Unicode ⓙⓔ úplně nejlepší

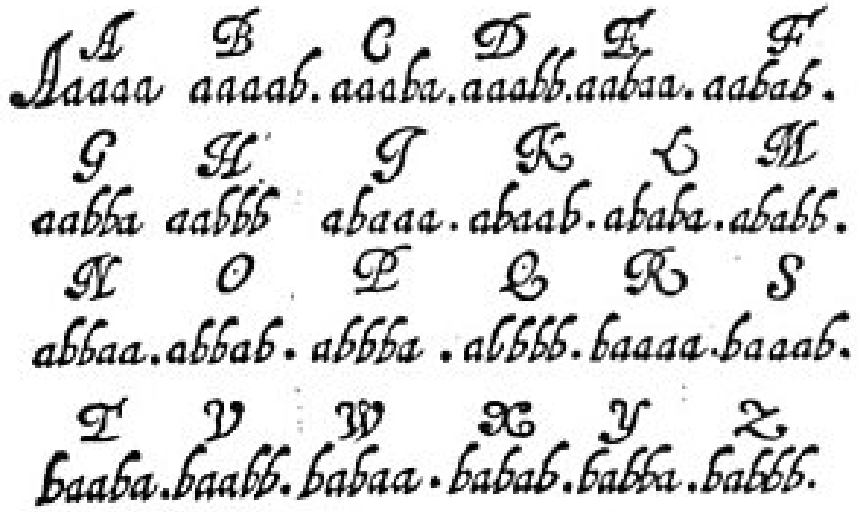
TomÃ¡Å¡ Herceg
CEO @ RIGANTI
Microsoft MVP

David Ge□vindr
MVP: Data Platform | MCSE: Data Platform | MCT
david@wug.cz
@gesvindr

Jak to začalo

- Baconova šifra (1605)
  - A zároveň asi první „character encoding“

- Používaly se dva druhy fontů k zakódování A a B
- Text byl konstruován tak, aby neznalého útočníka zmátl

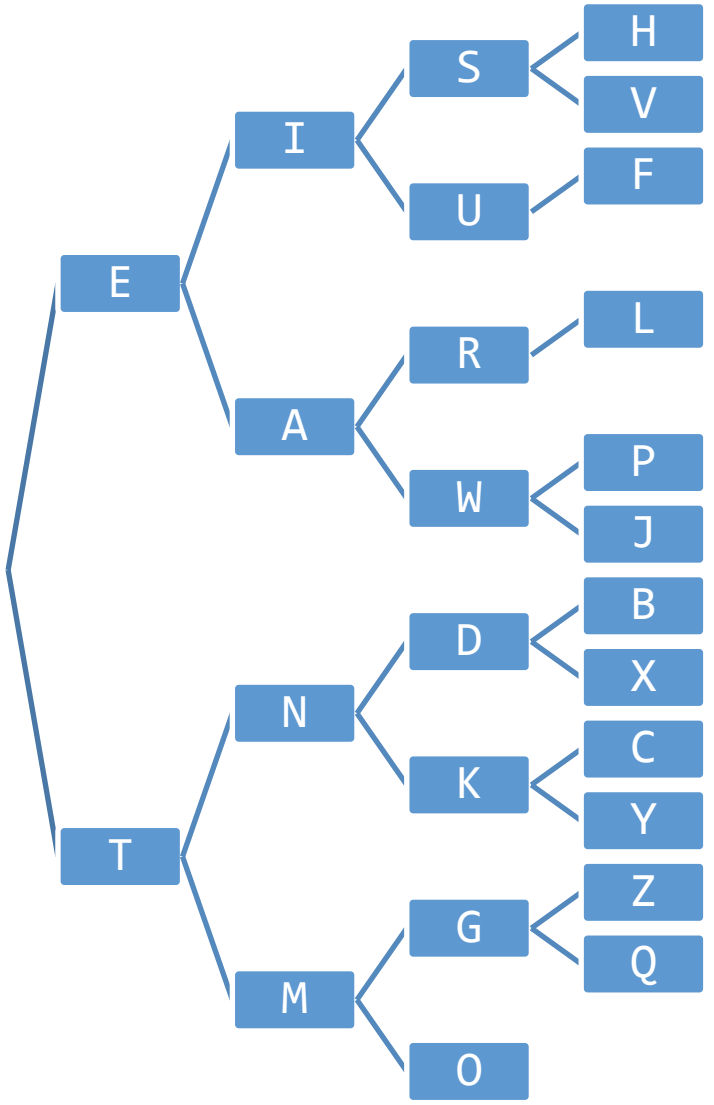


https://en.wikipedia.org/wiki/Bacon%27s_cipher#/media/File:FBacon_alfa1.jpg

- **V** l**e**sn**a**tý**ch** kr**ál**ích **je** loven**o** z**ak**á**z**at v**e**přové divoce.
- babaa baabb aabba aaabb aaaaa babba baaab
- WUG DAYS

Jak to začalo

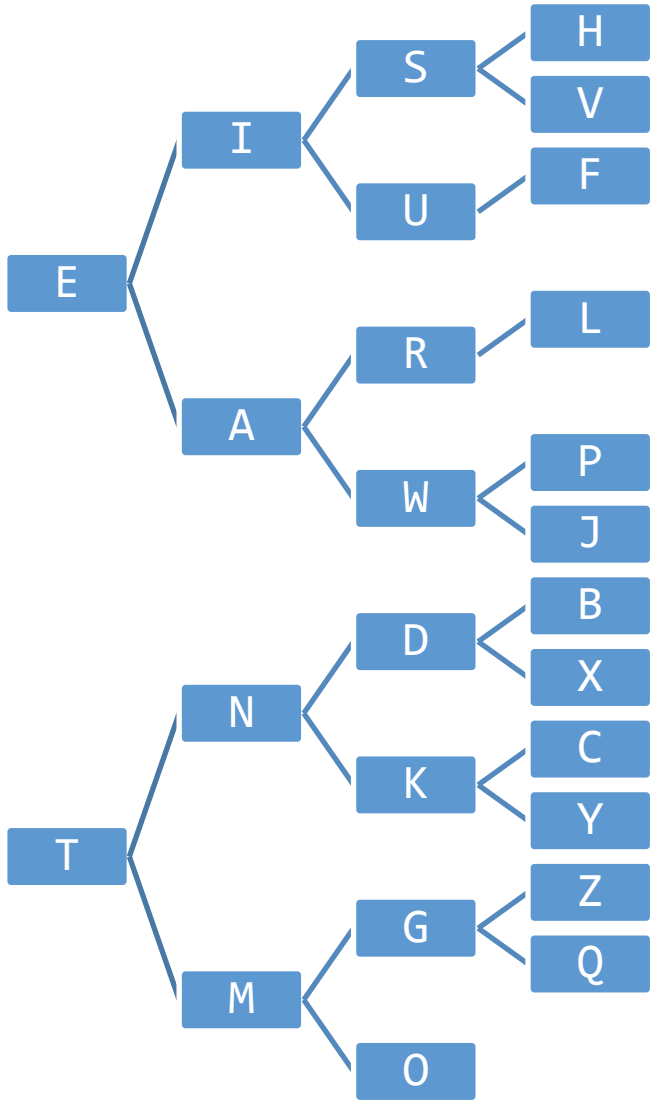
Uhádnete, co představuje tento strom?



Jak to začalo

- Morseova abeceda (1837)

- Postupně měněna a vylepšována
- Dnes používaná mezinárodní verze od roku 1865



ASCII

- 7-bitové kódování (1963)

0 – 31	(00xxxxxx)	řídící znaky
32 – 63	(01xxxxxx)	symboly a číslice
64 – 95	(10xxxxxx)	velká písmena
96 – 127	(11xxxxxx)	malá písmena

- Národní znaky se řeší rozšířením o osmý bit
  - CP437, Kód Kamenických – MS DOS
  - CP852 – konzolové aplikace
  - Windows 1250
  - ISO 8859-*

Unicode

- Vznikalo v letech 1988 – 1991
- Zahrnuje cca 150 000 znaků
  - Dimenzováno pro ~1,1 milionu znaků
- Zpětná kompatibilita s ASCII

- Různé formáty reprezentace
  - UTF-32 vždy 4 bajty / znak
  - UCS-2 vždy 2 bajty / znak – obsolete, nepodporuje všechny znaky
  - UTF-8 1 – 4 bajty / znak
  - UTF-16 2 nebo 4 bajty / znak

UTF-8

- ASCII znaky se kódují stejně
- Ostatní znaky zaberou 2 – 4 bajty
- Vhodné pro texty využívající latinku
- Designováno tak, aby neobsahovalo nechtěné nulové bajty

Rozsah	Byte 1	Byte 2	Byte 3	Byte 4
U+0000 – U+007F (127)	0xxxxxxx			
U+0080 – U+07FF (2047)	110xxxxx	10xxxxxx		
U+0800 – U+FFFF (65535)	1110xxxx	10xxxxxx	10xxxxxx	
U+10000 – U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

UTF-16

- Vhodné pro texty, kde většina znaků není obsažena v ASCII
  - Zejména země Afriky a Asie…
- Rozsah U+D800 – U+DFFF je v Unicode neobsazen a je vyhrazen právě pro „surrogate pairs“

Rozsah	Byte 1 a 2	Byte 3 a 4
U+000000 – U+00FFFF (65535)	xxxxxxxx xxxxxxxx	
U+010000 – U+10FFFF	110110xx xxxxxxxx	110111xx xxxxxxxx

Co je to BOM

- Hlavička na začátku souboru, která určuje použité kódování
- “Zneužití” znaku U+FEFF ZERO WIDTH NO-BREAK SPACE

- EF BB BF UTF-8
- FE FF UTF-16 big endian
- FF FE UTF-16 little endian
- …

Jak je to v .NETu

- `String`
  - Kódován v UTF-16
  - Sekvence `char`ů
  - `Length` může občas mystifikovat
  - Rozsekání na jednotlivé znaky nemusí být vždy validní
- `Char`
  - Vždy 2 bajty
  - Někdy tedy reprezentuje znak, někdy „surrogate“
- `Rune`
  - Reprezentuje 1 znak (pomocí 2 bajtů) nebo sekvenci 2 surrogate znaků
  - Validuje, že jsou hodnoty ve správných rozsazích

DEMO

Babylonský klínopis a .NET

C# a UTF-8

- Používání UTF-8 může znamenat výkonnostní úsporu
  - >90% datových přenosů na Internetu je v UTF-8

- C# 11 přidal UTF-8 string
  - `"Hello world"u8`
- Třída `Utf8String` je slibovaná, ale zatím není
  - Reprezentuje se pomocí `ReadOnlySpan<byte>`

- `System.Text.Json` umí pracovat nativně s UTF-8

Datové typy na uložení textu v SQL Serveru

- char
- nchar
- varchar
- nvarchar
- text ⚠️ - Deprecated, používejte varchar(max)
- ntext ⚠️ - Deprecated, používejte nvarchar(max)

UCS-2 vs. UTF-16 vs. UTF-8 v SQL Serveru

- Datové typy nchar a nvarchar jsou běžně kódovány s pomocí UCS-2
  - Podporuje pouze Basic Multilingual Plane, tedy rozsah 000000–00FFFF (65,536 znaků)
- SQL Server 2012 přidává podporu supplementary character (_SC) znakových sad
  - Zde je již plná podpora Unicode znaků v rozsahu (000000–10FFFF)
- SQL Server 2019 rozšiřuje datové typy char a varchar o podporu UTF-8 formou nových znakových sad _UTF8
  - Kolik znaků uložíte do sloupce s datovým typem varchar(50) v UTF-8?

DEMO

UTF-8 data z webu přímo do databáze

Ekvivalence a kompatibilita

- Některé sekvence znaků vedou k ekvivalentnímu výsledku
  - ˇ (0x02C7) + c (0x0063) = č (0x010D)
  - Zobrazování, řazení atd. se musí chovat stejně

- Některé sekvence jsou kompatibilní, ale reprezentuje jiný znak
  - ff (0xFB00) vs f (0x0066) + f (0x0066)
  - Je na aplikaci, jak s tím naloží

Normalizace

- Převod různých zápisů do jednotné podoby
  - NFC (kanonická kompozice)
  - NFD (kanonická dekompozice)
  - NFKC (kompatibilní kompozice)
  - NFKD (kompatibilní dekompozice)

- Příliš žluťoučký kůň — Form C
- Pr˘i´lis˘ z˘lut˘ouc˘ky´ ku°n˘ — Form D

Odstranění diakritiky

- Převedení na formu D
- Vynechání znaků, které mají kategorii Non-spacing mark
- Převedení na formu C

- Funguje pro češtinu
- V němčině se ä, ö, ü typicky přepisují jako ae, oe, ue…
- Pozor na norštinu nebo turečtinu

DEMO

Odstranění diakritiky

Jazyky jsou obecně peklo

- Turci mají kromě **I** a **i** ještě **ı** a **İ**

- My zase máme **ch**, které se chová jako spřežka (většinou)
  - Např. slovo **nichodný**
  - Zde by se měl použít Unicode znak **Combining Grapheme Joiner**
    - Navzdory názvů grafémy odděluje a nespojuje
    - Řekne aplikaci, že se v tomto případě nejedná o spřežku

DEMO

Pekelné cultures

Znakové sady v SQL Serveru

- Každá textová hodnota v SQL Serveru má definovanou znakovou sadu, což ovlivňuje:
  - Pořadí znaků v abecedě při řazení
  - Porovnání znaků (rozlišování malých/velkých písmen, diakritiky)
  - Způsob, jak jsou jednotlivé znaky binárně reprezentovány
    - Pro non-unicode texty určuje použitou code-page
    - Pro unicode texty určuje jestli se použije UCS-2, UTF-16 či UTF-8
- Znaková sada je definována na úrovni:
  - Serveru
  - Databáze
  - Sloupce v tabulce či výrazu

Speciality a perličky

- Symboly pro římské číslice
  - Ⅰ, Ⅱ, Ⅲ, Ⅳ, …
- Viditelné symboly pro reprezentaci řídících znaků z ASCII
  - ␀, ␍, ␊
- Boxy a bloky
  - ▙ ▒ ▓ ▓ ╔ ╗ ╠ ╣
- Dingbats
  - Různé tiskové značky pro ořez, oddělování kapitol, iniciály v knihách
- Braillovo písmo

Obskurní jazyky

- Hieroglyfy
- Klínopis
- Mayské číslice
- Historické hudební notace (Antické Řecko, Byzantská říše…)
- Alchymistické značky
- Hrací karty
- Mahjong a domino

Jak fungují emojis

- Unicode definuje blok se základními emotikonami
  - 😀 😄 😂
- Může následovat znak Variation Selector 15 nebo 16
- Dále je možné použít modifikátory
  - Např. barva kůže podle Fitzpatrickovy škály
  - 🫅 (U+1F645) 🫅🏻 (U+1F645 a U+1F3FB)

Když Unicode znaky změní kategorii…

- Identifikátory v C# můžou obsahovat různé Unicode kategorie
- Mimo jiné i Cf – Format control characters
  - Zero-width space
  - Mongolian Vowel Separator
  - Musical Symbol Begin Beam / End Beam
  - Right-to-Left Override
  - Egyptian Hieroglyph Horizontal Joiner
  - …
- Při porovnávání dvou identifikátorů se znaky z Cf ignorují

```
var XX = 15;
Console.WriteLine(X\u200bX);
```

Když Unicode znaky změní kategorii…

- Mongolian Vowel Separator se v historii Unicode přesouval z Cf do Zs (space separator) a později zase zpět
- Různé verze C# kompilátoru se chovají jinak
  - U+0180 se někdy bere jako whitespace (Zs)
  - A někdy jako součást identifikátoru, která se při porovnávání ignoruje…

```
ƀʮǰčʃą ƀʮǰčʃąỏ !+¹Ʊ¹ó
ƀʮǰčʃą5l̲EMΔáỏ !+¹Ż¹ó
Æğʃƀğėáúəǰčʮáİčʃá6ƀʮǰčʃąỏ7ó
```

https://codeblog.jonskeet.uk/2014/12/01/when-is-an-identifier-not-an-identifier-attack-of-the-mongolian-vowel-separator/

Vysvětlení názvu přednášky

Enclosed Alphanumerics

Combining Marks
- Combining X Above
- Combining X Below

Combining Marks
- Combining Short Solidus Overlay
- Combining Right Half Ring Above
- Combining Candrabindu
- Combining Vertical Line Below