reorganizando 0100.1010.0000. O caracter buscado é 'F'. Entao: Rg==$_{Base64} \equiv F_{ascii}$.

Alguns exemplos

de	conversão	para
tMz	ASCII→b64	dE16
Dx	ASCII→b64	RHg=
6	ASCII→b64	Ng==
ZGxO	b64→ASCII	dlN
cU4=	b64→ASCII	qN
SQ==	b64→ASCII	I

☞ Para você fazer

1. Converta de ASCII para Base64
   a) D1R
   b) 5/
   c) n
   d) ajE
2. Converta de Base64 para ASCII
   e) c2Fr
   f) b28=
   g) TQ==
   h) em9P

a	b	c	d
e	f	g	h

U Positivo - UTFPR - PUCPr –
20/05/2024 - 10:31:53.4
Prof Dr P Kantek
(pkantek@gmail.com)
O esquema de codificação Base64
VIVX037a, V: 1.30 76663 DANILO
DEPETRIS SOARES 24CC2202 - 2
SUBSTITUIR e entregar ate
30/maio _____ / _____ / _____

Base64

Base64 é um sistema de numeração posicional que usa 64 como base. Ao produzir um texto que possa ser imprimível em qualquer impressora do mundo ocidental, sem nenhuma chance de erro ou incoerência, apenas podem ser usados os caracteres A..Z, a..z e 0..9. Note que tem-se aqui 26+26+10=62 caracteres. Para completar o conjunto esperado de 64 caracteres, costuma-se usar "+"e "/", nesta ordem. (Note-se que muitos programas que usam este princípio costumam escolher outros 2 caracteres que não sejam "+"e "/", mas nenhum deles usa o rótulo de Base64. São exemplos deste caso: a B64 modificado para URL (usa -"e "_") ou B64 para expressões regulares que usa ("!"e -").

Há ainda um caracter importante, o "="(em Base64) que serve para sinalizar eventuais necessidades de preechimento de *stuffing*.

A idéia desta codificação é poder mandar qualquer conteúdo binário como parte de uma mensagem de e-mail, sem correr o risco de algum conteúdo parcial acabe gerando caracteres de controle do protocolo e ponha a perder a comunicação.

A conversão se dá trocando 3 bytes (8 bits × 3 = 24 bits) em 4 caracteres Base64, cada um deles representado por 6 bits. (então 6 bits × 4 = 24 bits).

Na codificação de 8 bits/byte, usa-se um código qualquer. Atualmente a grande maioria de ambientes usa o padrão ASCII, cuja pequena parte pode ser vista aqui.

uma parte da Tabela ASCII

Binário	Dec	Hex	Car.
0010.1000	40	28	(
0010.1001	41	29	)
0010.1010	42	2A	*
0010.1011	43	2B	+
0010.1100	44	2C	,
0010.1101	45	2D	-
0010.1110	46	2E	.
0010.1111	47	2F	/
0011.0000	48	30	0
0011.0001	49	31	1
0011.0010	50	32	2
0011.0011	51	33	3
0011.0100	52	34	4
0011.0101	53	35	5
0011.0110	54	36	6
0011.0111	55	37	7
0011.1000	56	38	8
0011.1001	57	39	9
0100.0001	65	41	A
0100.0010	66	42	B
0100.0011	67	43	C
0100.0100	68	44	D
0100.0101	69	45	E
0100.0110	70	46	F
0100.0111	71	47	G
0100.1000	72	48	H
0100.1001	73	49	I
0100.1010	74	4A	J
0100.1011	75	4B	K
0100.1100	76	4C	L
0100.1101	77	4D	M
0100.1110	78	4E	N
0100.1111	79	4F	O
0101.0000	80	50	P
0101.0001	81	51	Q
0101.0010	82	52	R
0101.0011	83	53	S
0101.0100	84	54	T
0101.0101	85	55	U
0101.0110	86	56	V
0101.0111	87	57	W
0101.1000	88	58	X
0101.1001	89	59	Y
0101.1010	90	5A	Z
0110.0001	97	61	a
0110.0010	98	62	b
0110.0011	99	63	c
0110.0100	100	64	d
0110.0101	101	65	e
0110.0110	102	66	f
0110.0111	103	67	g
0110.1000	104	68	h
0110.1001	105	69	i
0110.1010	106	6A	j
0110.1011	107	6B	k
0110.1100	108	6C	l
0110.1101	109	6D	m
0110.1110	110	6E	n
0110.1111	111	6F	o
0111.0000	112	70	p
0111.0001	113	71	q
0111.0010	114	72	r
0111.0011	115	73	s
0111.0100	116	74	t
0111.0101	117	75	u
0111.0110	118	76	v
0111.0111	119	77	w
0111.1000	120	78	x
0111.1001	121	79	y
0111.1010	122	7A	z

Para efeito de conversão entre os 2 códigos (ASCII e Base64) deve-se usar o seguinte vetor de conversão:

```
ABCDEFGHIJKLMN
OPQRSTUVWXYZab
cdefghijklmnop
qrstuvwxyz0123
456789+/
```

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

ASCII → Base64

Deve-se pegar 3 bytes a converter (24 bits) e colocá-los lado a lado. Daí, o numerão binário obtido deve ser quebrado em 4 partes, cada uma com 6 bits. Como se sabe, um número binário de 6 bits varia entre 0..63. Este número obtido é que deve indexar o vetor acima estabelecido, dando como resultado 4 caracteres Base64,

Acompanhe no exemplo: Seja passar o ASCII `Aa`) para Base64. Da tabela ASCII, sabe-se que A=0100.0001, a=0110.0001 e)=0010.1001. Juntando tudo, fica: 0100.0001.0110.0001.0010.1001. Rearrumando, fica: 010000.010110.000100.101001, números binários que valem: 16, 22, 4 e 41, respectivamente. Obtendo os elementos 17, 23, 5 e 42 do vetor de elementos do Base64, fica-se com QWEp. Então: $Aa)_{ascii} \equiv QWEp_{Base64}$.

Um cuidado especial deve ser tomado quando se chega ao final do conjunto de caracteres a converter. No fim, podem sobrar nenhum, 1 ou 2 caracteres. Se sobrar nenhum, o problema está resolvido. Se sobrarem 2 caracteres, estes devem ser colocados lado a lado (16 bits) e o conjunto deve ser completado com 2 zeros à direita, formando os 18 bits esperados. Para sinalizar estes 2 *stuffing bits*, após os 3 caracteres Base64 gerados, agrega-se o sinal "=". Por exemplo, seja converter o string ASCII `A1B2C`. O primeiro conjunto `A1B` é convertido normalmente, gerando `QTFC`. Sobra `2C`. Suas configurações em binário são 2=0011.0010 e C=0100.0011. Juntando tudo, fica 0011.0010.0100.0011 cujo comprimento é 16 bits. Agregando 2 zeros à direita e reordenando fica: 0011.0010.0100.0011.00 ou 001100.100100.001100, que correspondem a 12, 36 e 12 respectivamente. Obtendo os elementos 13, 37 e 13 do vetor formador, obtém-se M, k e M respectivamente. A maneira de informar que foram incluídos 2 zeros ao final é colocar um "="como quarto caractere da conversão. Assim: $2C_{ascii} \equiv MkM=_{Base64}$.

Finalmente, se sobrar um único caractere (8 bits), o conversor precisa incluir 4 zeros à direita (*stuffing bits*). Por exemplo, para converter o ASCII "M"para Base64, faz-se: M=0100.1101, com 8 bits. Acrescentando os 4 zeros a direita, fica: 0100.1101.0000 e quebrando em dois números de 6 bits cada, fica: 010011.010000 que são os números 19 e 16. Obtendo o vigésimo e décimo sétimo elementos do vetor formador do Base64 obtem-se "TQ". Para sinalizar que incluiram-se 4 zeros, o conversor acrescenta "==". Com isso, $M_{ascii} \equiv TQ==_{Base64}$.

Base64 → ASCII

A volta segue o mesmo procedimento, agora realizado em sentido inverso. Inicialmente, deve-se verificar que o comprimento da mensagem codificada em Base64 deve ter um número de caracteres múltiplos de 4. Se não for, há um erro. Tudo começa, verificando se o último caractere do grupo (de 4) em análise é igual a "=". Se não for, este grupo não é terminal e deve sofrer o processamento padrão que é juntar os 24 bits (cada letra dá origem a um número de 6 bits, obtido pela localização de cada letra no vetor gerador do Base64) que devem ser reorganizados a fim de obter os 3 bytes em ASCII. Por exemplo, seja o conjunto Base64 "VGU4". As posições destes caracteres (V, G, U e 4) no vetor formador são: 22, 7, 21 e 57. Subtraindo uma unidade (já que agora a origem dos índices é zero) tem-se 21, 6, 20 e 56. Convertendo tais números a binário tem-se 0 1 0 1 0 1, 0 0 0 1 1 0, 0 1 0 1 0 0 e 1 1 1 0 0 0. Juntando e reorganizando 0101.0100 0110.0101 e 0011.1000, que são "Te8". Entao: $VGU4_{Base64} \equiv Te8_{ascii}$.

Se o último caracter é "=", então deve-se olhar o penúltimo. Se o penúltimo é diferente de "=", então tem-se aqui 2 caracteres ASCII, e depois de juntar 3 x 6 bits = 18 bits, deve-se desprezar os últimos 2 bits e ficar com 16 bits que perfazem os 2 caracteres ASCII buscados.

Por exemplo, seja o Base64 igual a "QTY=". Como o último é = e o penúltimo não é =, então este grupo codifica 2 caracteres ASCII.

Q, T e Y são os caracteres 17, 20 e 25 do vetor formador. Subtraindo 1 fica: 16, 19 e 24. Convertendo a binário fica: 0 1 0 0 0 0, 0 1 0 0 1 1 e 0 1 1 0 0 0. Juntando e desprezando os dois últimos zeros: 0100.0001, 0011.0110 e 00. Com isso os dois caracteres ASCII são: A6, que é a codificação original. Ou $QTY=_{Base64} \equiv A6_{ascii}$.

Se o último e o penúltimo forem =, então é apenas um o caracter original. Deve-se converter os 2 caracteres iniciais (os que são diferentes de =) formando 12 bits e desprezar os últimos 4 bits.

Por exemplo, para desconverter "Rg==", descobre-se que R e g são os elementos 18 e 33 no vetor formador. Subtraindo fica 17 e 32. Convertendo: 0 1 0 0 1 0 e 1 0 0 0 0 0. Reorganizando 0100.1010.0000. O caracter buscado é 'F'. Entao: $Rg==_{Base64} \equiv F_{ascii}$.

Alguns exemplos

de	conversão	para
tMz	ASCII→b64	dE16
Dx	ASCII→b64	RHg=
6	ASCII→b64	Ng==
ZGxO	b64→ASCII	dlN
cU4=	b64→ASCII	qN
SQ==	b64→ASCII	I

☞ Para você fazer

1. Converta de ASCII para Base64
   a) //Y
   b) qd
   c) s
   d) 0gb
2. Converta de Base64 para ASCII
   e) UnVU
   f) Um8=
   g) VA==
   h) OFBF

a	b	c	d
e	f	g	h

202-76663 - IT/IR

U Positivo - UTFPR - PUCPr –
20/05/2024 - 10:31:53.4
Prof Dr P Kantek
(pkantek@gmail.com)
O esquema de codificação Base64
VIVX037a, V: 1.30 76513
EDUARDO CAVALHEIRO DIAS
24CC2202 - 3 SUBSTITUIR e
entregar ate 30/maio _____ /
/ _____

Base64

Base64 é um sistema de numeração posicional que usa 64 como base. Ao produzir um texto que possa ser imprimível em qualquer impressora do mundo ocidental, sem nenhuma chance de erro ou incoerência, apenas podem ser usados os caracteres A..Z, a..z e 0..9. Note que tem-se aqui 26+26+10=62 caracteres. Para completar o conjunto esperado de 64 caracteres, costuma-se usar "+"e "/", nesta ordem. (Note-se que muitos programas que usam este princípio costumam escolher outros 2 caracteres que não sejam "+"e "/", mas nenhum deles usa o rótulo de Base64. São exemplos deste caso: a B64 modificado para URL (usa -"e "_") ou B64 para expressões regulares que usa ("!"e -").

Há ainda um caracter importante, o "="(em Base64) que serve para sinalizar eventuais necessidades de preechimento de *stuffing*.

A idéia desta codificação é poder mandar qualquer conteúdo binário como parte de uma mensagem de e-mail, sem correr o risco de algum conteúdo parcial acabe gerando caracteres de controle do protocolo e ponha a perder a comunicação.

A conversão se dá trocando 3 bytes (8 bits × 3 = 24 bits) em 4 caracteres Base64, cada um deles representado por 6 bits. (então 6 bits × 4 = 24 bits).

Na codificação de 8 bits/byte, usa-se um código qualquer. Atualmente a grande maioria de ambientes usa o padrão ASCII, cuja pequena parte pode ser vista aqui.

uma parte da Tabela ASCII

Binário	Dec	Hex	Car.
0010.1000	40	28	(
0010.1001	41	29	)
0010.1010	42	2A	*
0010.1011	43	2B	+
0010.1100	44	2C	,
0010.1101	45	2D	-
0010.1110	46	2E	.
0010.1111	47	2F	/
0011.0000	48	30	0
0011.0001	49	31	1
0011.0010	50	32	2
0011.0011	51	33	3
0011.0100	52	34	4
0011.0101	53	35	5
0011.0110	54	36	6
0011.0111	55	37	7
0011.1000	56	38	8
0011.1001	57	39	9
0100.0001	65	41	A
0100.0010	66	42	B
0100.0011	67	43	C
0100.0100	68	44	D
0100.0101	69	45	E
0100.0110	70	46	F
0100.0111	71	47	G
0100.1000	72	48	H
0100.1001	73	49	I
0100.1010	74	4A	J
0100.1011	75	4B	K
0100.1100	76	4C	L
0100.1101	77	4D	M
0100.1110	78	4E	N
0100.1111	79	4F	O
0101.0000	80	50	P
0101.0001	81	51	Q
0101.0010	82	52	R
0101.0011	83	53	S
0101.0100	84	54	T
0101.0101	85	55	U
0101.0110	86	56	V
0101.0111	87	57	W
0101.1000	88	58	X
0101.1001	89	59	Y
0101.1010	90	5A	Z
0110.0001	97	61	a
0110.0010	98	62	b
0110.0011	99	63	c
0110.0100	100	64	d
0110.0101	101	65	e
0110.0110	102	66	f
0110.0111	103	67	g
0110.1000	104	68	h
0110.1001	105	69	i
0110.1010	106	6A	j
0110.1011	107	6B	k
0110.1100	108	6C	l
0110.1101	109	6D	m
0110.1110	110	6E	n
0110.1111	111	6F	o
0111.0000	112	70	p
0111.0001	113	71	q
0111.0010	114	72	r
0111.0011	115	73	s
0111.0100	116	74	t
0111.0101	117	75	u
0111.0110	118	76	v
0111.0111	119	77	w
0111.1000	120	78	x
0111.1001	121	79	y
0111.1010	122	7A	z

Para efeito de conversão entre os 2 códigos (ASCII e Base64) deve-se usar o seguinte vetor de conversão:

```
ABCDEFGHIJKLMN
OPQRSTUVWXYZab
cdefghijklmnop
qrstuvwxyz0123
456789+/
```

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

ASCII → Base64

Deve-se pegar 3 bytes a converter (24 bits) e colocá-los lado a lado. Daí, o numerão binário obtido deve ser quebrado em 4 partes, cada uma com 6 bits. Como se sabe, um número binário de 6 bits varia entre 0..63. Este número obtido é que deve indexar o vetor acima estabelecido, dando como resultado 4 caracteres Base64,

Acompanhe no exemplo: Seja passar o ASCII `Aa`) para Base64. Da tabela ASCII, sabe-se que A=0100.0001, a=0110.0001 e)=0010.1001. Juntando tudo, fica: 0100.0001.0110.0001.0010.1001. Rearrumando, fica: 010000.010110.000100.101001, números binários que valem: 16, 22, 4 e 41, respectivamente. Obtendo os elementos 17, 23, 5 e 42 do vetor de elementos do Base64, fica-se com QWEp. Então: $Aa)_{ascii} \equiv QWEp_{Base64}$.

Um cuidado especial deve ser tomado quando se chega ao final do conjunto de caracteres a converter. No fim, podem sobrar nenhum, 1 ou 2 caracteres. Se sobrar nenhum, o problema está resolvido. Se sobrarem 2 caracteres, estes devem ser colocados lado a lado (16 bits) e o conjunto deve ser completado com 2 zeros à direita, formando os 18 bits esperados. Para sinalizar estes 2 *stuffing bits*, após os 3 caracteres Base64 gerados, agrega-se o sinal "=". Por exemplo, seja converter o string ASCII `A1B2C`. O primeiro conjunto `A1B` é convertido normalmente, gerando `QTFC`. Sobra `2C`. Suas configurações em binário são 2=0011.0010 e C=0100.0011. Juntando tudo, fica 0011.0010.0100.0011 cujo comprimento é 16 bits. Agregando 2 zeros à direita e reordenando fica: 0011.0010.0100.0011.00 ou 001100.100100.001100, que correspondem a 12, 36 e 12 respectivamente. Obtendo os elementos 13, 37 e 13 do vetor formador, obtém-se M, k e M respectivamente. A maneira de informar que foram incluídos 2 zeros ao final é colocar um "="como quarto caractere da conversão. Assim: $2C_{ascii} \equiv MkM=_{Base64}$.

Finalmente, se sobrar um único caractere (8 bits), o conversor precisa incluir 4 zeros à direita (*stuffing bits*). Por exemplo, para converter o ASCII "M"para Base64, faz-se: M=0100.1101, com 8 bits. Acrescentando os 4 zeros a direita, fica: 0100.1101.0000 e quebrando em dois números de 6 bits cada, fica: 010011.010000 que são os números 19 e 16. Obtendo o vigésimo e décimo sétimo elementos do vetor formador do Base64 obtem-se "TQ". Para sinalizar que incluiram-se 4 zeros, o conversor acrescenta "==". Com isso, $M_{ascii} \equiv TQ==_{Base64}$.

Base64 → ASCII

A volta segue o mesmo procedimento, agora realizado em sentido inverso. Inicialmente, deve-se verificar que o comprimento da mensagem codificada em Base64 deve ter um número de caracteres múltiplos de 4. Se não for, há um erro. Tudo começa, verificando se o último caractere do grupo (de 4) em análise é igual a "=". Se não for, este grupo não é terminal e deve sofrer o processamento padrão que é juntar os 24 bits (cada letra dá origem a um número de 6 bits, obtido pela localização de cada letra no vetor gerador do Base64) que devem ser reorganizados a fim de obter os 3 bytes em ASCII. Por exemplo, seja o conjunto Base64 "VGU4". As posições destes caracteres (V, G, U e 4) no vetor formador são: 22, 7, 21 e 57. Subtraindo uma unidade (já que agora a origem dos índices é zero) tem-se 21, 6, 20 e 56. Convertendo tais números a binário tem-se 0 1 0 1 0 1, 0 0 0 1 1 0, 0 1 0 1 0 0 e 1 1 1 0 0 0. Juntando e reorganizando 0101.0100 0110.0101 e 0011.1000, que são "Te8". Entao: $VGU4_{Base64} \equiv Te8_{ascii}$.

Se o último caracter é "=", então deve-se olhar o penúltimo. Se o penúltimo é diferente de "=", então tem-se aqui 2 caracteres ASCII, e depois de juntar 3 x 6 bits = 18 bits, deve-se desprezar os últimos 2 bits e ficar com 16 bits que perfazem os 2 caracteres ASCII buscados.

Por exemplo, seja o Base64 igual a "QTY=". Como o último é = e o penúltimo não é =, então este grupo codifica 2 caracteres ASCII.

Q, T e Y são os caracteres 17, 20 e 25 do vetor formador. Subtraindo 1 fica: 16, 19 e 24. Convertendo a binário fica: 0 1 0 0 0 0, 0 1 0 0 1 1 e 0 1 1 0 0 0. Juntando e desprezando os dois últimos zeros: 0100.0001, 0011.0110 e 00. Com isso os dois caracteres ASCII são: A6, que é a codificação original. Ou $QTY=_{Base64} \equiv A6_{ascii}$.

Se o último e o penúltimo forem =, então é apenas um o caracter original. Deve-se converter os 2 caracteres iniciais (os que são diferentes de =) formando 12 bits e desprezar os últimos 4 bits.

Por exemplo, para desconverter "Rg==", descobre-se que R e g são os elementos 18 e 33 no vetor formador. Subtraindo fica 17 e 32. Convertendo: 0 1 0 0 1 0 e 1 0 0 0 0 0. Reorganizando 0100.1010.0000. O caracter buscado é 'F'. Entao: $Rg==_{Base64} \equiv F_{ascii}$.

Alguns exemplos

de	conversão	para
tMz	ASCII→b64	dE16
Dx	ASCII→b64	RHg=
6	ASCII→b64	Ng==
ZGxO	b64→ASCII	dlN
cU4=	b64→ASCII	qN
SQ==	b64→ASCII	I

☞ Para você fazer

1. Converta de ASCII para Base64
   a) 4HD
   b) tD
   c) V
   d) LNa
2. Converta de Base64 para ASCII
   e) VXJp
   f) cC8=
   g) aw==
   h) STJk

a	b	c	d
e	f	g	h

U Positivo - UTFPR - PUCPr –
20/05/2024 - 10:31:53.4
Prof Dr P Kantek
(pkantek@gmail.com)
O esquema de codificação Base64
VIVX037a, V: 1.30 76520 ERIC
CORDEIRO 24CC2202 - 4
SUBSTITUIR e entregar ate
30/maio _____ / _____ / _____

Base64

Base64 é um sistema de numeração posicional que usa 64 como base. Ao produzir um texto que possa ser imprimível em qualquer impressora do mundo ocidental, sem nenhuma chance de erro ou incoerência, apenas podem ser usados os caracteres A..Z, a..z e 0..9. Note que tem-se aqui 26+26+10=62 caracteres. Para completar o conjunto esperado de 64 caracteres, costuma-se usar "+"e "/", nesta ordem. (Note-se que muitos programas que usam este princípio costumam escolher outros 2 caracteres que não sejam "+"e "/", mas nenhum deles usa o rótulo de Base64. São exemplos deste caso: a B64 modificado para URL (usa -"e "_") ou B64 para expressões regulares que usa ("!"e -").

Há ainda um caracter importante, o "="(em Base64) que serve para sinalizar eventuais necessidades de preechimento de *stuffing*.

A idéia desta codificação é poder mandar qualquer conteúdo binário como parte de uma mensagem de e-mail, sem correr o risco de algum conteúdo parcial acabe gerando caracteres de controle do protocolo e ponha a perder a comunicação.

A conversão se dá trocando 3 bytes (8 bits × 3 = 24 bits) em 4 caracteres Base64, cada um deles representado por 6 bits. (então 6 bits × 4 = 24 bits).

Na codificação de 8 bits/byte, usa-se um código qualquer. Atualmente a grande maioria de ambientes usa o padrão ASCII, cuja pequena parte pode ser vista aqui.

uma parte da Tabela ASCII

Binário	Dec	Hex	Car.
0010.1000	40	28	(
0010.1001	41	29	)
0010.1010	42	2A	*
0010.1011	43	2B	+
0010.1100	44	2C	,
0010.1101	45	2D	-
0010.1110	46	2E	.
0010.1111	47	2F	/
0011.0000	48	30	0
0011.0001	49	31	1
0011.0010	50	32	2
0011.0011	51	33	3
0011.0100	52	34	4
0011.0101	53	35	5
0011.0110	54	36	6
0011.0111	55	37	7
0011.1000	56	38	8
0011.1001	57	39	9
0100.0001	65	41	A
0100.0010	66	42	B
0100.0011	67	43	C
0100.0100	68	44	D
0100.0101	69	45	E
0100.0110	70	46	F
0100.0111	71	47	G
0100.1000	72	48	H
0100.1001	73	49	I
0100.1010	74	4A	J
0100.1011	75	4B	K
0100.1100	76	4C	L
0100.1101	77	4D	M
0100.1110	78	4E	N
0100.1111	79	4F	O
0101.0000	80	50	P
0101.0001	81	51	Q
0101.0010	82	52	R
0101.0011	83	53	S
0101.0100	84	54	T
0101.0101	85	55	U
0101.0110	86	56	V
0101.0111	87	57	W
0101.1000	88	58	X
0101.1001	89	59	Y
0101.1010	90	5A	Z
0110.0001	97	61	a
0110.0010	98	62	b
0110.0011	99	63	c
0110.0100	100	64	d
0110.0101	101	65	e
0110.0110	102	66	f
0110.0111	103	67	g
0110.1000	104	68	h
0110.1001	105	69	i
0110.1010	106	6A	j
0110.1011	107	6B	k
0110.1100	108	6C	l
0110.1101	109	6D	m
0110.1110	110	6E	n
0110.1111	111	6F	o
0111.0000	112	70	p
0111.0001	113	71	q
0111.0010	114	72	r
0111.0011	115	73	s
0111.0100	116	74	t
0111.0101	117	75	u
0111.0110	118	76	v
0111.0111	119	77	w
0111.1000	120	78	x
0111.1001	121	79	y
0111.1010	122	7A	z

Para efeito de conversão entre os 2 códigos (ASCII e Base64) deve-se usar o seguinte vetor de conversão:

```
ABCDEFGHIJKLMN
OPQRSTUVWXYZab
cdefghijklmnop
qrstuvwxyz0123
456789+/
```

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

ASCII → Base64

Deve-se pegar 3 bytes a converter (24 bits) e colocá-los lado a lado. Daí, o numerão binário obtido deve ser quebrado em 4 partes, cada uma com 6 bits. Como se sabe, um número binário de 6 bits varia entre 0..63. Este número obtido é que deve indexar o vetor acima estabelecido, dando como resultado 4 caracteres Base64,

Acompanhe no exemplo: Seja passar o ASCII `Aa`) para Base64. Da tabela ASCII, sabe-se que A=0100.0001, a=0110.0001 e)=0010.1001. Juntando tudo, fica: 0100.0001.0110.0001.0010.1001. Rearrumando, fica: 010000.010110.000100.101001, números binários que valem: 16, 22, 4 e 41, respectivamente. Obtendo os elementos 17, 23, 5 e 42 do vetor de elementos do Base64, fica-se com QWEp. Então: $Aa)_{ascii} \equiv QWEp_{Base64}$.

Um cuidado especial deve ser tomado quando se chega ao final do conjunto de caracteres a converter. No fim, podem sobrar nenhum, 1 ou 2 caracteres. Se sobrar nenhum, o problema está resolvido. Se sobrarem 2 caracteres, estes devem ser colocados lado a lado (16 bits) e o conjunto deve ser completado com 2 zeros à direita, formando os 18 bits esperados. Para sinalizar estes 2 *stuffing bits*, após os 3 caracteres Base64 gerados, agrega-se o sinal "=". Por exemplo, seja converter o string ASCII `A1B2C`. O primeiro conjunto `A1B` é convertido normalmente, gerando `QTFC`. Sobra `2C`. Suas configurações em binário são 2=0011.0010 e C=0100.0011. Juntando tudo, fica 0011.0010.0100.0011 cujo comprimento é 16 bits. Agregando 2 zeros à direita e reordenando fica: 0011.0010.0100.0011.00 ou 001100.100100.001100, que correspondem a 12, 36 e 12 respectivamente. Obtendo os elementos 13, 37 e 13 do vetor formador, obtém-se M, k e M respectivamente. A maneira de informar que foram incluídos 2 zeros ao final é colocar um "="como quarto caractere da conversão. Assim: $2C_{ascii} \equiv MkM=_{Base64}$.

Finalmente, se sobrar um único caractere (8 bits), o conversor precisa incluir 4 zeros à direita (*stuffing bits*). Por exemplo, para converter o ASCII "M"para Base64, faz-se: M=0100.1101, com 8 bits. Acrescentando os 4 zeros a direita, fica: 0100.1101.0000 e quebrando em dois números de 6 bits cada, fica: 010011.010000 que são os números 19 e 16. Obtendo o vigésimo e décimo sétimo elementos do vetor formador do Base64 obtem-se "TQ". Para sinalizar que incluiram-se 4 zeros, o conversor acrescenta "==". Com isso, $M_{ascii} \equiv TQ==_{Base64}$.

Base64 → ASCII

A volta segue o mesmo procedimento, agora realizado em sentido inverso. Inicialmente, deve-se verificar que o comprimento da mensagem codificada em Base64 deve ter um número de caracteres múltiplos de 4. Se não for, há um erro. Tudo começa, verificando se o último caractere do grupo (de 4) em análise é igual a "=". Se não for, este grupo não é terminal e deve sofrer o processamento padrão que é juntar os 24 bits (cada letra dá origem a um número de 6 bits, obtido pela localização de cada letra no vetor gerador do Base64) que devem ser reorganizados a fim de obter os 3 bytes em ASCII. Por exemplo, seja o conjunto Base64 "VGU4". As posições destes caracteres (V, G, U e 4) no vetor formador são: 22, 7, 21 e 57. Subtraindo uma unidade (já que agora a origem dos índices é zero) tem-se 21, 6, 20 e 56. Convertendo tais números a binário tem-se 0 1 0 1 0 1, 0 0 0 1 1 0, 0 1 0 1 0 0 e 1 1 1 0 0 0. Juntando e reorganizando 0101.0100 0110.0101 e 0011.1000, que são "Te8". Entao: $VGU4_{Base64} \equiv Te8_{ascii}$.

Se o último caracter é "=", então deve-se olhar o penúltimo. Se o penúltimo é diferente de "=", então tem-se aqui 2 caracteres ASCII, e depois de juntar 3 x 6 bits = 18 bits, deve-se desprezar os últimos 2 bits e ficar com 16 bits que perfazem os 2 caracteres ASCII buscados.

Por exemplo, seja o Base64 igual a "QTY=". Como o último é = e o penúltimo não é =, então este grupo codifica 2 caracteres ASCII.

Q, T e Y são os caracteres 17, 20 e 25 do vetor formador. Subtraindo 1 fica: 16, 19 e 24. Convertendo a binário fica: 0 1 0 0 0 0, 0 1 0 0 1 1 e 0 1 1 0 0 0. Juntando e desprezando os dois últimos zeros: 0100.0001, 0011.0110 e 00. Com isso os dois caracteres ASCII são: A6, que é a codificação original. Ou $QTY=_{Base64} \equiv A6_{ascii}$.

Se o último e o penúltimo forem =, então é apenas um o caracter original. Deve-se converter os 2 caracteres iniciais (os que são diferentes de =) formando 12 bits e desprezar os últimos 4 bits.

Por exemplo, para desconverter "Rg==", descobre-se que R e g são os elementos 18 e 33 no vetor formador. Subtraindo fica 17 e 32. Convertendo: 0 1 0 0 1 0 e 1 0 0 0 0 0. Reorganizando 0100.1010.0000. O caracter buscado é 'F'. Entao: $Rg==_{Base64} \equiv F_{ascii}$.

Alguns exemplos

de	conversão	para
tMz	ASCII→b64	dE16
Dx	ASCII→b64	RHg=
6	ASCII→b64	Ng==
ZGxO	b64→ASCII	dlN
cU4=	b64→ASCII	qN
SQ==	b64→ASCII	I

☞ Para você fazer

1. Converta de ASCII para Base64
   a) 49Q
   b) BD
   c) D
   d) HVw
2. Converta de Base64 para ASCII
   e) NFJr
   f) TVE=
   g) VA==
   h) QWdX

a	b	c	d
e	f	g	h

202-76520 - IT/IR

U Positivo - UTFPR - PUCPr –
20/05/2024 - 10:31:53.4
Prof Dr P Kantek
(pkantek@gmail.com)
O esquema de codificação Base64
VIVX037a, V: 1.30 76537 FELIPE
EIJI KANO 24CC2202 - 5
SUBSTITUIR e entregar ate
30/maio _____ / _____ / _____

Base64

Base64 é um sistema de numeração posicional que usa 64 como base. Ao produzir um texto que possa ser imprimível em qualquer impressora do mundo ocidental, sem nenhuma chance de erro ou incoerência, apenas podem ser usados os caracteres A..Z, a..z e 0..9. Note que tem-se aqui 26+26+10=62 caracteres. Para completar o conjunto esperado de 64 caracteres, costuma-se usar "+"e "/", nesta ordem. (Note-se que muitos programas que usam este princípio costumam escolher outros 2 caracteres que não sejam "+"e "/", mas nenhum deles usa o rótulo de Base64. São exemplos deste caso: a B64 modificado para URL (usa -"e "_") ou B64 para expressões regulares que usa ("!"e -").

Há ainda um caracter importante, o "="(em Base64) que serve para sinalizar eventuais necessidades de preechimento de *stuffing*.

A idéia desta codificação é poder mandar qualquer conteúdo binário como parte de uma mensagem de e-mail, sem correr o risco de algum conteúdo parcial acabe gerando caracteres de controle do protocolo e ponha a perder a comunicação.

A conversão se dá trocando 3 bytes (8 bits × 3 = 24 bits) em 4 caracteres Base64, cada um deles representado por 6 bits. (então 6 bits × 4 = 24 bits).

Na codificação de 8 bits/byte, usa-se um código qualquer. Atualmente a grande maioria de ambientes usa o padrão ASCII, cuja pequena parte pode ser vista aqui.

uma parte da Tabela ASCII

Binário	Dec	Hex	Car.
0010.1000	40	28	(
0010.1001	41	29	)
0010.1010	42	2A	*
0010.1011	43	2B	+
0010.1100	44	2C	,
0010.1101	45	2D	-
0010.1110	46	2E	.
0010.1111	47	2F	/
0011.0000	48	30	0
0011.0001	49	31	1
0011.0010	50	32	2
0011.0011	51	33	3
0011.0100	52	34	4
0011.0101	53	35	5
0011.0110	54	36	6
0011.0111	55	37	7
0011.1000	56	38	8
0011.1001	57	39	9
0100.0001	65	41	A
0100.0010	66	42	B
0100.0011	67	43	C
0100.0100	68	44	D
0100.0101	69	45	E
0100.0110	70	46	F
0100.0111	71	47	G
0100.1000	72	48	H
0100.1001	73	49	I
0100.1010	74	4A	J
0100.1011	75	4B	K
0100.1100	76	4C	L
0100.1101	77	4D	M
0100.1110	78	4E	N
0100.1111	79	4F	O
0101.0000	80	50	P
0101.0001	81	51	Q
0101.0010	82	52	R
0101.0011	83	53	S
0101.0100	84	54	T
0101.0101	85	55	U
0101.0110	86	56	V
0101.0111	87	57	W
0101.1000	88	58	X
0101.1001	89	59	Y
0101.1010	90	5A	Z
0110.0001	97	61	a
0110.0010	98	62	b
0110.0011	99	63	c
0110.0100	100	64	d
0110.0101	101	65	e
0110.0110	102	66	f
0110.0111	103	67	g
0110.1000	104	68	h
0110.1001	105	69	i
0110.1010	106	6A	j
0110.1011	107	6B	k
0110.1100	108	6C	l
0110.1101	109	6D	m
0110.1110	110	6E	n
0110.1111	111	6F	o
0111.0000	112	70	p
0111.0001	113	71	q
0111.0010	114	72	r
0111.0011	115	73	s
0111.0100	116	74	t
0111.0101	117	75	u
0111.0110	118	76	v
0111.0111	119	77	w
0111.1000	120	78	x
0111.1001	121	79	y
0111.1010	122	7A	z

Para efeito de conversão entre os 2 códigos (ASCII e Base64) deve-se usar o seguinte vetor de conversão:

```
ABCDEFGHIJKLMN
OPQRSTUVWXYZab
cdefghijklmnop
qrstuvwxyz0123
456789+/
```

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

ASCII → Base64

Deve-se pegar 3 bytes a converter (24 bits) e colocá-los lado a lado. Daí, o numerão binário obtido deve ser quebrado em 4 partes, cada uma com 6 bits. Como se sabe, um número binário de 6 bits varia entre 0..63. Este número obtido é que deve indexar o vetor acima estabelecido, dando como resultado 4 caracteres Base64,

Acompanhe no exemplo: Seja passar o ASCII `Aa`) para Base64. Da tabela ASCII, sabe-se que A=0100.0001, a=0110.0001 e)=0010.1001. Juntando tudo, fica: 0100.0001.0110.0001.0010.1001. Rearrumando, fica: 010000.010110.000100.101001, números binários que valem: 16, 22, 4 e 41, respectivamente. Obtendo os elementos 17, 23, 5 e 42 do vetor de elementos do Base64, fica-se com QWEp. Então: $Aa)_{ascii} \equiv QWEp_{Base64}$.

Um cuidado especial deve ser tomado quando se chega ao final do conjunto de caracteres a converter. No fim, podem sobrar nenhum, 1 ou 2 caracteres. Se sobrar nenhum, o problema está resolvido. Se sobrarem 2 caracteres, estes devem ser colocados lado a lado (16 bits) e o conjunto deve ser completado com 2 zeros à direita, formando os 18 bits esperados. Para sinalizar estes 2 *stuffing bits*, após os 3 caracteres Base64 gerados, agrega-se o sinal "=". Por exemplo, seja converter o string ASCII `A1B2C`. O primeiro conjunto `A1B` é convertido normalmente, gerando `QTFC`. Sobra `2C`. Suas configurações em binário são 2=0011.0010 e C=0100.0011. Juntando tudo, fica 0011.0010.0100.0011 cujo comprimento é 16 bits. Agregando 2 zeros à direita e reordenando fica: 0011.0010.0100.0011.00 ou 001100.100100.001100, que correspondem a 12, 36 e 12 respectivamente. Obtendo os elementos 13, 37 e 13 do vetor formador, obtém-se M, k e M respectivamente. A maneira de informar que foram incluídos 2 zeros ao final é colocar um "="como quarto caractere da conversão. Assim: $2C_{ascii} \equiv MkM=_{Base64}$.

Finalmente, se sobrar um único caractere (8 bits), o conversor precisa incluir 4 zeros à direita (*stuffing bits*). Por exemplo, para converter o ASCII "M"para Base64, faz-se: M=0100.1101, com 8 bits. Acrescentando os 4 zeros a direita, fica: 0100.1101.0000 e quebrando em dois números de 6 bits cada, fica: 010011.010000 que são os números 19 e 16. Obtendo o vigésimo e décimo sétimo elementos do vetor formador do Base64 obtem-se "TQ". Para sinalizar que incluiram-se 4 zeros, o conversor acrescenta "==". Com isso, $M_{ascii} \equiv TQ==_{Base64}$.

Base64 → ASCII

A volta segue o mesmo procedimento, agora realizado em sentido inverso. Inicialmente, deve-se verificar que o comprimento da mensagem codificada em Base64 deve ter um número de caracteres múltiplos de 4. Se não for, há um erro. Tudo começa, verificando se o último caractere do grupo (de 4) em análise é igual a "=". Se não for, este grupo não é terminal e deve sofrer o processamento padrão que é juntar os 24 bits (cada letra dá origem a um número de 6 bits, obtido pela localização de cada letra no vetor gerador do Base64) que devem ser reorganizados a fim de obter os 3 bytes em ASCII. Por exemplo, seja o conjunto Base64 "VGU4". As posições destes caracteres (V, G, U e 4) no vetor formador são: 22, 7, 21 e 57. Subtraindo uma unidade (já que agora a origem dos índices é zero) tem-se 21, 6, 20 e 56. Convertendo tais números a binário tem-se 0 1 0 1 0 1, 0 0 0 1 1 0, 0 1 0 1 0 0 e 1 1 1 0 0 0. Juntando e reorganizando 0101.0100 0110.0101 e 0011.1000, que são "Te8". Entao: $VGU4_{Base64} \equiv Te8_{ascii}$.

Se o último caracter é "=", então deve-se olhar o penúltimo. Se o penúltimo é diferente de "=", então tem-se aqui 2 caracteres ASCII, e depois de juntar 3 x 6 bits = 18 bits, deve-se desprezar os últimos 2 bits e ficar com 16 bits que perfazem os 2 caracteres ASCII buscados.

Por exemplo, seja o Base64 igual a "QTY=". Como o último é = e o penúltimo não é =, então este grupo codifica 2 caracteres ASCII.

Q, T e Y são os caracteres 17, 20 e 25 do vetor formador. Subtraindo 1 fica: 16, 19 e 24. Convertendo a binário fica: 0 1 0 0 0 0, 0 1 0 0 1 1 e 0 1 1 0 0 0. Juntando e desprezando os dois últimos zeros: 0100.0001, 0011.0110 e 00. Com isso os dois caracteres ASCII são: A6, que é a codificação original. Ou QTY=$_{Base64} \equiv A6_{ascii}$.

Se o último e o penúltimo forem =, então é apenas um o caracter original. Deve-se converter os 2 caracteres iniciais (os que são diferentes de =) formando 12 bits e desprezar os últimos 4 bits.

Por exemplo, para desconverter "Rg==", descobre-se que R e g são os elementos 18 e 33 no vetor formador. Subtraindo fica 17 e 32. Convertendo: 0 1 0 0 1 0 e 1 0 0 0 0 0. Reorganizando 0100.1010.0000. O caracter buscado é 'F'. Entao: Rg==$_{Base64} \equiv F_{ascii}$.

Alguns exemplos

de	conversão	para
tMz	ASCII→b64	dE16
Dx	ASCII→b64	RHg=
6	ASCII→b64	Ng==
ZGxO	b64→ASCII	dlN
cU4=	b64→ASCII	qN
SQ==	b64→ASCII	I

☞ Para você fazer

1. Converta de ASCII para Base64
   a) 6pT
   b) 0V
   c) h
   d) BK8
2. Converta de Base64 para ASCII
   e) LzlL
   f) L1E=
   g) WQ==
   h) QTNs

a	b	c	d
e	f	g	h

202-76537 - IT/IR

U Positivo - UTFPR - PUCPr –
20/05/2024 - 10:31:53.4
Prof Dr P Kantek
(pkantek@gmail.com)
O esquema de codificação Base64
VIVX037a, V: 1.30 76544 FELIPE
JESMIEL LEITE 24CC2202 - 6
SUBSTITUIR e entregar ate
30/maio _____ / _____ / _____

Base64

Base64 é um sistema de numeração posicional que usa 64 como base. Ao produzir um texto que possa ser imprimível em qualquer impressora do mundo ocidental, sem nenhuma chance de erro ou incoerência, apenas podem ser usados os caracteres A..Z, a..z e 0..9. Note que tem-se aqui 26+26+10=62 caracteres. Para completar o conjunto esperado de 64 caracteres, costuma-se usar "+"e "/", nesta ordem. (Note-se que muitos programas que usam este princípio costumam escolher outros 2 caracteres que não sejam "+"e "/", mas nenhum deles usa o rótulo de Base64. São exemplos deste caso: a B64 modificado para URL (usa -"e "_") ou B64 para expressões regulares que usa ("!"e -").

Há ainda um caracter importante, o "="(em Base64) que serve para sinalizar eventuais necessidades de preechimento de *stuffing*.

A idéia desta codificação é poder mandar qualquer conteúdo binário como parte de uma mensagem de e-mail, sem correr o risco de algum conteúdo parcial acabe gerando caracteres de controle do protocolo e ponha a perder a comunicação.

A conversão se dá trocando 3 bytes (8 bits × 3 = 24 bits) em 4 caracteres Base64, cada um deles representado por 6 bits. (então 6 bits × 4 = 24 bits).

Na codificação de 8 bits/byte, usa-se um código qualquer. Atualmente a grande maioria de ambientes usa o padrão ASCII, cuja pequena parte pode ser vista aqui.

uma parte da Tabela ASCII

Binário	Dec	Hex	Car.
0010.1000	40	28	(
0010.1001	41	29	)
0010.1010	42	2A	*
0010.1011	43	2B	+
0010.1100	44	2C	,
0010.1101	45	2D	-
0010.1110	46	2E	.
0010.1111	47	2F	/
0011.0000	48	30	0
0011.0001	49	31	1
0011.0010	50	32	2
0011.0011	51	33	3
0011.0100	52	34	4
0011.0101	53	35	5
0011.0110	54	36	6
0011.0111	55	37	7
0011.1000	56	38	8
0011.1001	57	39	9
0100.0001	65	41	A
0100.0010	66	42	B
0100.0011	67	43	C
0100.0100	68	44	D
0100.0101	69	45	E
0100.0110	70	46	F
0100.0111	71	47	G
0100.1000	72	48	H
0100.1001	73	49	I
0100.1010	74	4A	J
0100.1011	75	4B	K
0100.1100	76	4C	L
0100.1101	77	4D	M
0100.1110	78	4E	N
0100.1111	79	4F	O
0101.0000	80	50	P
0101.0001	81	51	Q
0101.0010	82	52	R
0101.0011	83	53	S
0101.0100	84	54	T
0101.0101	85	55	U
0101.0110	86	56	V
0101.0111	87	57	W
0101.1000	88	58	X
0101.1001	89	59	Y
0101.1010	90	5A	Z
0110.0001	97	61	a
0110.0010	98	62	b
0110.0011	99	63	c
0110.0100	100	64	d
0110.0101	101	65	e
0110.0110	102	66	f
0110.0111	103	67	g
0110.1000	104	68	h
0110.1001	105	69	i
0110.1010	106	6A	j
0110.1011	107	6B	k
0110.1100	108	6C	l
0110.1101	109	6D	m
0110.1110	110	6E	n
0110.1111	111	6F	o
0111.0000	112	70	p
0111.0001	113	71	q
0111.0010	114	72	r
0111.0011	115	73	s
0111.0100	116	74	t
0111.0101	117	75	u
0111.0110	118	76	v
0111.0111	119	77	w
0111.1000	120	78	x
0111.1001	121	79	y
0111.1010	122	7A	z

Para efeito de conversão entre os 2 códigos (ASCII e Base64) deve-se usar o seguinte vetor de conversão:

```
ABCDEFGHIJKLMN
OPQRSTUVWXYZab
cdefghijklmnop
qrstuvwxyz0123
456789+/
```

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

ASCII → Base64

Deve-se pegar 3 bytes a converter (24 bits) e colocá-los lado a lado. Daí, o numerão binário obtido deve ser quebrado em 4 partes, cada uma com 6 bits. Como se sabe, um número binário de 6 bits varia entre 0..63. Este número obtido é que deve indexar o vetor acima estabelecido, dando como resultado 4 caracteres Base64,

Acompanhe no exemplo: Seja passar o ASCII `Aa`) para Base64. Da tabela ASCII, sabe-se que A=0100.0001, a=0110.0001 e)=0010.1001. Juntando tudo, fica: 0100.0001.0110.0001.0010.1001. Rearrumando, fica: 010000.010110.000100.101001, números binários que valem: 16, 22, 4 e 41, respectivamente. Obtendo os elementos 17, 23, 5 e 42 do vetor de elementos do Base64, fica-se com QWEp. Então: $Aa)_{ascii} \equiv QWEp_{Base64}$.

Um cuidado especial deve ser tomado quando se chega ao final do conjunto de caracteres a converter. No fim, podem sobrar nenhum, 1 ou 2 caracteres. Se sobrar nenhum, o problema está resolvido. Se sobrarem 2 caracteres, estes devem ser colocados lado a lado (16 bits) e o conjunto deve ser completado com 2 zeros à direita, formando os 18 bits esperados. Para sinalizar estes 2 *stuffing bits*, após os 3 caracteres Base64 gerados, agrega-se o sinal "=". Por exemplo, seja converter o string ASCII `A1B2C`. O primeiro conjunto `A1B` é convertido normalmente, gerando `QTFC`. Sobra `2C`. Suas configurações em binário são 2=0011.0010 e C=0100.0011. Juntando tudo, fica 0011.0010.0100.0011 cujo comprimento é 16 bits. Agregando 2 zeros à direita e reordenando fica: 0011.0010.0100.0011.00 ou 001100.100100.001100, que correspondem a 12, 36 e 12 respectivamente. Obtendo os elementos 13, 37 e 13 do vetor formador, obtém-se M, k e M respectivamente. A maneira de informar que foram incluídos 2 zeros ao final é colocar um "="como quarto caractere da conversão. Assim: $2C_{ascii} \equiv MkM=_{Base64}$.

Finalmente, se sobrar um único caractere (8 bits), o conversor precisa incluir 4 zeros à direita (*stuffing bits*). Por exemplo, para converter o ASCII "M"para Base64, faz-se: M=0100.1101, com 8 bits. Acrescentando os 4 zeros a direita, fica: 0100.1101.0000 e quebrando em dois números de 6 bits cada, fica: 010011.010000 que são os números 19 e 16. Obtendo o vigésimo e décimo sétimo elementos do vetor formador do Base64 obtem-se "TQ". Para sinalizar que incluiram-se 4 zeros, o conversor acrescenta "==". Com isso, $M_{ascii} \equiv TQ==_{Base64}$.

Base64 → ASCII

A volta segue o mesmo procedimento, agora realizado em sentido inverso. Inicialmente, deve-se verificar que o comprimento da mensagem codificada em Base64 deve ter um número de caracteres múltiplos de 4. Se não for, há um erro. Tudo começa, verificando se o último caractere do grupo (de 4) em análise é igual a "=". Se não for, este grupo não é terminal e deve sofrer o processamento padrão que é juntar os 24 bits (cada letra dá origem a um número de 6 bits, obtido pela localização de cada letra no vetor gerador do Base64) que devem ser reorganizados a fim de obter os 3 bytes em ASCII. Por exemplo, seja o conjunto Base64 "VGU4". As posições destes caracteres (V, G, U e 4) no vetor formador são: 22, 7, 21 e 57. Subtraindo uma unidade (já que agora a origem dos índices é zero) tem-se 21, 6, 20 e 56. Convertendo tais números a binário tem-se 0 1 0 1 0 1, 0 0 0 1 1 0, 0 1 0 1 0 0 e 1 1 1 0 0 0. Juntando e reorganizando 0101.0100 0110.0101 e 0011.1000, que são "Te8". Entao: $VGU4_{Base64} \equiv Te8_{ascii}$.

Se o último caracter é "=", então deve-se olhar o penúltimo. Se o penúltimo é diferente de "=", então tem-se aqui 2 caracteres ASCII, e depois de juntar 3 x 6 bits = 18 bits, deve-se desprezar os últimos 2 bits e ficar com 16 bits que perfazem os 2 caracteres ASCII buscados.

Por exemplo, seja o Base64 igual a "QTY=". Como o último é = e o penúltimo não é =, então este grupo codifica 2 caracteres ASCII.

Q, T e Y são os caracteres 17, 20 e 25 do vetor formador. Subtraindo 1 fica: 16, 19 e 24. Convertendo a binário fica: 0 1 0 0 0 0, 0 1 0 0 1 1 e 0 1 1 0 0 0. Juntando e desprezando os dois últimos zeros: 0100.0001, 0011.0110 e 00. Com isso os dois caracteres ASCII são: A6, que é a codificação original. Ou $QTY=_{Base64} \equiv A6_{ascii}$.

Se o último e o penúltimo forem =, então é apenas um o caracter original. Deve-se converter os 2 caracteres iniciais (os que são diferentes de =) formando 12 bits e desprezar os últimos 4 bits.

Por exemplo, para desconverter "Rg==", descobre-se que R e g são os elementos 18 e 33 no vetor formador. Subtraindo fica 17 e 32. Convertendo: 0 1 0 0 1 0 e 1 0 0 0 0 0. Reorganizando 0100.1010.0000. O caracter buscado é 'F'. Entao: $Rg==_{Base64} \equiv F_{ascii}$.

Alguns exemplos

de	conversão	para
tMz	ASCII→b64	dE16
Dx	ASCII→b64	RHg=
6	ASCII→b64	Ng==
ZGxO	b64→ASCII	dlN
cU4=	b64→ASCII	qN
SQ==	b64→ASCII	I

☞ Para você fazer

1. Converta de ASCII para Base64
   a) e+G
   b) Lp
   c) Q
   d) PsT
2. Converta de Base64 para ASCII
   e) OTl5
   f) R2I=
   g) VA==
   h) VWhV

a	b	c	d
e	f	g	h

202-76544 - IT/IR

U Positivo - UTFPR - PUCPr –
20/05/2024 - 10:31:53.4
Prof Dr P Kantek
(pkantek@gmail.com)
O esquema de codificação Base64
VIVX037a, V: 1.30 76551
GABRIELA BERNARDO
PASQUALOTTO 24CC2202 - 7
SUBSTITUIR e entregar ate
30/maio _____ / _____ / _____

Base64

Base64 é um sistema de numeração posicional que usa 64 como base. Ao produzir um texto que possa ser imprimível em qualquer impressora do mundo ocidental, sem nenhuma chance de erro ou incoerência, apenas podem ser usados os caracteres A..Z, a..z e 0..9. Note que tem-se aqui 26+26+10=62 caracteres. Para completar o conjunto esperado de 64 caracteres, costuma-se usar "+"e "/", nesta ordem. (Note-se que muitos programas que usam este princípio costumam escolher outros 2 caracteres que não sejam "+"e "/", mas nenhum deles usa o rótulo de Base64. São exemplos deste caso: a B64 modificado para URL (usa -"e "_") ou B64 para expressões regulares que usa ("!"e -").

Há ainda um caracter importante, o "="(em Base64) que serve para sinalizar eventuais necessidades de preechimento de *stuffing*.

A idéia desta codificação é poder mandar qualquer conteúdo binário como parte de uma mensagem de e-mail, sem correr o risco de algum conteúdo parcial acabe gerando caracteres de controle do protocolo e ponha a perder a comunicação.

A conversão se dá trocando 3 bytes (8 bits × 3 = 24 bits) em 4 caracteres Base64, cada um deles representado por 6 bits. (então 6 bits × 4 = 24 bits).

Na codificação de 8 bits/byte, usa-se um código qualquer. Atualmente a grande maioria de ambientes usa o padrão ASCII, cuja pequena parte pode ser vista aqui.

uma parte da Tabela ASCII

Binário	Dec	Hex	Car.
0010.1000	40	28	(
0010.1001	41	29	)
0010.1010	42	2A	*
0010.1011	43	2B	+
0010.1100	44	2C	,
0010.1101	45	2D	-
0010.1110	46	2E	.
0010.1111	47	2F	/
0011.0000	48	30	0
0011.0001	49	31	1
0011.0010	50	32	2
0011.0011	51	33	3
0011.0100	52	34	4
0011.0101	53	35	5
0011.0110	54	36	6
0011.0111	55	37	7
0011.1000	56	38	8
0011.1001	57	39	9
0100.0001	65	41	A
0100.0010	66	42	B
0100.0011	67	43	C
0100.0100	68	44	D
0100.0101	69	45	E
0100.0110	70	46	F
0100.0111	71	47	G
0100.1000	72	48	H
0100.1001	73	49	I
0100.1010	74	4A	J
0100.1011	75	4B	K
0100.1100	76	4C	L
0100.1101	77	4D	M
0100.1110	78	4E	N
0100.1111	79	4F	O
0101.0000	80	50	P
0101.0001	81	51	Q
0101.0010	82	52	R
0101.0011	83	53	S
0101.0100	84	54	T
0101.0101	85	55	U
0101.0110	86	56	V
0101.0111	87	57	W
0101.1000	88	58	X
0101.1001	89	59	Y
0101.1010	90	5A	Z
0110.0001	97	61	a
0110.0010	98	62	b
0110.0011	99	63	c
0110.0100	100	64	d
0110.0101	101	65	e
0110.0110	102	66	f
0110.0111	103	67	g
0110.1000	104	68	h
0110.1001	105	69	i
0110.1010	106	6A	j
0110.1011	107	6B	k
0110.1100	108	6C	l
0110.1101	109	6D	m
0110.1110	110	6E	n
0110.1111	111	6F	o
0111.0000	112	70	p
0111.0001	113	71	q
0111.0010	114	72	r
0111.0011	115	73	s
0111.0100	116	74	t
0111.0101	117	75	u
0111.0110	118	76	v
0111.0111	119	77	w
0111.1000	120	78	x
0111.1001	121	79	y
0111.1010	122	7A	z

Para efeito de conversão entre os 2 códigos (ASCII e Base64) deve-se usar o seguinte vetor de conversão:

```
ABCDEFGHIJKLMN
OPQRSTUVWXYZab
cdefghijklmnop
qrstuvwxyz0123
456789+/
```

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

ASCII → Base64

Deve-se pegar 3 bytes a converter (24 bits) e colocá-los lado a lado. Daí, o numerão binário obtido deve ser quebrado em 4 partes, cada uma com 6 bits. Como se sabe, um número binário de 6 bits varia entre 0..63. Este número obtido é que deve indexar o vetor acima estabelecido, dando como resultado 4 caracteres Base64,

Acompanhe no exemplo: Seja passar o ASCII `Aa`) para Base64. Da tabela ASCII, sabe-se que A=0100.0001, a=0110.0001 e)=0010.1001. Juntando tudo, fica: 0100.0001.0110.0001.0010.1001. Rearrumando, fica: 010000.010110.000100.101001, números binários que valem: 16, 22, 4 e 41, respectivamente. Obtendo os elementos 17, 23, 5 e 42 do vetor de elementos do Base64, fica-se com QWEp. Então: $Aa)_{ascii} \equiv QWEp_{Base64}$.

Um cuidado especial deve ser tomado quando se chega ao final do conjunto de caracteres a converter. No fim, podem sobrar nenhum, 1 ou 2 caracteres. Se sobrar nenhum, o problema está resolvido. Se sobrarem 2 caracteres, estes devem ser colocados lado a lado (16 bits) e o conjunto deve ser completado com 2 zeros à direita, formando os 18 bits esperados. Para sinalizar estes 2 *stuffing bits*, após os 3 caracteres Base64 gerados, agrega-se o sinal "=". Por exemplo, seja converter o string ASCII `A1B2C`. O primeiro conjunto `A1B` é convertido normalmente, gerando `QTFC`. Sobra `2C`. Suas configurações em binário são 2=0011.0010 e C=0100.0011. Juntando tudo, fica 0011.0010.0100.0011 cujo comprimento é 16 bits. Agregando 2 zeros à direita e reordenando fica: 0011.0010.0100.0011.00 ou 001100.100100.001100, que correspondem a 12, 36 e 12 respectivamente. Obtendo os elementos 13, 37 e 13 do vetor formador, obtém-se M, k e M respectivamente. A maneira de informar que foram incluídos 2 zeros ao final é colocar um "="como quarto caractere da conversão. Assim: $2C_{ascii} \equiv MkM=_{Base64}$.

Finalmente, se sobrar um único caractere (8 bits), o conversor precisa incluir 4 zeros à direita (*stuffing bits*). Por exemplo, para converter o ASCII "M"para Base64, faz-se: M=0100.1101, com 8 bits. Acrescentando os 4 zeros a direita, fica: 0100.1101.0000 e quebrando em dois números de 6 bits cada, fica: 010011.010000 que são os números 19 e 16. Obtendo o vigésimo e décimo sétimo elementos do vetor formador do Base64 obtem-se "TQ". Para sinalizar que incluiram-se 4 zeros, o conversor acrescenta "==". Com isso, $M_{ascii} \equiv TQ==_{Base64}$.

Base64 → ASCII

A volta segue o mesmo procedimento, agora realizado em sentido inverso. Inicialmente, deve-se verificar que o comprimento da mensagem codificada em Base64 deve ter um número de caracteres múltiplos de 4. Se não for, há um erro. Tudo começa, verificando se o último caractere do grupo (de 4) em análise é igual a "=". Se não for, este grupo não é terminal e deve sofrer o processamento padrão que é juntar os 24 bits (cada letra dá origem a um número de 6 bits, obtido pela localização de cada letra no vetor gerador do Base64) que devem ser reorganizados a fim de obter os 3 bytes em ASCII. Por exemplo, seja o conjunto Base64 "VGU4". As posições destes caracteres (V, G, U e 4) no vetor formador são: 22, 7, 21 e 57. Subtraindo uma unidade (já que agora a origem dos índices é zero) tem-se 21, 6, 20 e 56. Convertendo tais números a binário tem-se 0 1 0 1 0 1, 0 0 0 1 1 0, 0 1 0 1 0 0 e 1 1 1 0 0 0. Juntando e reorganizando 0101.0100 0110.0101 e 0011.1000, que são "Te8". Entao: $VGU4_{Base64} \equiv Te8_{ascii}$.

Se o último caracter é "=", então deve-se olhar o penúltimo. Se o penúltimo é diferente de "=", então tem-se aqui 2 caracteres ASCII, e depois de juntar 3 x 6 bits = 18 bits, deve-se desprezar os últimos 2 bits e ficar com 16 bits que perfazem os 2 caracteres ASCII buscados.

Por exemplo, seja o Base64 igual a "QTY=". Como o último é = e o penúltimo não é =, então este grupo codifica 2 caracteres ASCII.

Q, T e Y são os caracteres 17, 20 e 25 do vetor formador. Subtraindo 1 fica: 16, 19 e 24. Convertendo a binário fica: 0 1 0 0 0 0, 0 1 0 0 1 1 e 0 1 1 0 0 0. Juntando e desprezando os dois últimos zeros: 0100.0001, 0011.0110 e 00. Com isso os dois caracteres ASCII são: A6, que é a codificação original. Ou $QTY=_{Base64} \equiv A6_{ascii}$.

Se o último e o penúltimo forem =, então é apenas um o caracter original. Deve-se converter os 2 caracteres iniciais (os que são diferentes de =) formando 12 bits e desprezar os últimos 4 bits.

Por exemplo, para desconverter "Rg==", descobre-se que R e g são os elementos 18 e 33 no vetor formador. Subtraindo fica 17 e 32. Convertendo: 0 1 0 0 1 0 e 1 0 0 0 0 0. Reorganizando 0100.1010.0000. O caracter buscado é 'F'. Entao: $Rg==_{Base64} \equiv F_{ascii}$.

Alguns exemplos

de	conversão	para
tMz	ASCII→b64	dE16
Dx	ASCII→b64	RHg=
6	ASCII→b64	Ng==
ZGxO	b64→ASCII	dlN
cU4=	b64→ASCII	qN
SQ==	b64→ASCII	I

☞ Para você fazer

1. Converta de ASCII para Base64
   a) wE5
   b) oX
   c) B
   d) WnR
2. Converta de Base64 para ASCII
   e) ekpX
   f) Q2c=
   g) OA==
   h) bThw

a	b	c	d
e	f	g	h

202-76551 - IT/IR

U Positivo - UTFPR - PUCPr –
20/05/2024 - 10:31:53.4
Prof Dr P Kantek
(pkantek@gmail.com)
O esquema de codificação Base64
VIVX037a, V: 1.30 76568
GUILHERME VITOR DA SILVA
24CC2202 - 8 SUBSTITUIR e
entregar ate 30/maio _____ /
/ _____

Base64

Base64 é um sistema de numeração posicional que usa 64 como base. Ao produzir um texto que possa ser imprimível em qualquer impressora do mundo ocidental, sem nenhuma chance de erro ou incoerência, apenas podem ser usados os caracteres A..Z, a..z e 0..9. Note que tem-se aqui 26+26+10=62 caracteres. Para completar o conjunto esperado de 64 caracteres, costuma-se usar "+"e "/", nesta ordem. (Note-se que muitos programas que usam este princípio costumam escolher outros 2 caracteres que não sejam "+"e "/", mas nenhum deles usa o rótulo de Base64. São exemplos deste caso: a B64 modificado para URL (usa -"e "_") ou B64 para expressões regulares que usa ("!"e -").

Há ainda um caracter importante, o "="(em Base64) que serve para sinalizar eventuais necessidades de preechimento de *stuffing*.

A idéia desta codificação é poder mandar qualquer conteúdo binário como parte de uma mensagem de e-mail, sem correr o risco de algum conteúdo parcial acabe gerando caracteres de controle do protocolo e ponha a perder a comunicação.

A conversão se dá trocando 3 bytes (8 bits × 3 = 24 bits) em 4 caracteres Base64, cada um deles representado por 6 bits. (então 6 bits × 4 = 24 bits).

Na codificação de 8 bits/byte, usa-se um código qualquer. Atualmente a grande maioria de ambientes usa o padrão ASCII, cuja pequena parte pode ser vista aqui.

uma parte da Tabela ASCII

Binário	Dec	Hex	Car.
0010.1000	40	28	(
0010.1001	41	29	)
0010.1010	42	2A	*
0010.1011	43	2B	+
0010.1100	44	2C	,
0010.1101	45	2D	-
0010.1110	46	2E	.
0010.1111	47	2F	/
0011.0000	48	30	0
0011.0001	49	31	1
0011.0010	50	32	2
0011.0011	51	33	3
0011.0100	52	34	4
0011.0101	53	35	5
0011.0110	54	36	6
0011.0111	55	37	7
0011.1000	56	38	8
0011.1001	57	39	9
0100.0001	65	41	A
0100.0010	66	42	B
0100.0011	67	43	C
0100.0100	68	44	D
0100.0101	69	45	E
0100.0110	70	46	F
0100.0111	71	47	G
0100.1000	72	48	H
0100.1001	73	49	I
0100.1010	74	4A	J
0100.1011	75	4B	K
0100.1100	76	4C	L
0100.1101	77	4D	M
0100.1110	78	4E	N
0100.1111	79	4F	O
0101.0000	80	50	P
0101.0001	81	51	Q
0101.0010	82	52	R
0101.0011	83	53	S
0101.0100	84	54	T
0101.0101	85	55	U
0101.0110	86	56	V
0101.0111	87	57	W
0101.1000	88	58	X
0101.1001	89	59	Y
0101.1010	90	5A	Z
0110.0001	97	61	a
0110.0010	98	62	b
0110.0011	99	63	c
0110.0100	100	64	d
0110.0101	101	65	e
0110.0110	102	66	f
0110.0111	103	67	g
0110.1000	104	68	h
0110.1001	105	69	i
0110.1010	106	6A	j
0110.1011	107	6B	k
0110.1100	108	6C	l
0110.1101	109	6D	m
0110.1110	110	6E	n
0110.1111	111	6F	o
0111.0000	112	70	p
0111.0001	113	71	q
0111.0010	114	72	r
0111.0011	115	73	s
0111.0100	116	74	t
0111.0101	117	75	u
0111.0110	118	76	v
0111.0111	119	77	w
0111.1000	120	78	x
0111.1001	121	79	y
0111.1010	122	7A	z

Para efeito de conversão entre os 2 códigos (ASCII e Base64) deve-se usar o seguinte vetor de conversão:

```
ABCDEFGHIJKLMN
OPQRSTUVWXYZab
cdefghijklmnop
qrstuvwxyz0123
456789+/
```

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

ASCII → Base64

Deve-se pegar 3 bytes a converter (24 bits) e colocá-los lado a lado. Daí, o numerão binário obtido deve ser quebrado em 4 partes, cada uma com 6 bits. Como se sabe, um número binário de 6 bits varia entre 0..63. Este número obtido é que deve indexar o vetor acima estabelecido, dando como resultado 4 caracteres Base64,

Acompanhe no exemplo: Seja passar o ASCII `Aa`) para Base64. Da tabela ASCII, sabe-se que A=0100.0001, a=0110.0001 e)=0010.1001. Juntando tudo, fica: 0100.0001.0110.0001.0010.1001. Rearrumando, fica: 010000.010110.000100.101001, números binários que valem: 16, 22, 4 e 41, respectivamente. Obtendo os elementos 17, 23, 5 e 42 do vetor de elementos do Base64, fica-se com QWEp. Então: $Aa)_{ascii} \equiv QWEp_{Base64}$.

Um cuidado especial deve ser tomado quando se chega ao final do conjunto de caracteres a converter. No fim, podem sobrar nenhum, 1 ou 2 caracteres. Se sobrar nenhum, o problema está resolvido. Se sobrarem 2 caracteres, estes devem ser colocados lado a lado (16 bits) e o conjunto deve ser completado com 2 zeros à direita, formando os 18 bits esperados. Para sinalizar estes 2 *stuffing bits*, após os 3 caracteres Base64 gerados, agrega-se o sinal "=". Por exemplo, seja converter o string ASCII `A1B2C`. O primeiro conjunto `A1B` é convertido normalmente, gerando `QTFC`. Sobra `2C`. Suas configurações em binário são 2=0011.0010 e C=0100.0011. Juntando tudo, fica 0011.0010.0100.0011 cujo comprimento é 16 bits. Agregando 2 zeros à direita e reordenando fica: 0011.0010.0100.0011.00 ou 001100.100100.001100, que correspondem a 12, 36 e 12 respectivamente. Obtendo os elementos 13, 37 e 13 do vetor formador, obtém-se M, k e M respectivamente. A maneira de informar que foram incluídos 2 zeros ao final é colocar um "="como quarto caractere da conversão. Assim: $2C_{ascii} \equiv MkM=_{Base64}$.

Finalmente, se sobrar um único caractere (8 bits), o conversor precisa incluir 4 zeros à direita (*stuffing bits*). Por exemplo, para converter o ASCII "M"para Base64, faz-se: M=0100.1101, com 8 bits. Acrescentando os 4 zeros a direita, fica: 0100.1101.0000 e quebrando em dois números de 6 bits cada, fica: 010011.010000 que são os números 19 e 16. Obtendo o vigésimo e décimo sétimo elementos do vetor formador do Base64 obtem-se "TQ". Para sinalizar que incluiram-se 4 zeros, o conversor acrescenta "==". Com isso, $M_{ascii} \equiv TQ==_{Base64}$.

Base64 → ASCII

A volta segue o mesmo procedimento, agora realizado em sentido inverso. Inicialmente, deve-se verificar que o comprimento da mensagem codificada em Base64 deve ter um número de caracteres múltiplos de 4. Se não for, há um erro. Tudo começa, verificando se o último caractere do grupo (de 4) em análise é igual a "=". Se não for, este grupo não é terminal e deve sofrer o processamento padrão que é juntar os 24 bits (cada letra dá origem a um número de 6 bits, obtido pela localização de cada letra no vetor gerador do Base64) que devem ser reorganizados a fim de obter os 3 bytes em ASCII. Por exemplo, seja o conjunto Base64 "VGU4". As posições destes caracteres (V, G, U e 4) no vetor formador são: 22, 7, 21 e 57. Subtraindo uma unidade (já que agora a origem dos índices é zero) tem-se 21, 6, 20 e 56. Convertendo tais números a binário tem-se 0 1 0 1 0 1, 0 0 0 1 1 0, 0 1 0 1 0 0 e 1 1 1 0 0 0. Juntando e reorganizando 0101.0100 0110.0101 e 0011.1000, que são "Te8". Entao: $VGU4_{Base64} \equiv Te8_{ascii}$.

Se o último caracter é "=", então deve-se olhar o penúltimo. Se o penúltimo é diferente de "=", então tem-se aqui 2 caracteres ASCII, e depois de juntar 3 x 6 bits = 18 bits, deve-se desprezar os últimos 2 bits e ficar com 16 bits que perfazem os 2 caracteres ASCII buscados.

Por exemplo, seja o Base64 igual a "QTY=". Como o último é = e o penúltimo não é =, então este grupo codifica 2 caracteres ASCII.

Q, T e Y são os caracteres 17, 20 e 25 do vetor formador. Subtraindo 1 fica: 16, 19 e 24. Convertendo a binário fica: 0 1 0 0 0 0, 0 1 0 0 1 1 e 0 1 1 0 0 0. Juntando e desprezando os dois últimos zeros: 0100.0001, 0011.0110 e 00. Com isso os dois caracteres ASCII são: A6, que é a codificação original. Ou $QTY=_{Base64} \equiv A6_{ascii}$.

Se o último e o penúltimo forem =, então é apenas um o caracter original. Deve-se converter os 2 caracteres iniciais (os que são diferentes de =) formando 12 bits e desprezar os últimos 4 bits.

Por exemplo, para desconverter "Rg==", descobre-se que R e g são os elementos 18 e 33 no vetor formador. Subtraindo fica 17 e 32. Convertendo: 0 1 0 0 0 1 0 e 1 0 0 0 0 0. Reorganizando 0100.1010.0000. O caracter buscado é 'F'. Entao: $Rg==_{Base64} \equiv F_{ascii}$.

Alguns exemplos

de	conversão	para
tMz	ASCII→b64	dE16
Dx	ASCII→b64	RHg=
6	ASCII→b64	Ng==
ZGxO	b64→ASCII	dlN
cU4=	b64→ASCII	qN
SQ==	b64→ASCII	I

☞ Para você fazer

1. Converta de ASCII para Base64
   a) wT3
   b) 9y
   c) H
   d) Lw/
2. Converta de Base64 para ASCII
   e) dWxC
   f) dmU=
   g) Tw==
   h) UFFl

a	b	c	d
e	f	g	h

U Positivo - UTFPR - PUCPr –
20/05/2024 - 10:31:53.4
Prof Dr P Kantek
(pkantek@gmail.com)
O esquema de codificação Base64
VIVX037a, V: 1.30 76575 LAURA
PACHECO DELATTRE 24CC2202
- 9 SUBSTITUIR e entregar ate
30/maio _____ / _____ / _____

Base64

Base64 é um sistema de numeração posicional que usa 64 como base. Ao produzir um texto que possa ser imprimível em qualquer impressora do mundo ocidental, sem nenhuma chance de erro ou incoerência, apenas podem ser usados os caracteres A..Z, a..z e 0..9. Note que tem-se aqui 26+26+10=62 caracteres. Para completar o conjunto esperado de 64 caracteres, costuma-se usar "+"e "/", nesta ordem. (Note-se que muitos programas que usam este princípio costumam escolher outros 2 caracteres que não sejam "+"e "/", mas nenhum deles usa o rótulo de Base64. São exemplos deste caso: a B64 modificado para URL (usa -"e "_") ou B64 para expressões regulares que usa ("!"e -").

Há ainda um caracter importante, o "="(em Base64) que serve para sinalizar eventuais necessidades de preechimento de *stuffing*.

A idéia desta codificação é poder mandar qualquer conteúdo binário como parte de uma mensagem de e-mail, sem correr o risco de algum conteúdo parcial acabe gerando caracteres de controle do protocolo e ponha a perder a comunicação.

A conversão se dá trocando 3 bytes (8 bits × 3 = 24 bits) em 4 caracteres Base64, cada um deles representado por 6 bits. (então 6 bits × 4 = 24 bits).

Na codificação de 8 bits/byte, usa-se um código qualquer. Atualmente a grande maioria de ambientes usa o padrão ASCII, cuja pequena parte pode ser vista aqui.

uma parte da Tabela ASCII

Binário	Dec	Hex	Car.
0010.1000	40	28	(
0010.1001	41	29	)
0010.1010	42	2A	*
0010.1011	43	2B	+
0010.1100	44	2C	,
0010.1101	45	2D	-
0010.1110	46	2E	.
0010.1111	47	2F	/
0011.0000	48	30	0
0011.0001	49	31	1
0011.0010	50	32	2
0011.0011	51	33	3
0011.0100	52	34	4
0011.0101	53	35	5
0011.0110	54	36	6
0011.0111	55	37	7
0011.1000	56	38	8
0011.1001	57	39	9
0100.0001	65	41	A
0100.0010	66	42	B
0100.0011	67	43	C
0100.0100	68	44	D
0100.0101	69	45	E
0100.0110	70	46	F
0100.0111	71	47	G
0100.1000	72	48	H
0100.1001	73	49	I
0100.1010	74	4A	J
0100.1011	75	4B	K
0100.1100	76	4C	L
0100.1101	77	4D	M
0100.1110	78	4E	N
0100.1111	79	4F	O
0101.0000	80	50	P
0101.0001	81	51	Q
0101.0010	82	52	R
0101.0011	83	53	S
0101.0100	84	54	T
0101.0101	85	55	U
0101.0110	86	56	V
0101.0111	87	57	W
0101.1000	88	58	X
0101.1001	89	59	Y
0101.1010	90	5A	Z
0110.0001	97	61	a
0110.0010	98	62	b
0110.0011	99	63	c
0110.0100	100	64	d
0110.0101	101	65	e
0110.0110	102	66	f
0110.0111	103	67	g
0110.1000	104	68	h
0110.1001	105	69	i
0110.1010	106	6A	j
0110.1011	107	6B	k
0110.1100	108	6C	l
0110.1101	109	6D	m
0110.1110	110	6E	n
0110.1111	111	6F	o
0111.0000	112	70	p
0111.0001	113	71	q
0111.0010	114	72	r
0111.0011	115	73	s
0111.0100	116	74	t
0111.0101	117	75	u
0111.0110	118	76	v
0111.0111	119	77	w
0111.1000	120	78	x
0111.1001	121	79	y
0111.1010	122	7A	z

Para efeito de conversão entre os 2 códigos (ASCII e Base64) deve-se usar o seguinte vetor de conversão:

```
ABCDEFGHIJKLMN
OPQRSTUVWXYZab
cdefghijklmnop
qrstuvwxyz0123
456789+/
```

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

ASCII → Base64

Deve-se pegar 3 bytes a converter (24 bits) e colocá-los lado a lado. Daí, o numerão binário obtido deve ser quebrado em 4 partes, cada uma com 6 bits. Como se sabe, um número binário de 6 bits varia entre 0..63. Este número obtido é que deve indexar o vetor acima estabelecido, dando como resultado 4 caracteres Base64,

Acompanhe no exemplo: Seja passar o ASCII `Aa`) para Base64. Da tabela ASCII, sabe-se que A=0100.0001, a=0110.0001 e)=0010.1001. Juntando tudo, fica: 0100.0001.0110.0001.0010.1001. Rearrumando, fica: 010000.010110.000100.101001, números binários que valem: 16, 22, 4 e 41, respectivamente. Obtendo os elementos 17, 23, 5 e 42 do vetor de elementos do Base64, fica-se com QWEp. Então: $Aa)_{ascii} \equiv QWEp_{Base64}$.

Um cuidado especial deve ser tomado quando se chega ao final do conjunto de caracteres a converter. No fim, podem sobrar nenhum, 1 ou 2 caracteres. Se sobrar nenhum, o problema está resolvido. Se sobrarem 2 caracteres, estes devem ser colocados lado a lado (16 bits) e o conjunto deve ser completado com 2 zeros à direita, formando os 18 bits esperados. Para sinalizar estes 2 *stuffing bits*, após os 3 caracteres Base64 gerados, agrega-se o sinal "=". Por exemplo, seja converter o string ASCII `A1B2C`. O primeiro conjunto `A1B` é convertido normalmente, gerando `QTFC`. Sobra `2C`. Suas configurações em binário são 2=0011.0010 e C=0100.0011. Juntando tudo, fica 0011.0010.0100.0011 cujo comprimento é 16 bits. Agregando 2 zeros à direita e reordenando fica: 0011.0010.0100.0011.00 ou 001100.100100.001100, que correspondem a 12, 36 e 12 respectivamente. Obtendo os elementos 13, 37 e 13 do vetor formador, obtém-se M, k e M respectivamente. A maneira de informar que foram incluídos 2 zeros ao final é colocar um "="como quarto caractere da conversão. Assim: $2C_{ascii} \equiv MkM=_{Base64}$.

Finalmente, se sobrar um único caractere (8 bits), o conversor precisa incluir 4 zeros à direita (*stuffing bits*). Por exemplo, para converter o ASCII "M"para Base64, faz-se: M=0100.1101, com 8 bits. Acrescentando os 4 zeros a direita, fica: 0100.1101.0000 e quebrando em dois números de 6 bits cada, fica: 010011.010000 que são os números 19 e 16. Obtendo o vigésimo e décimo sétimo elementos do vetor formador do Base64 obtem-se "TQ". Para sinalizar que incluiram-se 4 zeros, o conversor acrescenta "==". Com isso, $M_{ascii} \equiv TQ==_{Base64}$.

Base64 → ASCII

A volta segue o mesmo procedimento, agora realizado em sentido inverso. Inicialmente, deve-se verificar que o comprimento da mensagem codificada em Base64 deve ter um número de caracteres múltiplos de 4. Se não for, há um erro. Tudo começa, verificando se o último caractere do grupo (de 4) em análise é igual a "=". Se não for, este grupo não é terminal e deve sofrer o processamento padrão que é juntar os 24 bits (cada letra dá origem a um número de 6 bits, obtido pela localização de cada letra no vetor gerador do Base64) que devem ser reorganizados a fim de obter os 3 bytes em ASCII. Por exemplo, seja o conjunto Base64 "VGU4". As posições destes caracteres (V, G, U e 4) no vetor formador são: 22, 7, 21 e 57. Subtraindo uma unidade (já que agora a origem dos índices é zero) tem-se 21, 6, 20 e 56. Convertendo tais números a binário tem-se 0 1 0 1 0 1, 0 0 0 1 1 0, 0 1 0 1 0 0 e 1 1 1 0 0 0. Juntando e reorganizando 0101.0100 0110.0101 e 0011.1000, que são "Te8". Entao: $VGU4_{Base64} \equiv Te8_{ascii}$.

Se o último caracter é "=", então deve-se olhar o penúltimo. Se o penúltimo é diferente de "=", então tem-se aqui 2 caracteres ASCII, e depois de juntar 3 x 6 bits = 18 bits, deve-se desprezar os últimos 2 bits e ficar com 16 bits que perfazem os 2 caracteres ASCII buscados.

Por exemplo, seja o Base64 igual a "QTY=". Como o último é = e o penúltimo não é =, então este grupo codifica 2 caracteres ASCII.

Q, T e Y são os caracteres 17, 20 e 25 do vetor formador. Subtraindo 1 fica: 16, 19 e 24. Convertendo a binário fica: 0 1 0 0 0 0, 0 1 0 0 1 1 e 0 1 1 0 0 0. Juntando e desprezando os dois últimos zeros: 0100.0001, 0011.0110 e 00. Com isso os dois caracteres ASCII são: A6, que é a codificação original. Ou $QTY=_{Base64} \equiv A6_{ascii}$.

Se o último e o penúltimo forem =, então é apenas um o caracter original. Deve-se converter os 2 caracteres iniciais (os que são diferentes de =) formando 12 bits e desprezar os últimos 4 bits.

Por exemplo, para desconverter "Rg==", descobre-se que R e g são os elementos 18 e 33 no vetor formador. Subtraindo fica 17 e 32. Convertendo: 0 1 0 0 1 0 e 1 0 0 0 0 0. Reorganizando 0100.1010.0000. O caracter buscado é 'F'. Entao: $Rg==_{Base64} \equiv F_{ascii}$.

Alguns exemplos

de	conversão	para
tMz	ASCII→b64	dE16
Dx	ASCII→b64	RHg=
6	ASCII→b64	Ng==
ZGxO	b64→ASCII	dlN
cU4=	b64→ASCII	qN
SQ==	b64→ASCII	I

☞ Para você fazer

1. Converta de ASCII para Base64
   a) rkm
   b) zh
   c) P
   d) GlU
2. Converta de Base64 para ASCII
   e) eldY
   f) WUE=
   g) Tw==
   h) VnF3

a	b	c	d
e	f	g	h

202-76575 - IT/IR

U Positivo - UTFPR - PUCPr –
20/05/2024 - 10:31:53.4
Prof Dr P Kantek
(pkantek@gmail.com)
O esquema de codificação Base64
VIVX037a, V: 1.30 76582 LUIZ
MIGUEL OLINEK 24CC2202 - 10
SUBSTITUIR e entregar ate
30/maio _____ / _____ / _____

Base64

Base64 é um sistema de numeração posicional que usa 64 como base. Ao produzir um texto que possa ser imprimível em qualquer impressora do mundo ocidental, sem nenhuma chance de erro ou incoerência, apenas podem ser usados os caracteres A..Z, a..z e 0..9. Note que tem-se aqui 26+26+10=62 caracteres. Para completar o conjunto esperado de 64 caracteres, costuma-se usar "+"e "/", nesta ordem. (Note-se que muitos programas que usam este princípio costumam escolher outros 2 caracteres que não sejam "+"e "/", mas nenhum deles usa o rótulo de Base64. São exemplos deste caso: a B64 modificado para URL (usa -"e "_") ou B64 para expressões regulares que usa ("!"e -").

Há ainda um caracter importante, o "="(em Base64) que serve para sinalizar eventuais necessidades de preechimento de *stuffing*.

A idéia desta codificação é poder mandar qualquer conteúdo binário como parte de uma mensagem de e-mail, sem correr o risco de algum conteúdo parcial acabe gerando caracteres de controle do protocolo e ponha a perder a comunicação.

A conversão se dá trocando 3 bytes (8 bits × 3 = 24 bits) em 4 caracteres Base64, cada um deles representado por 6 bits. (então 6 bits × 4 = 24 bits).

Na codificação de 8 bits/byte, usa-se um código qualquer. Atualmente a grande maioria de ambientes usa o padrão ASCII, cuja pequena parte pode ser vista aqui.

uma parte da Tabela ASCII

Binário	Dec	Hex	Car.
0010.1000	40	28	(
0010.1001	41	29	)
0010.1010	42	2A	*
0010.1011	43	2B	+
0010.1100	44	2C	,
0010.1101	45	2D	-
0010.1110	46	2E	.
0010.1111	47	2F	/
0011.0000	48	30	0
0011.0001	49	31	1
0011.0010	50	32	2
0011.0011	51	33	3
0011.0100	52	34	4
0011.0101	53	35	5
0011.0110	54	36	6
0011.0111	55	37	7
0011.1000	56	38	8
0011.1001	57	39	9
0100.0001	65	41	A
0100.0010	66	42	B
0100.0011	67	43	C
0100.0100	68	44	D
0100.0101	69	45	E
0100.0110	70	46	F
0100.0111	71	47	G
0100.1000	72	48	H
0100.1001	73	49	I
0100.1010	74	4A	J
0100.1011	75	4B	K
0100.1100	76	4C	L
0100.1101	77	4D	M
0100.1110	78	4E	N
0100.1111	79	4F	O
0101.0000	80	50	P
0101.0001	81	51	Q
0101.0010	82	52	R
0101.0011	83	53	S
0101.0100	84	54	T
0101.0101	85	55	U
0101.0110	86	56	V
0101.0111	87	57	W
0101.1000	88	58	X
0101.1001	89	59	Y
0101.1010	90	5A	Z
0110.0001	97	61	a
0110.0010	98	62	b
0110.0011	99	63	c
0110.0100	100	64	d
0110.0101	101	65	e
0110.0110	102	66	f
0110.0111	103	67	g
0110.1000	104	68	h
0110.1001	105	69	i
0110.1010	106	6A	j
0110.1011	107	6B	k
0110.1100	108	6C	l
0110.1101	109	6D	m
0110.1110	110	6E	n
0110.1111	111	6F	o
0111.0000	112	70	p
0111.0001	113	71	q
0111.0010	114	72	r
0111.0011	115	73	s
0111.0100	116	74	t
0111.0101	117	75	u
0111.0110	118	76	v
0111.0111	119	77	w
0111.1000	120	78	x
0111.1001	121	79	y
0111.1010	122	7A	z

Para efeito de conversão entre os 2 códigos (ASCII e Base64) deve-se usar o seguinte vetor de conversão:

```
ABCDEFGHIJKLMN
OPQRSTUVWXYZab
cdefghijklmnop
qrstuvwxyz0123
456789+/
```

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

ASCII → Base64

Deve-se pegar 3 bytes a converter (24 bits) e colocá-los lado a lado. Daí, o numerão binário obtido deve ser quebrado em 4 partes, cada uma com 6 bits. Como se sabe, um número binário de 6 bits varia entre 0..63. Este número obtido é que deve indexar o vetor acima estabelecido, dando como resultado 4 caracteres Base64,

Acompanhe no exemplo: Seja passar o ASCII `Aa`) para Base64. Da tabela ASCII, sabe-se que A=0100.0001, a=0110.0001 e)=0010.1001. Juntando tudo, fica: 0100.0001.0110.0001.0010.1001. Rearrumando, fica: 010000.010110.000100.101001, números binários que valem: 16, 22, 4 e 41, respectivamente. Obtendo os elementos 17, 23, 5 e 42 do vetor de elementos do Base64, fica-se com QWEp. Então: $Aa)_{ascii} \equiv QWEp_{Base64}$.

Um cuidado especial deve ser tomado quando se chega ao final do conjunto de caracteres a converter. No fim, podem sobrar nenhum, 1 ou 2 caracteres. Se sobrar nenhum, o problema está resolvido. Se sobrarem 2 caracteres, estes devem ser colocados lado a lado (16 bits) e o conjunto deve ser completado com 2 zeros à direita, formando os 18 bits esperados. Para sinalizar estes 2 *stuffing bits*, após os 3 caracteres Base64 gerados, agrega-se o sinal "=". Por exemplo, seja converter o string ASCII `A1B2C`. O primeiro conjunto `A1B` é convertido normalmente, gerando `QTFC`. Sobra `2C`. Suas configurações em binário são 2=0011.0010 e C=0100.0011. Juntando tudo, fica 0011.0010.0100.0011 cujo comprimento é 16 bits. Agregando 2 zeros à direita e reordenando fica: 0011.0010.0100.0011.00 ou 001100.100100.001100, que correspondem a 12, 36 e 12 respectivamente. Obtendo os elementos 13, 37 e 13 do vetor formador, obtém-se M, k e M respectivamente. A maneira de informar que foram incluídos 2 zeros ao final é colocar um "="como quarto caractere da conversão. Assim: $2C_{ascii} \equiv MkM=_{Base64}$.

Finalmente, se sobrar um único caractere (8 bits), o conversor precisa incluir 4 zeros à direita (*stuffing bits*). Por exemplo, para converter o ASCII "M"para Base64, faz-se: M=0100.1101, com 8 bits. Acrescentando os 4 zeros a direita, fica: 0100.1101.0000 e quebrando em dois números de 6 bits cada, fica: 010011.010000 que são os números 19 e 16. Obtendo o vigésimo e décimo sétimo elementos do vetor formador do Base64 obtem-se "TQ". Para sinalizar que incluiram-se 4 zeros, o conversor acrescenta "==". Com isso, $M_{ascii} \equiv TQ==_{Base64}$.

Base64 → ASCII

A volta segue o mesmo procedimento, agora realizado em sentido inverso. Inicialmente, deve-se verificar que o comprimento da mensagem codificada em Base64 deve ter um número de caracteres múltiplos de 4. Se não for, há um erro. Tudo começa, verificando se o último caractere do grupo (de 4) em análise é igual a "=". Se não for, este grupo não é terminal e deve sofrer o processamento padrão que é juntar os 24 bits (cada letra dá origem a um número de 6 bits, obtido pela localização de cada letra no vetor gerador do Base64) que devem ser reorganizados a fim de obter os 3 bytes em ASCII. Por exemplo, seja o conjunto Base64 "VGU4". As posições destes caracteres (V, G, U e 4) no vetor formador são: 22, 7, 21 e 57. Subtraindo uma unidade (já que agora a origem dos índices é zero) tem-se 21, 6, 20 e 56. Convertendo tais números a binário tem-se 0 1 0 1 0 1, 0 0 0 1 1 0, 0 1 0 1 0 0 e 1 1 1 0 0 0. Juntando e reorganizando 0101.0100 0110.0101 e 0011.1000, que são "Te8". Entao: $VGU4_{Base64} \equiv Te8_{ascii}$.

Se o último caracter é "=", então deve-se olhar o penúltimo. Se o penúltimo é diferente de "=", então tem-se aqui 2 caracteres ASCII, e depois de juntar 3 x 6 bits = 18 bits, deve-se desprezar os últimos 2 bits e ficar com 16 bits que perfazem os 2 caracteres ASCII buscados.

Por exemplo, seja o Base64 igual a "QTY=". Como o último é = e o penúltimo não é =, então este grupo codifica 2 caracteres ASCII.

Q, T e Y são os caracteres 17, 20 e 25 do vetor formador. Subtraindo 1 fica: 16, 19 e 24. Convertendo a binário fica: 0 1 0 0 0 0, 0 1 0 0 1 1 e 0 1 1 0 0 0. Juntando e desprezando os dois últimos zeros: 0100.0001, 0011.0110 e 00. Com isso os dois caracteres ASCII são: A6, que é a codificação original. Ou $QTY=_{Base64} \equiv A6_{ascii}$.

Se o último e o penúltimo forem =, então é apenas um o caracter original. Deve-se converter os 2 caracteres iniciais (os que são diferentes de =) formando 12 bits e desprezar os últimos 4 bits.

Por exemplo, para desconverter "Rg==", descobre-se que R e g são os elementos 18 e 33 no vetor formador. Subtraindo fica 17 e 32. Convertendo: 0 1 0 0 1 0 e 1 0 0 0 0 0. Reorganizando 0100.1010.0000. O caracter buscado é 'F'. Entao: $Rg==_{Base64} \equiv F_{ascii}$.

Alguns exemplos

de	conversão	para
tMz	ASCII→b64	dE16
Dx	ASCII→b64	RHg=
6	ASCII→b64	Ng==
ZGxO	b64→ASCII	dlN
cU4=	b64→ASCII	qN
SQ==	b64→ASCII	I

☞ Para você fazer

1. Converta de ASCII para Base64
   a) Pn1
   b) 5p
   c) C
   d) dQV
2. Converta de Base64 para ASCII
   e) VitB
   f) TjI=
   g) SA==
   h) WGRa

a	b	c	d
e	f	g	h

202-76582 - IT/IR

U Positivo - UTFPR - PUCPr –
20/05/2024 - 10:31:53.4
Prof Dr P Kantek
(pkantek@gmail.com)
O esquema de codificação Base64
VIVX037a, V: 1.30 76599 NICOLE
DA SILVA DE AZEVEDO
24CC2202 - 11 SUBSTITUIR e
entregar ate 30/maio _____ /
/ _____

Base64

Base64 é um sistema de numeração posicional que usa 64 como base. Ao produzir um texto que possa ser imprimível em qualquer impressora do mundo ocidental, sem nenhuma chance de erro ou incoerência, apenas podem ser usados os caracteres A..Z, a..z e 0..9. Note que tem-se aqui 26+26+10=62 caracteres. Para completar o conjunto esperado de 64 caracteres, costuma-se usar "+"e "/", nesta ordem. (Note-se que muitos programas que usam este princípio costumam escolher outros 2 caracteres que não sejam "+"e "/", mas nenhum deles usa o rótulo de Base64. São exemplos deste caso: a B64 modificado para URL (usa -"e "_") ou B64 para expressões regulares que usa ("!"e -").

Há ainda um caracter importante, o "="(em Base64) que serve para sinalizar eventuais necessidades de preechimento de *stuffing*.

A idéia desta codificação é poder mandar qualquer conteúdo binário como parte de uma mensagem de e-mail, sem correr o risco de algum conteúdo parcial acabe gerando caracteres de controle do protocolo e ponha a perder a comunicação.

A conversão se dá trocando 3 bytes (8 bits × 3 = 24 bits) em 4 caracteres Base64, cada um deles representado por 6 bits. (então 6 bits × 4 = 24 bits).

Na codificação de 8 bits/byte, usa-se um código qualquer. Atualmente a grande maioria de ambientes usa o padrão ASCII, cuja pequena parte pode ser vista aqui.

uma parte da Tabela ASCII

Binário	Dec	Hex	Car.
0010.1000	40	28	(
0010.1001	41	29	)
0010.1010	42	2A	*
0010.1011	43	2B	+
0010.1100	44	2C	,
0010.1101	45	2D	-
0010.1110	46	2E	.
0010.1111	47	2F	/
0011.0000	48	30	0
0011.0001	49	31	1
0011.0010	50	32	2
0011.0011	51	33	3
0011.0100	52	34	4
0011.0101	53	35	5
0011.0110	54	36	6
0011.0111	55	37	7
0011.1000	56	38	8
0011.1001	57	39	9
0100.0001	65	41	A
0100.0010	66	42	B
0100.0011	67	43	C
0100.0100	68	44	D
0100.0101	69	45	E
0100.0110	70	46	F
0100.0111	71	47	G
0100.1000	72	48	H
0100.1001	73	49	I
0100.1010	74	4A	J
0100.1011	75	4B	K
0100.1100	76	4C	L
0100.1101	77	4D	M
0100.1110	78	4E	N
0100.1111	79	4F	O
0101.0000	80	50	P
0101.0001	81	51	Q
0101.0010	82	52	R
0101.0011	83	53	S
0101.0100	84	54	T
0101.0101	85	55	U
0101.0110	86	56	V
0101.0111	87	57	W
0101.1000	88	58	X
0101.1001	89	59	Y
0101.1010	90	5A	Z
0110.0001	97	61	a
0110.0010	98	62	b
0110.0011	99	63	c
0110.0100	100	64	d
0110.0101	101	65	e
0110.0110	102	66	f
0110.0111	103	67	g
0110.1000	104	68	h
0110.1001	105	69	i
0110.1010	106	6A	j
0110.1011	107	6B	k
0110.1100	108	6C	l
0110.1101	109	6D	m
0110.1110	110	6E	n
0110.1111	111	6F	o
0111.0000	112	70	p
0111.0001	113	71	q
0111.0010	114	72	r
0111.0011	115	73	s
0111.0100	116	74	t
0111.0101	117	75	u
0111.0110	118	76	v
0111.0111	119	77	w
0111.1000	120	78	x
0111.1001	121	79	y
0111.1010	122	7A	z

Para efeito de conversão entre os 2 códigos (ASCII e Base64) deve-se usar o seguinte vetor de conversão:

```
ABCDEFGHIJKLMN
OPQRSTUVWXYZab
cdefghijklmnop
qrstuvwxyz0123
456789+/
```

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

ASCII → Base64

Deve-se pegar 3 bytes a converter (24 bits) e colocá-los lado a lado. Daí, o numerão binário obtido deve ser quebrado em 4 partes, cada uma com 6 bits. Como se sabe, um número binário de 6 bits varia entre 0..63. Este número obtido é que deve indexar o vetor acima estabelecido, dando como resultado 4 caracteres Base64,

Acompanhe no exemplo: Seja passar o ASCII `Aa`) para Base64. Da tabela ASCII, sabe-se que A=0100.0001, a=0110.0001 e)=0010.1001. Juntando tudo, fica: 0100.0001.0110.0001.0010.1001. Rearrumando, fica: 010000.010110.000100.101001, números binários que valem: 16, 22, 4 e 41, respectivamente. Obtendo os elementos 17, 23, 5 e 42 do vetor de elementos do Base64, fica-se com QWEp. Então: $Aa)_{ascii} \equiv QWEp_{Base64}$.

Um cuidado especial deve ser tomado quando se chega ao final do conjunto de caracteres a converter. No fim, podem sobrar nenhum, 1 ou 2 caracteres. Se sobrar nenhum, o problema está resolvido. Se sobrarem 2 caracteres, estes devem ser colocados lado a lado (16 bits) e o conjunto deve ser completado com 2 zeros à direita, formando os 18 bits esperados. Para sinalizar estes 2 *stuffing bits*, após os 3 caracteres Base64 gerados, agrega-se o sinal "=". Por exemplo, seja converter o string ASCII `A1B2C`. O primeiro conjunto `A1B` é convertido normalmente, gerando `QTFC`. Sobra `2C`. Suas configurações em binário são 2=0011.0010 e C=0100.0011. Juntando tudo, fica 0011.0010.0100.0011 cujo comprimento é 16 bits. Agregando 2 zeros à direita e reordenando fica: 0011.0010.0100.0011.00 ou 001100.100100.001100, que correspondem a 12, 36 e 12 respectivamente. Obtendo os elementos 13, 37 e 13 do vetor formador, obtém-se M, k e M respectivamente. A maneira de informar que foram incluídos 2 zeros ao final é colocar um "="como quarto caractere da conversão. Assim: $2C_{ascii} \equiv MkM=_{Base64}$.

Finalmente, se sobrar um único caractere (8 bits), o conversor precisa incluir 4 zeros à direita (*stuffing bits*). Por exemplo, para converter o ASCII "M"para Base64, faz-se: M=0100.1101, com 8 bits. Acrescentando os 4 zeros a direita, fica: 0100.1101.0000 e quebrando em dois números de 6 bits cada, fica: 010011.010000 que são os números 19 e 16. Obtendo o vigésimo e décimo sétimo elementos do vetor formador do Base64 obtem-se "TQ". Para sinalizar que incluiram-se 4 zeros, o conversor acrescenta "==". Com isso, $M_{ascii} \equiv TQ==_{Base64}$.

Base64 → ASCII

A volta segue o mesmo procedimento, agora realizado em sentido inverso. Inicialmente, deve-se verificar que o comprimento da mensagem codificada em Base64 deve ter um número de caracteres múltiplos de 4. Se não for, há um erro. Tudo começa, verificando se o último caractere do grupo (de 4) em análise é igual a "=". Se não for, este grupo não é terminal e deve sofrer o processamento padrão que é juntar os 24 bits (cada letra dá origem a um número de 6 bits, obtido pela localização de cada letra no vetor gerador do Base64) que devem ser reorganizados a fim de obter os 3 bytes em ASCII. Por exemplo, seja o conjunto Base64 "VGU4". As posições destes caracteres (V, G, U e 4) no vetor formador são: 22, 7, 21 e 57. Subtraindo uma unidade (já que agora a origem dos índices é zero) tem-se 21, 6, 20 e 56. Convertendo tais números a binário tem-se 0 1 0 1 0 1, 0 0 0 1 1 0, 0 1 0 1 0 0 e 1 1 1 0 0 0. Juntando e reorganizando 0101.0100 0110.0101 e 0011.1000, que são "Te8". Entao: $VGU4_{Base64} \equiv Te8_{ascii}$.

Se o último caracter é "=", então deve-se olhar o penúltimo. Se o penúltimo é diferente de "=", então tem-se aqui 2 caracteres ASCII, e depois de juntar 3 x 6 bits = 18 bits, deve-se desprezar os últimos 2 bits e ficar com 16 bits que perfazem os 2 caracteres ASCII buscados.

Por exemplo, seja o Base64 igual a "QTY=". Como o último é = e o penúltimo não é =, então este grupo codifica 2 caracteres ASCII.

Q, T e Y são os caracteres 17, 20 e 25 do vetor formador. Subtraindo 1 fica: 16, 19 e 24. Convertendo a binário fica: 0 1 0 0 0 0, 0 1 0 0 1 1 e 0 1 1 0 0 0. Juntando e desprezando os dois últimos zeros: 0100.0001, 0011.0110 e 00. Com isso os dois caracteres ASCII são: A6, que é a codificação original. Ou $QTY=_{Base64} \equiv A6_{ascii}$.

Se o último e o penúltimo forem =, então é apenas um o caracter original. Deve-se converter os 2 caracteres iniciais (os que são diferentes de =) formando 12 bits e desprezar os últimos 4 bits.

Por exemplo, para desconverter "Rg==", descobre-se que R e g são os elementos 18 e 33 no vetor formador. Subtraindo fica 17 e 32. Convertendo: 0 1 0 0 1 0 e 1 0 0 0 0 0. Reorganizando 0100.1010.0000. O caracter buscado é 'F'. Entao: $Rg==_{Base64} \equiv F_{ascii}$.

Alguns exemplos

de	conversão	para
tMz	ASCII→b64	dE16
Dx	ASCII→b64	RHg=
6	ASCII→b64	Ng==
ZGxO	b64→ASCII	dlN
cU4=	b64→ASCII	qN
SQ==	b64→ASCII	I

☞ Para você fazer

1. Converta de ASCII para Base64
   a) mvW
   b) GD
   c) U
   d) rYF
2. Converta de Base64 para ASCII
   e) RmVP
   f) Mlk=
   g) bw==
   h) K2RF

a	b	c	d
e	f	g	h

U Positivo - UTFPR - PUCPr –
20/05/2024 - 10:31:53.4
Prof Dr P Kantek
(pkantek@gmail.com)
O esquema de codificação Base64
VIVX037a, V: 1.30 76601 PAULO
JOSE FREIRE CORREA 24CC2202
- 12 SUBSTITUIR e entregar ate
30/maio _____ / _____ / _____

Base64

Base64 é um sistema de numeração posicional que usa 64 como base. Ao produzir um texto que possa ser imprimível em qualquer impressora do mundo ocidental, sem nenhuma chance de erro ou incoerência, apenas podem ser usados os caracteres A..Z, a..z e 0..9. Note que tem-se aqui 26+26+10=62 caracteres. Para completar o conjunto esperado de 64 caracteres, costuma-se usar "+"e "/", nesta ordem. (Note-se que muitos programas que usam este princípio costumam escolher outros 2 caracteres que não sejam "+"e "/", mas nenhum deles usa o rótulo de Base64. São exemplos deste caso: a B64 modificado para URL (usa -"e "_") ou B64 para expressões regulares que usa ("!"e -").

Há ainda um caracter importante, o "="(em Base64) que serve para sinalizar eventuais necessidades de preechimento de *stuffing*.

A idéia desta codificação é poder mandar qualquer conteúdo binário como parte de uma mensagem de e-mail, sem correr o risco de algum conteúdo parcial acabe gerando caracteres de controle do protocolo e ponha a perder a comunicação.

A conversão se dá trocando 3 bytes (8 bits × 3 = 24 bits) em 4 caracteres Base64, cada um deles representado por 6 bits. (então 6 bits × 4 = 24 bits).

Na codificação de 8 bits/byte, usa-se um código qualquer. Atualmente a grande maioria de ambientes usa o padrão ASCII, cuja pequena parte pode ser vista aqui.

uma parte da Tabela ASCII

Binário	Dec	Hex	Car.
0010.1000	40	28	(
0010.1001	41	29	)
0010.1010	42	2A	*
0010.1011	43	2B	+
0010.1100	44	2C	,
0010.1101	45	2D	-
0010.1110	46	2E	.
0010.1111	47	2F	/
0011.0000	48	30	0
0011.0001	49	31	1
0011.0010	50	32	2
0011.0011	51	33	3
0011.0100	52	34	4
0011.0101	53	35	5
0011.0110	54	36	6
0011.0111	55	37	7
0011.1000	56	38	8
0011.1001	57	39	9
0100.0001	65	41	A
0100.0010	66	42	B
0100.0011	67	43	C
0100.0100	68	44	D
0100.0101	69	45	E
0100.0110	70	46	F
0100.0111	71	47	G
0100.1000	72	48	H
0100.1001	73	49	I
0100.1010	74	4A	J
0100.1011	75	4B	K
0100.1100	76	4C	L
0100.1101	77	4D	M
0100.1110	78	4E	N
0100.1111	79	4F	O
0101.0000	80	50	P
0101.0001	81	51	Q
0101.0010	82	52	R
0101.0011	83	53	S
0101.0100	84	54	T
0101.0101	85	55	U
0101.0110	86	56	V
0101.0111	87	57	W
0101.1000	88	58	X
0101.1001	89	59	Y
0101.1010	90	5A	Z
0110.0001	97	61	a
0110.0010	98	62	b
0110.0011	99	63	c
0110.0100	100	64	d
0110.0101	101	65	e
0110.0110	102	66	f
0110.0111	103	67	g
0110.1000	104	68	h
0110.1001	105	69	i
0110.1010	106	6A	j
0110.1011	107	6B	k
0110.1100	108	6C	l
0110.1101	109	6D	m
0110.1110	110	6E	n
0110.1111	111	6F	o
0111.0000	112	70	p
0111.0001	113	71	q
0111.0010	114	72	r
0111.0011	115	73	s
0111.0100	116	74	t
0111.0101	117	75	u
0111.0110	118	76	v
0111.0111	119	77	w
0111.1000	120	78	x
0111.1001	121	79	y
0111.1010	122	7A	z

Para efeito de conversão entre os 2 códigos (ASCII e Base64) deve-se usar o seguinte vetor de conversão:

```
ABCDEFGHIJKLMN
OPQRSTUVWXYZab
cdefghijklmnop
qrstuvwxyz0123
456789+/
```

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

ASCII → Base64

Deve-se pegar 3 bytes a converter (24 bits) e colocá-los lado a lado. Daí, o numerão binário obtido deve ser quebrado em 4 partes, cada uma com 6 bits. Como se sabe, um número binário de 6 bits varia entre 0..63. Este número obtido é que deve indexar o vetor acima estabelecido, dando como resultado 4 caracteres Base64,

Acompanhe no exemplo: Seja passar o ASCII `Aa`) para Base64. Da tabela ASCII, sabe-se que A=0100.0001, a=0110.0001 e)=0010.1001. Juntando tudo, fica: 0100.0001.0110.0001.0010.1001. Rearrumando, fica: 010000.010110.000100.101001, números binários que valem: 16, 22, 4 e 41, respectivamente. Obtendo os elementos 17, 23, 5 e 42 do vetor de elementos do Base64, fica-se com QWEp. Então: $Aa)_{ascii} \equiv QWEp_{Base64}$.

Um cuidado especial deve ser tomado quando se chega ao final do conjunto de caracteres a converter. No fim, podem sobrar nenhum, 1 ou 2 caracteres. Se sobrar nenhum, o problema está resolvido. Se sobrarem 2 caracteres, estes devem ser colocados lado a lado (16 bits) e o conjunto deve ser completado com 2 zeros à direita, formando os 18 bits esperados. Para sinalizar estes 2 *stuffing bits*, após os 3 caracteres Base64 gerados, agrega-se o sinal "=". Por exemplo, seja converter o string ASCII `A1B2C`. O primeiro conjunto `A1B` é convertido normalmente, gerando `QTFC`. Sobra `2C`. Suas configurações em binário são 2=0011.0010 e C=0100.0011. Juntando tudo, fica 0011.0010.0100.0011 cujo comprimento é 16 bits. Agregando 2 zeros à direita e reordenando fica: 0011.0010.0100.0011.00 ou 001100.100100.001100, que correspondem a 12, 36 e 12 respectivamente. Obtendo os elementos 13, 37 e 13 do vetor formador, obtém-se M, k e M respectivamente. A maneira de informar que foram incluídos 2 zeros ao final é colocar um "="como quarto caractere da conversão. Assim: $2C_{ascii} \equiv MkM=_{Base64}$.

Finalmente, se sobrar um único caractere (8 bits), o conversor precisa incluir 4 zeros à direita (*stuffing bits*). Por exemplo, para converter o ASCII "M"para Base64, faz-se: M=0100.1101, com 8 bits. Acrescentando os 4 zeros a direita, fica: 0100.1101.0000 e quebrando em dois números de 6 bits cada, fica: 010011.010000 que são os números 19 e 16. Obtendo o vigésimo e décimo sétimo elementos do vetor formador do Base64 obtem-se "TQ". Para sinalizar que incluiram-se 4 zeros, o conversor acrescenta "==". Com isso, $M_{ascii} \equiv TQ==_{Base64}$.

Base64 → ASCII

A volta segue o mesmo procedimento, agora realizado em sentido inverso. Inicialmente, deve-se verificar que o comprimento da mensagem codificada em Base64 deve ter um número de caracteres múltiplos de 4. Se não for, há um erro. Tudo começa, verificando se o último caractere do grupo (de 4) em análise é igual a "=". Se não for, este grupo não é terminal e deve sofrer o processamento padrão que é juntar os 24 bits (cada letra dá origem a um número de 6 bits, obtido pela localização de cada letra no vetor gerador do Base64) que devem ser reorganizados a fim de obter os 3 bytes em ASCII. Por exemplo, seja o conjunto Base64 "VGU4". As posições destes caracteres (V, G, U e 4) no vetor formador são: 22, 7, 21 e 57. Subtraindo uma unidade (já que agora a origem dos índices é zero) tem-se 21, 6, 20 e 56. Convertendo tais números a binário tem-se 0 1 0 1 0 1, 0 0 0 1 1 0, 0 1 0 1 0 0 e 1 1 1 0 0 0. Juntando e reorganizando 0101.0100 0110.0101 e 0011.1000, que são "Te8". Entao: $VGU4_{Base64} \equiv Te8_{ascii}$.

Se o último caracter é "=", então deve-se olhar o penúltimo. Se o penúltimo é diferente de "=", então tem-se aqui 2 caracteres ASCII, e depois de juntar 3 x 6 bits = 18 bits, deve-se desprezar os últimos 2 bits e ficar com 16 bits que perfazem os 2 caracteres ASCII buscados.

Por exemplo, seja o Base64 igual a "QTY=". Como o último é = e o penúltimo não é =, então este grupo codifica 2 caracteres ASCII.

Q, T e Y são os caracteres 17, 20 e 25 do vetor formador. Subtraindo 1 fica: 16, 19 e 24. Convertendo a binário fica: 0 1 0 0 0 0, 0 1 0 0 1 1 e 0 1 1 0 0 0. Juntando e desprezando os dois últimos zeros: 0100.0001, 0011.0110 e 00. Com isso os dois caracteres ASCII são: A6, que é a codificação original. Ou $QTY=_{Base64} \equiv A6_{ascii}$.

Se o último e o penúltimo forem =, então é apenas um o caracter original. Deve-se converter os 2 caracteres iniciais (os que são diferentes de =) formando 12 bits e desprezar os últimos 4 bits.

Por exemplo, para desconverter "Rg==", descobre-se que R e g são os elementos 18 e 33 no vetor formador. Subtraindo fica 17 e 32. Convertendo: 0 1 0 0 0 1 0 e 1 0 0 0 0 0. Reorganizando 0100.1010.0000. O caracter buscado é 'F'. Entao: $Rg==_{Base64} \equiv F_{ascii}$.

Alguns exemplos

de	conversão	para
tMz	ASCII→b64	dE16
Dx	ASCII→b64	RHg=
6	ASCII→b64	Ng==
ZGxO	b64→ASCII	dlN
cU4=	b64→ASCII	qN
SQ==	b64→ASCII	I

☞ Para você fazer

1. Converta de ASCII para Base64
   a) 2Ig
   b) j7
   c) D
   d) oLt
2. Converta de Base64 para ASCII
   e) b2lI
   f) c08=
   g) Sw==
   h) QnQ1

a	b	c	d
e	f	g	h

202-76601 - IT/IR

U Positivo - UTFPR - PUCPr –
20/05/2024 - 10:31:53.4
Prof Dr P Kantek
(pkantek@gmail.com)
O esquema de codificação Base64
VIVX037a, V: 1.30 76618 PEDRO
HENRIQUE PIEKARSKI DE OL
24CC2202 - 13 SUBSTITUIR e
entregar ate 30/maio _____ /
/ _____

Base64

Base64 é um sistema de numeração posicional que usa 64 como base. Ao produzir um texto que possa ser imprimível em qualquer impressora do mundo ocidental, sem nenhuma chance de erro ou incoerência, apenas podem ser usados os caracteres A..Z, a..z e 0..9. Note que tem-se aqui 26+26+10=62 caracteres. Para completar o conjunto esperado de 64 caracteres, costuma-se usar "+"e "/", nesta ordem. (Note-se que muitos programas que usam este princípio costumam escolher outros 2 caracteres que não sejam "+"e "/", mas nenhum deles usa o rótulo de Base64. São exemplos deste caso: a B64 modificado para URL (usa -"e "_") ou B64 para expressões regulares que usa ("!"e -").

Há ainda um caracter importante, o "="(em Base64) que serve para sinalizar eventuais necessidades de preechimento de *stuffing*.

A idéia desta codificação é poder mandar qualquer conteúdo binário como parte de uma mensagem de e-mail, sem correr o risco de algum conteúdo parcial acabe gerando caracteres de controle do protocolo e ponha a perder a comunicação.

A conversão se dá trocando 3 bytes (8 bits × 3 = 24 bits) em 4 caracteres Base64, cada um deles representado por 6 bits. (então 6 bits × 4 = 24 bits).

Na codificação de 8 bits/byte, usa-se um código qualquer. Atualmente a grande maioria de ambientes usa o padrão ASCII, cuja pequena parte pode ser vista aqui.

uma parte da Tabela ASCII

Binário	Dec	Hex	Car.
0010.1000	40	28	(
0010.1001	41	29	)
0010.1010	42	2A	*
0010.1011	43	2B	+
0010.1100	44	2C	,
0010.1101	45	2D	-
0010.1110	46	2E	.
0010.1111	47	2F	/
0011.0000	48	30	0
0011.0001	49	31	1
0011.0010	50	32	2
0011.0011	51	33	3
0011.0100	52	34	4
0011.0101	53	35	5
0011.0110	54	36	6
0011.0111	55	37	7
0011.1000	56	38	8
0011.1001	57	39	9

0100.0001	65	41	A
0100.0010	66	42	B
0100.0011	67	43	C
0100.0100	68	44	D
0100.0101	69	45	E
0100.0110	70	46	F
0100.0111	71	47	G
0100.1000	72	48	H
0100.1001	73	49	I
0100.1010	74	4A	J
0100.1011	75	4B	K
0100.1100	76	4C	L
0100.1101	77	4D	M
0100.1110	78	4E	N
0100.1111	79	4F	O
0101.0000	80	50	P
0101.0001	81	51	Q
0101.0010	82	52	R
0101.0011	83	53	S
0101.0100	84	54	T
0101.0101	85	55	U
0101.0110	86	56	V
0101.0111	87	57	W
0101.1000	88	58	X
0101.1001	89	59	Y
0101.1010	90	5A	Z

0110.0001	97	61	a
0110.0010	98	62	b
0110.0011	99	63	c
0110.0100	100	64	d
0110.0101	101	65	e
0110.0110	102	66	f
0110.0111	103	67	g
0110.1000	104	68	h
0110.1001	105	69	i
0110.1010	106	6A	j
0110.1011	107	6B	k
0110.1100	108	6C	l
0110.1101	109	6D	m
0110.1110	110	6E	n
0110.1111	111	6F	o
0111.0000	112	70	p
0111.0001	113	71	q
0111.0010	114	72	r
0111.0011	115	73	s
0111.0100	116	74	t
0111.0101	117	75	u
0111.0110	118	76	v
0111.0111	119	77	w
0111.1000	120	78	x
0111.1001	121	79	y
0111.1010	122	7A	z

Para efeito de conversão entre os 2 códigos (ASCII e Base64) deve-se usar o seguinte vetor de conversão:

```
ABCDEFGHIJKLMN
OPQRSTUVWXYZab
cdefghijklmnop
qrstuvwxyz0123
456789+/
```

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

ASCII → Base64

Deve-se pegar 3 bytes a converter (24 bits) e colocá-los lado a lado. Daí, o numerão binário obtido deve ser quebrado em 4 partes, cada uma com 6 bits. Como se sabe, um número binário de 6 bits varia entre 0..63. Este número obtido é que deve indexar o vetor acima estabelecido, dando como resultado 4 caracteres Base64,

Acompanhe no exemplo: Seja passar o ASCII `Aa`) para Base64. Da tabela ASCII, sabe-se que A=0100.0001, a=0110.0001 e)=0010.1001. Juntando tudo, fica: 0100.0001.0110.0001.0010.1001. Rearrumando, fica: 010000.010110.000100.101001, números binários que valem: 16, 22, 4 e 41, respectivamente. Obtendo os elementos 17, 23, 5 e 42 do vetor de elementos do Base64, fica-se com QWEp. Então: $Aa)_{ascii} \equiv QWEp_{Base64}$.

Um cuidado especial deve ser tomado quando se chega ao final do conjunto de caracteres a converter. No fim, podem sobrar nenhum, 1 ou 2 caracteres. Se sobrar nenhum, o problema está resolvido. Se sobrarem 2 caracteres, estes devem ser colocados lado a lado (16 bits) e o conjunto deve ser completado com 2 zeros à direita, formando os 18 bits esperados. Para sinalizar estes 2 *stuffing bits*, após os 3 caracteres Base64 gerados, agrega-se o sinal "=". Por exemplo, seja converter o string ASCII `A1B2C`. O primeiro conjunto `A1B` é convertido normalmente, gerando `QTFC`. Sobra `2C`. Suas configurações em binário são 2=0011.0010 e C=0100.0011. Juntando tudo, fica 0011.0010.0100.0011 cujo comprimento é 16 bits. Agregando 2 zeros à direita e reordenando fica: 0011.0010.0100.0011.00 ou 001100.100100.001100, que correspondem a 12, 36 e 12 respectivamente. Obtendo os elementos 13, 37 e 13 do vetor formador, obtém-se M, k e M respectivamente. A maneira de informar que foram incluídos 2 zeros ao final é colocar um "="como quarto caractere da conversão. Assim: $2C_{ascii} \equiv MkM=_{Base64}$.

Finalmente, se sobrar um único caractere (8 bits), o conversor precisa incluir 4 zeros à direita (*stuffing bits*). Por exemplo, para converter o ASCII "M"para Base64, faz-se: M=0100.1101, com 8 bits. Acrescentando os 4 zeros a direita, fica: 0100.1101.0000 e quebrando em dois números de 6 bits cada, fica: 010011.010000 que são os números 19 e 16. Obtendo o vigésimo e décimo sétimo elementos do vetor formador do Base64 obtem-se "TQ". Para sinalizar que incluiram-se 4 zeros, o conversor acrescenta "==". Com isso, $M_{ascii} \equiv TQ==_{Base64}$.

Base64 → ASCII

A volta segue o mesmo procedimento, agora realizado em sentido inverso. Inicialmente, deve-se verificar que o comprimento da mensagem codificada em Base64 deve ter um número de caracteres múltiplos de 4. Se não for, há um erro. Tudo começa, verificando se o último caractere do grupo (de 4) em análise é igual a "=". Se não for, este grupo não é terminal e deve sofrer o processamento padrão que é juntar os 24 bits (cada letra dá origem a um número de 6 bits, obtido pela localização de cada letra no vetor gerador do Base64) que devem ser reorganizados a fim de obter os 3 bytes em ASCII. Por exemplo, seja o conjunto Base64 "VGU4". As posições destes caracteres (V, G, U e 4) no vetor formador são: 22, 7, 21 e 57. Subtraindo uma unidade (já que agora a origem dos índices é zero) tem-se 21, 6, 20 e 56. Convertendo tais números a binário tem-se 0 1 0 1 0 1, 0 0 0 1 1 0, 0 1 0 1 0 0 e 1 1 1 0 0 0. Juntando e reorganizando 0101.0100 0110.0101 e 0011.1000, que são "Te8". Entao: $VGU4_{Base64} \equiv Te8_{ascii}$.

Se o último caracter é "=", então deve-se olhar o penúltimo. Se o penúltimo é diferente de "=", então tem-se aqui 2 caracteres ASCII, e depois de juntar 3 x 6 bits = 18 bits, deve-se desprezar os últimos 2 bits e ficar com 16 bits que perfazem os 2 caracteres ASCII buscados.

Por exemplo, seja o Base64 igual a "QTY=". Como o último é = e o penúltimo não é =, então este grupo codifica 2 caracteres ASCII.

Q, T e Y são os caracteres 17, 20 e 25 do vetor formador. Subtraindo 1 fica: 16, 19 e 24. Convertendo a binário fica: 0 1 0 0 0 0, 0 1 0 0 1 1 e 0 1 1 0 0 0. Juntando e desprezando os dois últimos zeros: 0100.0001, 0011.0110 e 00. Com isso os dois caracteres ASCII são: A6, que é a codificação original. Ou $QTY=_{Base64} \equiv A6_{ascii}$.

Se o último e o penúltimo forem =, então é apenas um o caracter original. Deve-se converter os 2 caracteres iniciais (os que são diferentes de =) formando 12 bits e desprezar os últimos 4 bits.

Por exemplo, para desconverter "Rg==", descobre-se que R e g são os elementos 18 e 33 no vetor formador. Subtraindo fica 17 e 32. Convertendo: 0 1 0 0 1 0 e 1 0 0 0 0 0. Reorganizando 0100.1010.0000. O caracter buscado é 'F'. Entao: $Rg==_{Base64} \equiv F_{ascii}$.

Alguns exemplos

de	conversão	para
tMz	ASCII→b64	dE16
Dx	ASCII→b64	RHg=
6	ASCII→b64	Ng==
ZGxO	b64→ASCII	dlN
cU4=	b64→ASCII	qN
SQ==	b64→ASCII	I

☞ Para você fazer

1. Converta de ASCII para Base64
   a) 3It
   b) uf
   c) P
   d) Hvt
2. Converta de Base64 para ASCII
   e) QW9C
   f) OFo=
   g) Zw==
   h) ZWJN

a	b	c	d
e	f	g	h

U Positivo - UTFPR - PUCPr –
20/05/2024 - 10:31:53.4
Prof Dr P Kantek
(pkantek@gmail.com)
O esquema de codificação Base64
VIVX037a, V: 1.30 76625 RAUL
KNAPKI CUNHA 24CC2202 - 14
SUBSTITUIR e entregar ate
30/maio _____ / _____ / _____

Base64

Base64 é um sistema de numeração posicional que usa 64 como base. Ao produzir um texto que possa ser imprimível em qualquer impressora do mundo ocidental, sem nenhuma chance de erro ou incoerência, apenas podem ser usados os caracteres A..Z, a..z e 0..9. Note que tem-se aqui 26+26+10=62 caracteres. Para completar o conjunto esperado de 64 caracteres, costuma-se usar "+"e "/", nesta ordem. (Note-se que muitos programas que usam este princípio costumam escolher outros 2 caracteres que não sejam "+"e "/", mas nenhum deles usa o rótulo de Base64. São exemplos deste caso: a B64 modificado para URL (usa -"e "_") ou B64 para expressões regulares que usa ("!"e -").

Há ainda um caracter importante, o "="(em Base64) que serve para sinalizar eventuais necessidades de preechimento de *stuffing*.

A idéia desta codificação é poder mandar qualquer conteúdo binário como parte de uma mensagem de e-mail, sem correr o risco de algum conteúdo parcial acabe gerando caracteres de controle do protocolo e ponha a perder a comunicação.

A conversão se dá trocando 3 bytes (8 bits × 3 = 24 bits) em 4 caracteres Base64, cada um deles representado por 6 bits. (então 6 bits × 4 = 24 bits).

Na codificação de 8 bits/byte, usa-se um código qualquer. Atualmente a grande maioria de ambientes usa o padrão ASCII, cuja pequena parte pode ser vista aqui.

uma parte da Tabela ASCII

Binário	Dec	Hex	Car.
0010.1000	40	28	(
0010.1001	41	29	)
0010.1010	42	2A	*
0010.1011	43	2B	+
0010.1100	44	2C	,
0010.1101	45	2D	-
0010.1110	46	2E	.
0010.1111	47	2F	/
0011.0000	48	30	0
0011.0001	49	31	1
0011.0010	50	32	2
0011.0011	51	33	3
0011.0100	52	34	4
0011.0101	53	35	5
0011.0110	54	36	6
0011.0111	55	37	7
0011.1000	56	38	8
0011.1001	57	39	9
0100.0001	65	41	A
0100.0010	66	42	B
0100.0011	67	43	C
0100.0100	68	44	D
0100.0101	69	45	E
0100.0110	70	46	F
0100.0111	71	47	G
0100.1000	72	48	H
0100.1001	73	49	I
0100.1010	74	4A	J
0100.1011	75	4B	K
0100.1100	76	4C	L
0100.1101	77	4D	M
0100.1110	78	4E	N
0100.1111	79	4F	O
0101.0000	80	50	P
0101.0001	81	51	Q
0101.0010	82	52	R
0101.0011	83	53	S
0101.0100	84	54	T
0101.0101	85	55	U
0101.0110	86	56	V
0101.0111	87	57	W
0101.1000	88	58	X
0101.1001	89	59	Y
0101.1010	90	5A	Z
0110.0001	97	61	a
0110.0010	98	62	b
0110.0011	99	63	c
0110.0100	100	64	d
0110.0101	101	65	e
0110.0110	102	66	f
0110.0111	103	67	g
0110.1000	104	68	h
0110.1001	105	69	i
0110.1010	106	6A	j
0110.1011	107	6B	k
0110.1100	108	6C	l
0110.1101	109	6D	m
0110.1110	110	6E	n
0110.1111	111	6F	o
0111.0000	112	70	p
0111.0001	113	71	q
0111.0010	114	72	r
0111.0011	115	73	s
0111.0100	116	74	t
0111.0101	117	75	u
0111.0110	118	76	v
0111.0111	119	77	w
0111.1000	120	78	x
0111.1001	121	79	y
0111.1010	122	7A	z

Para efeito de conversão entre os 2 códigos (ASCII e Base64) deve-se usar o seguinte vetor de conversão:

```
ABCDEFGHIJKLMN
OPQRSTUVWXYZab
cdefghijklmnop
qrstuvwxyz0123
456789+/
```

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

ASCII → Base64

Deve-se pegar 3 bytes a converter (24 bits) e colocá-los lado a lado. Daí, o numerão binário obtido deve ser quebrado em 4 partes, cada uma com 6 bits. Como se sabe, um número binário de 6 bits varia entre 0..63. Este número obtido é que deve indexar o vetor acima estabelecido, dando como resultado 4 caracteres Base64,

Acompanhe no exemplo: Seja passar o ASCII `Aa`) para Base64. Da tabela ASCII, sabe-se que A=0100.0001, a=0110.0001 e)=0010.1001. Juntando tudo, fica: 0100.0001.0110.0001.0010.1001. Rearrumando, fica: 010000.010110.000100.101001, números binários que valem: 16, 22, 4 e 41, respectivamente. Obtendo os elementos 17, 23, 5 e 42 do vetor de elementos do Base64, fica-se com QWEp. Então: $Aa)_{ascii} \equiv QWEp_{Base64}$.

Um cuidado especial deve ser tomado quando se chega ao final do conjunto de caracteres a converter. No fim, podem sobrar nenhum, 1 ou 2 caracteres. Se sobrar nenhum, o problema está resolvido. Se sobrarem 2 caracteres, estes devem ser colocados lado a lado (16 bits) e o conjunto deve ser completado com 2 zeros à direita, formando os 18 bits esperados. Para sinalizar estes 2 *stuffing bits*, após os 3 caracteres Base64 gerados, agrega-se o sinal "=". Por exemplo, seja converter o string ASCII `A1B2C`. O primeiro conjunto `A1B` é convertido normalmente, gerando `QTFC`. Sobra `2C`. Suas configurações em binário são 2=0011.0010 e C=0100.0011. Juntando tudo, fica 0011.0010.0100.0011 cujo comprimento é 16 bits. Agregando 2 zeros à direita e reordenando fica: 0011.0010.0100.0011.00 ou 001100.100100.001100, que correspondem a 12, 36 e 12 respectivamente. Obtendo os elementos 13, 37 e 13 do vetor formador, obtém-se M, k e M respectivamente. A maneira de informar que foram incluídos 2 zeros ao final é colocar um "="como quarto caractere da conversão. Assim: $2C_{ascii} \equiv MkM=_{Base64}$.

Finalmente, se sobrar um único caractere (8 bits), o conversor precisa incluir 4 zeros à direita (*stuffing bits*). Por exemplo, para converter o ASCII "M"para Base64, faz-se: M=0100.1101, com 8 bits. Acrescentando os 4 zeros a direita, fica: 0100.1101.0000 e quebrando em dois números de 6 bits cada, fica: 010011.010000 que são os números 19 e 16. Obtendo o vigésimo e décimo sétimo elementos do vetor formador do Base64 obtem-se "TQ". Para sinalizar que incluiram-se 4 zeros, o conversor acrescenta "==". Com isso, $M_{ascii} \equiv TQ==_{Base64}$.

Base64 → ASCII

A volta segue o mesmo procedimento, agora realizado em sentido inverso. Inicialmente, deve-se verificar que o comprimento da mensagem codificada em Base64 deve ter um número de caracteres múltiplos de 4. Se não for, há um erro. Tudo começa, verificando se o último caractere do grupo (de 4) em análise é igual a "=". Se não for, este grupo não é terminal e deve sofrer o processamento padrão que é juntar os 24 bits (cada letra dá origem a um número de 6 bits, obtido pela localização de cada letra no vetor gerador do Base64) que devem ser reorganizados a fim de obter os 3 bytes em ASCII. Por exemplo, seja o conjunto Base64 "VGU4". As posições destes caracteres (V, G, U e 4) no vetor formador são: 22, 7, 21 e 57. Subtraindo uma unidade (já que agora a origem dos índices é zero) tem-se 21, 6, 20 e 56. Convertendo tais números a binário tem-se 0 1 0 1 0 1, 0 0 0 1 1 0, 0 1 0 1 0 0 e 1 1 1 0 0 0. Juntando e reorganizando 0101.0100 0110.0101 e 0011.1000, que são "Te8". Entao: $VGU4_{Base64} \equiv Te8_{ascii}$.

Se o último caracter é "=", então deve-se olhar o penúltimo. Se o penúltimo é diferente de "=", então tem-se aqui 2 caracteres ASCII, e depois de juntar 3 x 6 bits = 18 bits, deve-se desprezar os últimos 2 bits e ficar com 16 bits que perfazem os 2 caracteres ASCII buscados.

Por exemplo, seja o Base64 igual a "QTY=". Como o último é = e o penúltimo não é =, então este grupo codifica 2 caracteres ASCII.

Q, T e Y são os caracteres 17, 20 e 25 do vetor formador. Subtraindo 1 fica: 16, 19 e 24. Convertendo a binário fica: 0 1 0 0 0 0, 0 1 0 0 1 1 e 0 1 1 0 0 0. Juntando e desprezando os dois últimos zeros: 0100.0001, 0011.0110 e 00. Com isso os dois caracteres ASCII são: A6, que é a codificação original. Ou $QTY=_{Base64} \equiv A6_{ascii}$.

Se o último e o penúltimo forem =, então é apenas um o caracter original. Deve-se converter os 2 caracteres iniciais (os que são diferentes de =) formando 12 bits e desprezar os últimos 4 bits.

Por exemplo, para desconverter "Rg==", descobre-se que R e g são os elementos 18 e 33 no vetor formador. Subtraindo fica 17 e 32. Convertendo: 0 1 0 0 1 0 e 1 0 0 0 0 0. Reorganizando 0100.1010.0000. O caracter buscado é 'F'. Entao: $Rg==_{Base64} \equiv F_{ascii}$.

Alguns exemplos

de	conversão	para
tMz	ASCII→b64	dE16
Dx	ASCII→b64	RHg=
6	ASCII→b64	Ng==
ZGxO	b64→ASCII	dlN
cU4=	b64→ASCII	qN
SQ==	b64→ASCII	I

☞ Para você fazer

1. Converta de ASCII para Base64
   a) 4gJ
   b) yR
   c) J
   d) XD+
2. Converta de Base64 para ASCII
   e) Vy9B
   f) ODY=
   g) Mw==
   h) RG9u

a	b	c	d
e	f	g	h

202-76625 - IT/IR

U Positivo - UTFPR - PUCPr –
20/05/2024 - 10:31:53.4
Prof Dr P Kantek
(pkantek@gmail.com)
O esquema de codificação Base64
VIVX037a, V: 1.30 76632 VICTOR BRONDANI DOS SANTOS
24CC2202 - 15 SUBSTITUIR e entregar ate 30/maio _____ / / _____

Base64

Base64 é um sistema de numeração posicional que usa 64 como base. Ao produzir um texto que possa ser imprimível em qualquer impressora do mundo ocidental, sem nenhuma chance de erro ou incoerência, apenas podem ser usados os caracteres A..Z, a..z e 0..9. Note que tem-se aqui 26+26+10=62 caracteres. Para completar o conjunto esperado de 64 caracteres, costuma-se usar "+"e "/", nesta ordem. (Note-se que muitos programas que usam este princípio costumam escolher outros 2 caracteres que não sejam "+"e "/", mas nenhum deles usa o rótulo de Base64. São exemplos deste caso: a B64 modificado para URL (usa -"e "_") ou B64 para expressões regulares que usa ("!"e -").

Há ainda um caracter importante, o "="(em Base64) que serve para sinalizar eventuais necessidades de preechimento de *stuffing*.

A idéia desta codificação é poder mandar qualquer conteúdo binário como parte de uma mensagem de e-mail, sem correr o risco de algum conteúdo parcial acabe gerando caracteres de controle do protocolo e ponha a perder a comunicação.

A conversão se dá trocando 3 bytes (8 bits × 3 = 24 bits) em 4 caracteres Base64, cada um deles representado por 6 bits. (então 6 bits × 4 = 24 bits).

Na codificação de 8 bits/byte, usa-se um código qualquer. Atualmente a grande maioria de ambientes usa o padrão ASCII, cuja pequena parte pode ser vista aqui.

uma parte da Tabela ASCII

Binário	Dec	Hex	Car.
0010.1000	40	28	(
0010.1001	41	29	)
0010.1010	42	2A	*
0010.1011	43	2B	+
0010.1100	44	2C	,
0010.1101	45	2D	-
0010.1110	46	2E	.
0010.1111	47	2F	/
0011.0000	48	30	0
0011.0001	49	31	1
0011.0010	50	32	2
0011.0011	51	33	3
0011.0100	52	34	4
0011.0101	53	35	5
0011.0110	54	36	6
0011.0111	55	37	7
0011.1000	56	38	8
0011.1001	57	39	9
0100.0001	65	41	A
0100.0010	66	42	B
0100.0011	67	43	C
0100.0100	68	44	D
0100.0101	69	45	E
0100.0110	70	46	F
0100.0111	71	47	G
0100.1000	72	48	H
0100.1001	73	49	I
0100.1010	74	4A	J
0100.1011	75	4B	K
0100.1100	76	4C	L
0100.1101	77	4D	M
0100.1110	78	4E	N
0100.1111	79	4F	O
0101.0000	80	50	P
0101.0001	81	51	Q
0101.0010	82	52	R
0101.0011	83	53	S
0101.0100	84	54	T
0101.0101	85	55	U
0101.0110	86	56	V
0101.0111	87	57	W
0101.1000	88	58	X
0101.1001	89	59	Y
0101.1010	90	5A	Z
0110.0001	97	61	a
0110.0010	98	62	b
0110.0011	99	63	c
0110.0100	100	64	d
0110.0101	101	65	e
0110.0110	102	66	f
0110.0111	103	67	g
0110.1000	104	68	h
0110.1001	105	69	i
0110.1010	106	6A	j
0110.1011	107	6B	k
0110.1100	108	6C	l
0110.1101	109	6D	m
0110.1110	110	6E	n
0110.1111	111	6F	o
0111.0000	112	70	p
0111.0001	113	71	q
0111.0010	114	72	r
0111.0011	115	73	s
0111.0100	116	74	t
0111.0101	117	75	u
0111.0110	118	76	v
0111.0111	119	77	w
0111.1000	120	78	x
0111.1001	121	79	y
0111.1010	122	7A	z

Para efeito de conversão entre os 2 códigos (ASCII e Base64) deve-se usar o seguinte vetor de conversão:

```
ABCDEFGHIJKLMN
OPQRSTUVWXYZab
cdefghijklmnop
qrstuvwxyz0123
456789+/
```

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

ASCII → Base64

Deve-se pegar 3 bytes a converter (24 bits) e colocá-los lado a lado. Daí, o numerão binário obtido deve ser quebrado em 4 partes, cada uma com 6 bits. Como se sabe, um número binário de 6 bits varia entre 0..63. Este número obtido é que deve indexar o vetor acima estabelecido, dando como resultado 4 caracteres Base64,

Acompanhe no exemplo: Seja passar o ASCII `Aa`) para Base64. Da tabela ASCII, sabe-se que A=0100.0001, a=0110.0001 e)=0010.1001. Juntando tudo, fica: 0100.0001.0110.0001.0010.1001. Rearrumando, fica: 010000.010110.000100.101001, números binários que valem: 16, 22, 4 e 41, respectivamente. Obtendo os elementos 17, 23, 5 e 42 do vetor de elementos do Base64, fica-se com QWEp. Então: $Aa)_{ascii} \equiv QWEp_{Base64}$.

Um cuidado especial deve ser tomado quando se chega ao final do conjunto de caracteres a converter. No fim, podem sobrar nenhum, 1 ou 2 caracteres. Se sobrar nenhum, o problema está resolvido. Se sobrarem 2 caracteres, estes devem ser colocados lado a lado (16 bits) e o conjunto deve ser completado com 2 zeros à direita, formando os 18 bits esperados. Para sinalizar estes 2 *stuffing bits*, após os 3 caracteres Base64 gerados, agrega-se o sinal "=". Por exemplo, seja converter o string ASCII `A1B2C`. O primeiro conjunto `A1B` é convertido normalmente, gerando `QTFC`. Sobra `2C`. Suas configurações em binário são 2=0011.0010 e C=0100.0011. Juntando tudo, fica 0011.0010.0100.0011 cujo comprimento é 16 bits. Agregando 2 zeros à direita e reordenando fica: 0011.0010.0100.0011.00 ou 001100.100100.001100, que correspondem a 12, 36 e 12 respectivamente. Obtendo os elementos 13, 37 e 13 do vetor formador, obtém-se M, k e M respectivamente. A maneira de informar que foram incluídos 2 zeros ao final é colocar um "="como quarto caractere da conversão. Assim: $2C_{ascii} \equiv MkM=_{Base64}$.

Finalmente, se sobrar um único caractere (8 bits), o conversor precisa incluir 4 zeros à direita (*stuffing bits*). Por exemplo, para converter o ASCII "M"para Base64, faz-se: M=0100.1101, com 8 bits. Acrescentando os 4 zeros a direita, fica: 0100.1101.0000 e quebrando em dois números de 6 bits cada, fica: 010011.010000 que são os números 19 e 16. Obtendo o vigésimo e décimo sétimo elementos do vetor formador do Base64 obtem-se "TQ". Para sinalizar que incluiram-se 4 zeros, o conversor acrescenta "==". Com isso, $M_{ascii} \equiv TQ==_{Base64}$.

Base64 → ASCII

A volta segue o mesmo procedimento, agora realizado em sentido inverso. Inicialmente, deve-se verificar que o comprimento da mensagem codificada em Base64 deve ter um número de caracteres múltiplos de 4. Se não for, há um erro. Tudo começa, verificando se o último caractere do grupo (de 4) em análise é igual a "=". Se não for, este grupo não é terminal e deve sofrer o processamento padrão que é juntar os 24 bits (cada letra dá origem a um número de 6 bits, obtido pela localização de cada letra no vetor gerador do Base64) que devem ser reorganizados a fim de obter os 3 bytes em ASCII. Por exemplo, seja o conjunto Base64 "VGU4". As posições destes caracteres (V, G, U e 4) no vetor formador são: 22, 7, 21 e 57. Subtraindo uma unidade (já que agora a origem dos índices é zero) tem-se 21, 6, 20 e 56. Convertendo tais números a binário tem-se 0 1 0 1 0 1, 0 0 0 1 1 0, 0 1 0 1 0 0 e 1 1 1 0 0 0. Juntando e reorganizando 0101.0100 0110.0101 e 0011.1000, que são "Te8". Entao: $VGU4_{Base64} \equiv Te8_{ascii}$.

Se o último caracter é "=", então deve-se olhar o penúltimo. Se o penúltimo é diferente de "=", então tem-se aqui 2 caracteres ASCII, e depois de juntar 3 x 6 bits = 18 bits, deve-se desprezar os últimos 2 bits e ficar com 16 bits que perfazem os 2 caracteres ASCII buscados.

Por exemplo, seja o Base64 igual a "QTY=". Como o último é = e o penúltimo não é =, então este grupo codifica 2 caracteres ASCII.

Q, T e Y são os caracteres 17, 20 e 25 do vetor formador. Subtraindo 1 fica: 16, 19 e 24. Convertendo a binário fica: 0 1 0 0 0 0, 0 1 0 0 1 1 e 0 1 1 0 0 0. Juntando e desprezando os dois últimos zeros: 0100.0001, 0011.0110 e 00. Com isso os dois caracteres ASCII são: A6, que é a codificação original. Ou QTY=$_{Base64} \equiv A6_{ascii}$.

Se o último e o penúltimo forem =, então é apenas um o caracter original. Deve-se converter os 2 caracteres iniciais (os que são diferentes de =) formando 12 bits e desprezar os últimos 4 bits.

Por exemplo, para desconverter "Rg==", descobre-se que R e g são os elementos 18 e 33 no vetor formador. Subtraindo fica 17 e 32. Convertendo: 0 1 0 0 1 0 e 1 0 0 0 0 0. Reorganizando 0100.1010.0000. O caracter buscado é 'F'. Entao: Rg==$_{Base64} \equiv F_{ascii}$.

Alguns exemplos

de	conversão	para
tMz	ASCII→b64	dE16
Dx	ASCII→b64	RHg=
6	ASCII→b64	Ng==
ZGxO	b64→ASCII	dlN
cU4=	b64→ASCII	qN
SQ==	b64→ASCII	I

☞ Para você fazer

1. Converta de ASCII para Base64
   a) +bk
   b) ue
   c) f
   d) 6ww
2. Converta de Base64 para ASCII
   e) Sjhy
   f) QUM=
   g) Vg==
   h) Y1o4

a	b	c	d
e	f	g	h

U Positivo - UTFPR - PUCPr –
20/05/2024 - 10:31:53.4
Prof Dr P Kantek
(pkantek@gmail.com)
O esquema de codificação Base64
VIVX037a, V: 1.30 76649 VICTOR
HUGO DOS SANTOS DE CAMA
24CC2202 - 16 SUBSTITUIR e
entregar ate 30/maio _____ /
/ _____

Base64

Base64 é um sistema de numeração posicional que usa 64 como base. Ao produzir um texto que possa ser imprimível em qualquer impressora do mundo ocidental, sem nenhuma chance de erro ou incoerência, apenas podem ser usados os caracteres A..Z, a..z e 0..9. Note que tem-se aqui 26+26+10=62 caracteres. Para completar o conjunto esperado de 64 caracteres, costuma-se usar "+"e "/", nesta ordem. (Note-se que muitos programas que usam este princípio costumam escolher outros 2 caracteres que não sejam "+"e "/", mas nenhum deles usa o rótulo de Base64. São exemplos deste caso: a B64 modificado para URL (usa -"e "_") ou B64 para expressões regulares que usa ("!"e -").

Há ainda um caracter importante, o "="(em Base64) que serve para sinalizar eventuais necessidades de preechimento de *stuffing*.

A idéia desta codificação é poder mandar qualquer conteúdo binário como parte de uma mensagem de e-mail, sem correr o risco de algum conteúdo parcial acabe gerando caracteres de controle do protocolo e ponha a perder a comunicação.

A conversão se dá trocando 3 bytes (8 bits × 3 = 24 bits) em 4 caracteres Base64, cada um deles representado por 6 bits. (então 6 bits × 4 = 24 bits).

Na codificação de 8 bits/byte, usa-se um código qualquer. Atualmente a grande maioria de ambientes usa o padrão ASCII, cuja pequena parte pode ser vista aqui.

uma parte da Tabela ASCII

Binário	Dec	Hex	Car.
0010.1000	40	28	(
0010.1001	41	29	)
0010.1010	42	2A	*
0010.1011	43	2B	+
0010.1100	44	2C	,
0010.1101	45	2D	-
0010.1110	46	2E	.
0010.1111	47	2F	/
0011.0000	48	30	0
0011.0001	49	31	1
0011.0010	50	32	2
0011.0011	51	33	3
0011.0100	52	34	4
0011.0101	53	35	5
0011.0110	54	36	6
0011.0111	55	37	7
0011.1000	56	38	8
0011.1001	57	39	9
0100.0001	65	41	A
0100.0010	66	42	B
0100.0011	67	43	C
0100.0100	68	44	D
0100.0101	69	45	E
0100.0110	70	46	F
0100.0111	71	47	G
0100.1000	72	48	H
0100.1001	73	49	I
0100.1010	74	4A	J
0100.1011	75	4B	K
0100.1100	76	4C	L
0100.1101	77	4D	M
0100.1110	78	4E	N
0100.1111	79	4F	O
0101.0000	80	50	P
0101.0001	81	51	Q
0101.0010	82	52	R
0101.0011	83	53	S
0101.0100	84	54	T
0101.0101	85	55	U
0101.0110	86	56	V
0101.0111	87	57	W
0101.1000	88	58	X
0101.1001	89	59	Y
0101.1010	90	5A	Z
0110.0001	97	61	a
0110.0010	98	62	b
0110.0011	99	63	c
0110.0100	100	64	d
0110.0101	101	65	e
0110.0110	102	66	f
0110.0111	103	67	g
0110.1000	104	68	h
0110.1001	105	69	i
0110.1010	106	6A	j
0110.1011	107	6B	k
0110.1100	108	6C	l
0110.1101	109	6D	m
0110.1110	110	6E	n
0110.1111	111	6F	o
0111.0000	112	70	p
0111.0001	113	71	q
0111.0010	114	72	r
0111.0011	115	73	s
0111.0100	116	74	t
0111.0101	117	75	u
0111.0110	118	76	v
0111.0111	119	77	w
0111.1000	120	78	x
0111.1001	121	79	y
0111.1010	122	7A	z

Para efeito de conversão entre os 2 códigos (ASCII e Base64) deve-se usar o seguinte vetor de conversão:

```
ABCDEFGHIJKLMN
OPQRSTUVWXYZab
cdefghijklmnop
qrstuvwxyz0123
456789+/
```

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

ASCII → Base64

Deve-se pegar 3 bytes a converter (24 bits) e colocá-los lado a lado. Daí, o numerão binário obtido deve ser quebrado em 4 partes, cada uma com 6 bits. Como se sabe, um número binário de 6 bits varia entre 0..63. Este número obtido é que deve indexar o vetor acima estabelecido, dando como resultado 4 caracteres Base64,

Acompanhe no exemplo: Seja passar o ASCII `Aa`) para Base64. Da tabela ASCII, sabe-se que A=0100.0001, a=0110.0001 e)=0010.1001. Juntando tudo, fica: 0100.0001.0110.0001.0010.1001. Rearrumando, fica: 010000.010110.000100.101001, números binários que valem: 16, 22, 4 e 41, respectivamente. Obtendo os elementos 17, 23, 5 e 42 do vetor de elementos do Base64, fica-se com QWEp. Então: $Aa)_{ascii} \equiv QWEp_{Base64}$.

Um cuidado especial deve ser tomado quando se chega ao final do conjunto de caracteres a converter. No fim, podem sobrar nenhum, 1 ou 2 caracteres. Se sobrar nenhum, o problema está resolvido. Se sobrarem 2 caracteres, estes devem ser colocados lado a lado (16 bits) e o conjunto deve ser completado com 2 zeros à direita, formando os 18 bits esperados. Para sinalizar estes 2 *stuffing bits*, após os 3 caracteres Base64 gerados, agrega-se o sinal "=". Por exemplo, seja converter o string ASCII `A1B2C`. O primeiro conjunto `A1B` é convertido normalmente, gerando `QTFC`. Sobra `2C`. Suas configurações em binário são 2=0011.0010 e C=0100.0011. Juntando tudo, fica 0011.0010.0100.0011 cujo comprimento é 16 bits. Agregando 2 zeros à direita e reordenando fica: 0011.0010.0100.0011.00 ou 001100.100100.001100, que correspondem a 12, 36 e 12 respectivamente. Obtendo os elementos 13, 37 e 13 do vetor formador, obtém-se M, k e M respectivamente. A maneira de informar que foram incluídos 2 zeros ao final é colocar um "="como quarto caractere da conversão. Assim: $2C_{ascii} \equiv MkM=_{Base64}$.

Finalmente, se sobrar um único caractere (8 bits), o conversor precisa incluir 4 zeros à direita (*stuffing bits*). Por exemplo, para converter o ASCII "M"para Base64, faz-se: M=0100.1101, com 8 bits. Acrescentando os 4 zeros a direita, fica: 0100.1101.0000 e quebrando em dois números de 6 bits cada, fica: 010011.010000 que são os números 19 e 16. Obtendo o vigésimo e décimo sétimo elementos do vetor formador do Base64 obtem-se "TQ". Para sinalizar que incluiram-se 4 zeros, o conversor acrescenta "==". Com isso, $M_{ascii} \equiv TQ==_{Base64}$.

Base64 → ASCII

A volta segue o mesmo procedimento, agora realizado em sentido inverso. Inicialmente, deve-se verificar que o comprimento da mensagem codificada em Base64 deve ter um número de caracteres múltiplos de 4. Se não for, há um erro. Tudo começa, verificando se o último caractere do grupo (de 4) em análise é igual a "=". Se não for, este grupo não é terminal e deve sofrer o processamento padrão que é juntar os 24 bits (cada letra dá origem a um número de 6 bits, obtido pela localização de cada letra no vetor gerador do Base64) que devem ser reorganizados a fim de obter os 3 bytes em ASCII. Por exemplo, seja o conjunto Base64 "VGU4". As posições destes caracteres (V, G, U e 4) no vetor formador são: 22, 7, 21 e 57. Subtraindo uma unidade (já que agora a origem dos índices é zero) tem-se 21, 6, 20 e 56. Convertendo tais números a binário tem-se 0 1 0 1 0 1, 0 0 0 1 1 0, 0 1 0 1 0 0 e 1 1 1 0 0 0. Juntando e reorganizando 0101.0100 0110.0101 e 0011.1000, que são "Te8". Entao: $VGU4_{Base64} \equiv Te8_{ascii}$.

Se o último caracter é "=", então deve-se olhar o penúltimo. Se o penúltimo é diferente de "=", então tem-se aqui 2 caracteres ASCII, e depois de juntar 3 x 6 bits = 18 bits, deve-se desprezar os últimos 2 bits e ficar com 16 bits que perfazem os 2 caracteres ASCII buscados.

Por exemplo, seja o Base64 igual a "QTY=". Como o último é = e o penúltimo não é =, então este grupo codifica 2 caracteres ASCII.

Q, T e Y são os caracteres 17, 20 e 25 do vetor formador. Subtraindo 1 fica: 16, 19 e 24. Convertendo a binário fica: 0 1 0 0 0 0, 0 1 0 0 1 1 e 0 1 1 0 0 0. Juntando e desprezando os dois últimos zeros: 0100.0001, 0011.0110 e 00. Com isso os dois caracteres ASCII são: A6, que é a codificação original. Ou $QTY=_{Base64} \equiv A6_{ascii}$.

Se o último e o penúltimo forem =, então é apenas um o caracter original. Deve-se converter os 2 caracteres iniciais (os que são diferentes de =) formando 12 bits e desprezar os últimos 4 bits.

Por exemplo, para desconverter "Rg==", descobre-se que R e g são os elementos 18 e 33 no vetor formador. Subtraindo fica 17 e 32. Convertendo: 0 1 0 0 0 1 0 e 1 0 0 0 0 0. Reorganizando 0100.1010.0000. O caracter buscado é 'F'. Entao: $Rg==_{Base64} \equiv F_{ascii}$.

Alguns exemplos

de	conversão	para
tMz	ASCII→b64	dE16
Dx	ASCII→b64	RHg=
6	ASCII→b64	Ng==
ZGxO	b64→ASCII	dlN
cU4=	b64→ASCII	qN
SQ==	b64→ASCII	I

☞ Para você fazer

1. Converta de ASCII para Base64
   a) kxC
   b) Ct
   c) t
   d) 3Eo
2. Converta de Base64 para ASCII
   e) UEZs
   f) TW8=
   g) Wg==
   h) ayta

a	b	c	d
e	f	g	h

U Positivo - UTFPR - PUCPr –
20/05/2024 - 10:31:53.4
Prof Dr P Kantek
(pkantek@gmail.com)
O esquema de codificação Base64
VIVX037a, V: 1.30 76656 VITOR
EDUARDO DE GOES FONTES
24CC2202 - 17 SUBSTITUIR e
entregar ate 30/maio _____ /
/ _____

Base64

Base64 é um sistema de numeração posicional que usa 64 como base. Ao produzir um texto que possa ser imprimível em qualquer impressora do mundo ocidental, sem nenhuma chance de erro ou incoerência, apenas podem ser usados os caracteres A..Z, a..z e 0..9. Note que tem-se aqui 26+26+10=62 caracteres. Para completar o conjunto esperado de 64 caracteres, costuma-se usar "+"e "/", nesta ordem. (Note-se que muitos programas que usam este princípio costumam escolher outros 2 caracteres que não sejam "+"e "/", mas nenhum deles usa o rótulo de Base64. São exemplos deste caso: a B64 modificado para URL (usa -"e "_") ou B64 para expressões regulares que usa ("!"e -").

Há ainda um caracter importante, o "="(em Base64) que serve para sinalizar eventuais necessidades de preechimento de *stuffing*.

A idéia desta codificação é poder mandar qualquer conteúdo binário como parte de uma mensagem de e-mail, sem correr o risco de algum conteúdo parcial acabe gerando caracteres de controle do protocolo e ponha a perder a comunicação.

A conversão se dá trocando 3 bytes (8 bits × 3 = 24 bits) em 4 caracteres Base64, cada um deles representado por 6 bits. (então 6 bits × 4 = 24 bits).

Na codificação de 8 bits/byte, usa-se um código qualquer. Atualmente a grande maioria de ambientes usa o padrão ASCII, cuja pequena parte pode ser vista aqui.

uma parte da Tabela ASCII

Binário	Dec	Hex	Car.
0010.1000	40	28	(
0010.1001	41	29	)
0010.1010	42	2A	*
0010.1011	43	2B	+
0010.1100	44	2C	,
0010.1101	45	2D	-
0010.1110	46	2E	.
0010.1111	47	2F	/
0011.0000	48	30	0
0011.0001	49	31	1
0011.0010	50	32	2
0011.0011	51	33	3
0011.0100	52	34	4
0011.0101	53	35	5
0011.0110	54	36	6
0011.0111	55	37	7
0011.1000	56	38	8
0011.1001	57	39	9
0100.0001	65	41	A
0100.0010	66	42	B
0100.0011	67	43	C
0100.0100	68	44	D
0100.0101	69	45	E
0100.0110	70	46	F
0100.0111	71	47	G
0100.1000	72	48	H
0100.1001	73	49	I
0100.1010	74	4A	J
0100.1011	75	4B	K
0100.1100	76	4C	L
0100.1101	77	4D	M
0100.1110	78	4E	N
0100.1111	79	4F	O
0101.0000	80	50	P
0101.0001	81	51	Q
0101.0010	82	52	R
0101.0011	83	53	S
0101.0100	84	54	T
0101.0101	85	55	U
0101.0110	86	56	V
0101.0111	87	57	W
0101.1000	88	58	X
0101.1001	89	59	Y
0101.1010	90	5A	Z
0110.0001	97	61	a
0110.0010	98	62	b
0110.0011	99	63	c
0110.0100	100	64	d
0110.0101	101	65	e
0110.0110	102	66	f
0110.0111	103	67	g
0110.1000	104	68	h
0110.1001	105	69	i
0110.1010	106	6A	j
0110.1011	107	6B	k
0110.1100	108	6C	l
0110.1101	109	6D	m
0110.1110	110	6E	n
0110.1111	111	6F	o
0111.0000	112	70	p
0111.0001	113	71	q
0111.0010	114	72	r
0111.0011	115	73	s
0111.0100	116	74	t
0111.0101	117	75	u
0111.0110	118	76	v
0111.0111	119	77	w
0111.1000	120	78	x
0111.1001	121	79	y
0111.1010	122	7A	z

Para efeito de conversão entre os 2 códigos (ASCII e Base64) deve-se usar o seguinte vetor de conversão:

```
ABCDEFGHIJKLMN
OPQRSTUVWXYZab
cdefghijklmnop
qrstuvwxyz0123
456789+/
```

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

ASCII → Base64

Deve-se pegar 3 bytes a converter (24 bits) e colocá-los lado a lado. Daí, o numerão binário obtido deve ser quebrado em 4 partes, cada uma com 6 bits. Como se sabe, um número binário de 6 bits varia entre 0..63. Este número obtido é que deve indexar o vetor acima estabelecido, dando como resultado 4 caracteres Base64,

Acompanhe no exemplo: Seja passar o ASCII `Aa`) para Base64. Da tabela ASCII, sabe-se que A=0100.0001, a=0110.0001 e)=0010.1001. Juntando tudo, fica: 0100.0001.0110.0001.0010.1001. Rearrumando, fica: 010000.010110.000100.101001, números binários que valem: 16, 22, 4 e 41, respectivamente. Obtendo os elementos 17, 23, 5 e 42 do vetor de elementos do Base64, fica-se com QWEp. Então: $Aa)_{ascii} \equiv QWEp_{Base64}$.

Um cuidado especial deve ser tomado quando se chega ao final do conjunto de caracteres a converter. No fim, podem sobrar nenhum, 1 ou 2 caracteres. Se sobrar nenhum, o problema está resolvido. Se sobrarem 2 caracteres, estes devem ser colocados lado a lado (16 bits) e o conjunto deve ser completado com 2 zeros à direita, formando os 18 bits esperados. Para sinalizar estes 2 *stuffing bits*, após os 3 caracteres Base64 gerados, agrega-se o sinal "=". Por exemplo, seja converter o string ASCII `A1B2C`. O primeiro conjunto `A1B` é convertido normalmente, gerando `QTFC`. Sobra `2C`. Suas configurações em binário são 2=0011.0010 e C=0100.0011. Juntando tudo, fica 0011.0010.0100.0011 cujo comprimento é 16 bits. Agregando 2 zeros à direita e reordenando fica: 0011.0010.0100.0011.00 ou 001100.100100.001100, que correspondem a 12, 36 e 12 respectivamente. Obtendo os elementos 13, 37 e 13 do vetor formador, obtém-se M, k e M respectivamente. A maneira de informar que foram incluídos 2 zeros ao final é colocar um "="como quarto caractere da conversão. Assim: $2C_{ascii} \equiv MkM=_{Base64}$.

Finalmente, se sobrar um único caractere (8 bits), o conversor precisa incluir 4 zeros à direita (*stuffing bits*). Por exemplo, para converter o ASCII "M"para Base64, faz-se: M=0100.1101, com 8 bits. Acrescentando os 4 zeros a direita, fica: 0100.1101.0000 e quebrando em dois números de 6 bits cada, fica: 010011.010000 que são os números 19 e 16. Obtendo o vigésimo e décimo sétimo elementos do vetor formador do Base64 obtem-se "TQ". Para sinalizar que incluiram-se 4 zeros, o conversor acrescenta "==". Com isso, $M_{ascii} \equiv TQ==_{Base64}$.

Base64 → ASCII

A volta segue o mesmo procedimento, agora realizado em sentido inverso. Inicialmente, deve-se verificar que o comprimento da mensagem codificada em Base64 deve ter um número de caracteres múltiplos de 4. Se não for, há um erro. Tudo começa, verificando se o último caractere do grupo (de 4) em análise é igual a "=". Se não for, este grupo não é terminal e deve sofrer o processamento padrão que é juntar os 24 bits (cada letra dá origem a um número de 6 bits, obtido pela localização de cada letra no vetor gerador do Base64) que devem ser reorganizados a fim de obter os 3 bytes em ASCII. Por exemplo, seja o conjunto Base64 "VGU4". As posições destes caracteres (V, G, U e 4) no vetor formador são: 22, 7, 21 e 57. Subtraindo uma unidade (já que agora a origem dos índices é zero) tem-se 21, 6, 20 e 56. Convertendo tais números a binário tem-se 0 1 0 1 0 1, 0 0 0 1 1 0, 0 1 0 1 0 0 e 1 1 1 0 0 0. Juntando e reorganizando 0101.0100 0110.0101 e 0011.1000, que são "Te8". Entao: $VGU4_{Base64} \equiv Te8_{ascii}$.

Se o último caracter é "=", então deve-se olhar o penúltimo. Se o penúltimo é diferente de "=", então tem-se aqui 2 caracteres ASCII, e depois de juntar 3 x 6 bits = 18 bits, deve-se desprezar os últimos 2 bits e ficar com 16 bits que perfazem os 2 caracteres ASCII buscados.

Por exemplo, seja o Base64 igual a "QTY=". Como o último é = e o penúltimo não é =, então este grupo codifica 2 caracteres ASCII.

Q, T e Y são os caracteres 17, 20 e 25 do vetor formador. Subtraindo 1 fica: 16, 19 e 24. Convertendo a binário fica: 0 1 0 0 0 0, 0 1 0 0 1 1 e 0 1 1 0 0 0. Juntando e desprezando os dois últimos zeros: 0100.0001, 0011.0110 e 00. Com isso os dois caracteres ASCII são: A6, que é a codificação original. Ou $QTY=_{Base64} \equiv A6_{ascii}$.

Se o último e o penúltimo forem =, então é apenas um o caracter original. Deve-se converter os 2 caracteres iniciais (os que são diferentes de =) formando 12 bits e desprezar os últimos 4 bits.

Por exemplo, para desconverter "Rg==", descobre-se que R e g são os elementos 18 e 33 no vetor formador. Subtraindo fica 17 e 32. Convertendo: 0 1 0 0 0 1 0 e 1 0 0 0 0 0. Reorganizando 0100.1010.0000. O caracter buscado é 'F'. Entao: $Rg==_{Base64} \equiv F_{ascii}$.

Alguns exemplos

de	conversão	para
tMz	ASCII→b64	dE16
Dx	ASCII→b64	RHg=
6	ASCII→b64	Ng==
ZGxO	b64→ASCII	dlN
cU4=	b64→ASCII	qN
SQ==	b64→ASCII	I

☞ Para você fazer

1. Converta de ASCII para Base64
   a) ypf
   b) Ow
   c) g
   d) F+x
2. Converta de Base64 para ASCII
   e) YVk5
   f) Y1c=
   g) ZQ==
   h) N3Vz

a	b	c	d
e	f	g	h

202-76656 - IT/IR