Base-64 Encoding

Fall Term 2025/2026

Emmanuel Benoist | BFH-TI

Base-64 Encoding

- **Motivation**
- **Base64**
- **End of file**

Motivation

Motivation

- **How to send binary data in a text?**
  - Text (mail, html, source code) expect ASCII / UTF8 / ISO Latin1 encoding
  - Binary data can contain all bytes between 0x00 and 0xFF.

  $\Rightarrow$ Binary data cannot be nicely inserted into a text file.
- **Practical use-cases for Base64 encoding**
  - Cryptographic keys inside emails
  - Images inside an HTML document
  - Cryptographic hashes inside XML documents
  - Keys in a Java program
  - …

Base64

How does it work?

- **Bytes are encoded in characters**
  - Binary data are given as input;
  - A string (encoded in ASCII) is given as output.
- **Example**
  - A program is already available on Linux:

```
$ cd images
$ base64 < bullet—bfh.jpg
```


/9j/4AAQSkZJRgABAQEASABIAAD//gAcQ3JlYXRlZCB3aXRoIEdJTVAgb24gYSBNYWP/2wBDAAUD
BAQEAwUEBAQFBQUGBwwIBwcHBw8LCwkMEQ8SEhEPERETFhwXExQaFRERGCEYGhodHx8fExciJCIe
JBweHx7/2wBDAQUFBQcGBw4ICA4eFBEUHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4e
Hh4eHh4eHh4eHh4eHh4eHh7/wAARCAASABYDASIAAhEBAxEB/8QAGQABAQADAQAAAAAAAAAAAAAAAA
AAMCBQcI/8QAKhAAAQQBAgQEBwAAAAAAAAAAAgABAwQRBSEGBxIyIzFBYUJRcYGRsbL/xAAYAQAD
AQEAAAAAAAAAAAAAAAAAADBgcCBP/EACkRAAAFAgQEBwAAAAAAAAAAAAAABAgMRBDEFEiEyBkFRcRMz
QmFyssH/2gAMAwEAAhEDEQA/APSHNmxPULSbFaY4ZQKVxMCw7bCocN8we2vrYezWYx/oW/bfhZ84
RIgosRZyJykZmZt37VrOG+A7t3pn1RypwPuoePFJvp8P339lLK57FkcQvJw6T2yXp2JvyLvofQNl
O3Rqw1B1Ol463Ow6bUs17dcbFWaOaluowLLOihpOmUdKqtWowDDHnL43cn+bv5u6KnMG6bafGIiV
GsWn2kKzmTMeS3KRaSGI7UUhxARxiXQTizuOcZw/orIiDT+a78i+qRpzant+mCIi6wIf/9k=

How does it work?

- **Principle**
  - We transfrom groups of 3 bytes into groups of four characters.
- **Algorithm**
  1. We read three bytes (i.e. 24 bits)
  2. We split the 24 bits in 4 groups of 6 bits
  3. We have four numbers between 0 and 63 (numbers on 6 bits)
  4. For each number we look inside a lookup table to find the corresponding character.

Example

- We read the first three bytes of the file `bullet-bfh.jpg`
  oxff oxd8 oxff
- It can also be seen in bits:
  ob11111111 ob11011000 ob11111111
- We group those 24 bits in groups of 6 bits (creating 4 numbers)
  ob111111 ob111101 ob100011 ob111111
  63 61 35 63
- We look into the lookup table and find the corresponding characters
  '/' '9' 'j' '/'

Lookup Table

Base64 Encoding Table

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

End of file

End of file

- **The file terminates with a group of 3 characters**
  Just do as usual (encode in 4 characters).
- **The file terminates with a group of 2 characters**
  - We add two additional 0-bits at the end (to obtain 18 bits giving 3 characters)
  - We add a single '=' character at the end of the output
- **The file terminates with a group of 1 single character**
  - We add four additional 0-bits at the end (to obtain 12 bits giving 2 characters)
  - We add two '=' characters at the end of the output

End of file: examples

- **2 bytes**
  - We read two bytes
    oxd8 oxff
    ob11011000 ob11111111
  - We make groups of 6 bits (and add two **0** at the end)
    ob110110 ob001111 ob1111**00**
    54 15 60
  - We look inside the lookup table and add one '='
    '2' 'P' '8' '='
- **1 byte**
  - We read one byte
    oxd8
    ob11011000
  - We make groups of 6 bits (and add four **0** at the end)
    ob110110 ob000000
    54 0
  - We look inside the lookup table and add two '='
    '2' 'A' '=' '='