

Enkódování

crypto

Challenge status: Available

Challenge flags: 6

232

Při zkoumání světa kódování znaků narazíte na různé formáty a metody. Základním stavebním kamenem je ASCII, který reprezentuje znaky pomocí sedmibitového kódu. UTF-8 se stalo populárnější alternativou, poskytující kompatibilitu s ASCII a podporu pro širokou škálu mezinárodních znaků. Metody jako Base64 a URL kódování ukazují, jak lze data převést do textových formátů, které lze přenášet přes různé komunikační protokoly.

Theory

Flags

Handbook

Solves

Úvod

V této úloze se budeme bavit o enkódování znaků. Jedná se o způsob, kterým můžeme my, jako lidé číst různá data z počítače. Počítač totiž nezná nic jako písmena, zná pouze jedničky a nuly. Druhů enkódování je mnoho. Každý má své výhody a nevýhody. Začneme rovnou tím nejznámějším. Pravděpodobně ho znáte i Vy!

ASCII

Na úplném počátku nebyl na jeden znak potřeba ani celý bajt, pouze 7 bitů. Vzhledem k tomu, že 1111111 je 127, tak bylo možné takto zakódovat celkem 128 znaků. Po rozkliknutí tabulky níže můžete vidět, že jsou znaky označovány, jak decimálním číslem, tak hexadecimálním. Nejčastěji se však setkáte s označováním znaků právě hexadecimálním číslem. Takže znak [A](#) je 0x41 v hexu.

► ASCII tabulka

Později bylo ASCII rozšířeno o dalších 128 znaků, takže každý znak zabíral 8 bitů neboli jeden bajt.

ASCII tabulky si můžete zobrazit pomocí manuálu



`man ascii`

UTF-8

Jedná se o nejpoužívanější a nejrozšířenější enkódování znaků dodnes. UTF-8 je enkódování s proměnlivou délkou, takže jeden znak může být zakódován pomocí jednoho bajtu, ale také dvou, tří a nejvýše čtyř.

Jeho prvních 128 znaků je identických s ASCII, takže je vždy zaručena zpětná kompatibilita.

Pokud ale UTF-8 může používat až 4 bajty pro reprezentaci jednoho znaku, tak jak odlišíme čtyři unikátní jednobajtové znaky a jeden čtyřbajtový znak? Pojdeme se podívat na tabulku níže, která je výstřížkem z manuálu

Encoding

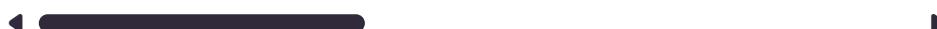
The following byte sequences are used to represent characters:

(hex) 0x00000000 - 0x0000007F:
(bin) 0xxxxxxx

(hex) 0x00000080 - 0x000007FF:
(bin) 110xxxxx 10xxxxxx

(hex) 0x00000800 - 0x0000FFFF:
(bin) 1110xxxx 10xxxxxx 10xxxxxx

(hex) 0x00010000 - 0x001FFFFF:
(bin) 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx



Je vidět že opravdu prvních 128 znaků je stejných, jako původní ASCII, ale zde jednobajtová zábava končí. Sto dvacátý devátý znak, je už totiž reprezentován dvěma bajty. Nahoře je vždy napsáno rozmezí znaků v hexu a pod ním rozsah v binárce. Část prvního bajtu tedy udává kolika bajty je jeden znak reprezentován.

Veškeré informace o UTF-8 naleznete v manuálu [man utf8](#).

Unicode

Je potřeba mít nějakou centrální autoritu, která bude říkat, který znak má jaké číslo. Přesně o to se stará Unicode. Jedná se o standard v enkódování znaků v informačních technologiích. Jeho cílem je

obsáhnout všechny znaky světa. K dnešnímu datu obsahuje přes 140 tisíc znaků.

Tabulka ukazující znak, jeho hexadecimální číslo v Unicode a jeho binární reprezentaci enkódování pomocí UTF-8:

Znak	Unicode	UTF-8 bin
A	U+0041	01000001
a	U+0061	01100001
Ø	U+00D8	11000011 10011000
ȝ	U+0C9A	11100000 10110010 10011010
😊	U+1F601	11110000 10011111 10011000 10000001

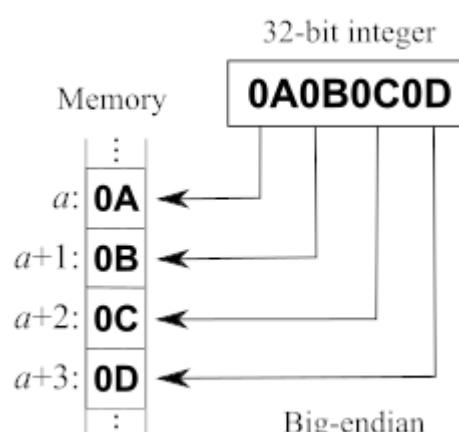
BOM

Na závěr je dobré zmínit ještě takzvaný **Byte-Order Mark**. Nachází se na úplném začátku textového souboru a udává o jaký druh enkódování se jedná. Hlavně udává endianitu dat v souboru.

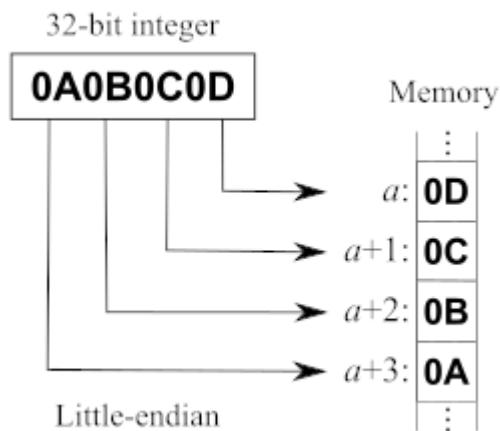
Například UTF-8, má jako Byte-Order Mark **EF BB BF** (hexadecimální sekvence). Tyto znaky nejsou tisknutelné, ale pro software, který vykresluje text velmi důležité. Pokud standardně vypisuje pouze ASCII, ale umí i UTF-8, tak právě díky této páru prvním bajtům ví, že jsou následující data znaky enkódované pomocí UTF-8. BOM pro ostatní enkódování si můžete přečíst [zde](#).

Tedí k té endianitě. Ta je hlavně důležitá pro UTF-16 a 32. Pokud nevíte, o co se jedná, tak endianita udává pořadí bajtů v počítačové paměti. Dělí se na Big-endian a Little-endian.

- Big-endian
 - Ukládá nejvýznamnější bajt na nejmenší paměťové adresu



- Little-endian
 - Ukládá nejméně významný bajt na nejmenší paměťové adresu



UTF-16 a UTF-32

Jak už název napovídá, tak enkódování bude fungovat dost podobně, jako u UTF-8, ale místo několika bajtů, bude znak reprezentován několika šestnácti nebo dvaatřiceti bitovými čísly. Tato enkódování nejsou oproti UTF-8 tak populární, takže pouze zmíním, že existují, ale tím povídání o nich končí.

Base64

Pokud potřebujete přenést data, která nejsou textová, ale putují cestou, kde je povolen pouze přenos textu, zakódujte je přes Base64.

Tím se data přenesou do textového ASCII kódu. V něm je můžete, jak přenášet, tak i ukládat. Kódování funguje tak, že místo toho, aby byl každý znak v bitech reprezentován jedním oktetem (osmi bity – jedním bajtem), tak je reprezentován jen šesti bity a hodnoty těchto šestic udávají index v tabulce Base64.

Fungování Base64 lépe pochopíte na příkladu. Zakódujme si společně slovo 'ahoj'. To převedené do binární soustavy vypadá, jako čtyři znaky po osmi bitech

01100001 01101000 01101111 01101010

Ted' tato čísla rozdělte místo po osmi po šesti. Výsledek bude vypadat takto

011000 010110 100001 101111 011010 100000

Pokud poslední šestice nemá šest číslic, doplňte je nulami. (10 --> 100000)

Hodnoty šestic bitů převeďte na decimální čísla

24 22 33 47 26 32

Tato čísla použijte jako index v následující abecedě, kde A je bráno, jako index 0

ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz01
23456789+/ .

Z tohoto Vám vyjde 24 - Y, 22 - W, 33 - h, 47 - v, 26 - a a 32 - g. Dohromady **YWhvag**.

A na závěr... Base64 zakódovaný text musí obsahovat počet znaků v násobcích čtyř, **YWhvag** obsahuje znaků pouze šest. Jako padding se přidávají '='. V tomto případě přidejte dvě, takže slovo 'ahoj' zakódované pomocí Base64 je

YWhvag==

Není to komplikované, ale je to zdlouhavé. V praxi spíše použijete online nástroj pro zakódování či dekódování. Třeba <https://www.base64encode.org>. Popřípadě pokud preferujete práci v terminálu, tak nástroj **base64**.

Zde jsme si ukázali příklad zakódování textu, to je trochu zbytečné, protože jsme z textu udělali text. Ale jak jsem již zmiňoval na začátku, tak pravá síla Base64 spočívá v tom, že může do ASCII znaků zakódovat úplně libovolná data (obrázky, audiosubory, videa, ale i programy).

URL Encoding

Aby se přes internet mohly přenášet znaky, je nutné je zakódovat. Kódování URL převádí znaky do formátu, který lze odesílat přes internet. Tímto formátem je pouze ASCII znaková sada. V URL kódování jsou zakázány znaky jako mezera, čárka, tečka a mnoho dalších. To se obchází popisem znaku v ASCII tabulce, přesněji jeho hexadecimální hodnotou, kde se tato hodnota uvádí za znak procenta %.

Zde je pár příkladů: %20 - space neboli mezera. Slovní spojení **ahoj ahoj** se zapíše jako **ahoj%20ahoj**

%2B - plus + . 7+3 -> 7%2B3

%26 - ampersand & . pat&mat -> pat%26mat

%22 - double quot mark " . "KBB" -> %22KBB%22

%23 - hashtag # . #KBB -> %23KBB

%28 a %29 - závorky (a) . (ahoj) -> %28ahoj%29

%2E - tečka . konec. -> konec%2E