



Unicode

Unicode Is...

The “phonebook” model for character encoding.

A symbol/character can be mapped to via a registered number.

Thomas Stover	512-867-5309
integral (∫)	u+222b

Quick History of Character Sets

- Morse Code 1836
- Baudot 1870 - 5 bit word telegraph line system
- Murray Code 1901 – added CR/LF
- ITA2/USTTY circa 1930 – still used in TDDs
- EBCDIC 1963 - 8bit
- ASCII 1963 – 7bit
- Many Various ISO-*
- Universal Character Set (Unicode) 1990 – still being worked on; encoding agnostic



Pre-Unicode – Extended ASCII

Proprietary way of encoding. Not standard. Difficult to adapt to other languages, symbols, and environments.

248 (Decimal)

F8 (HEX)

11111000



Degree Symbol

U+00B0

English: I can eat glass, and it doesn't hurt me.

French: Je peux manger du verre, ça ne me fait pas mal.

Korean: 나는 유리를 먹을 수 있어요 . 그래도 아프지 않아요

Classical Greek: ὕαλον φαγεῖν δύναμαι· τοῦτο οὐ με βλάπτει.

Spanish: Puedo comer vidrio, no me hace daño.

Hawaiian: Hiki ia‘u ke ‘ai i ke aniani; ‘a‘ole nō lā au e ‘eha.

Ukrainian: Я можу їсти шкло, й воно мені не пошкодить.

Turkish (Ottoman): جام ييه بلورم گا ضررى طوقونمز

Sources:

<http://www.columbia.edu/kermi/utf8.html>

http://www.alanwood.net/unicode/miscellaneous_symbols.html

Not Just International Text

- [illegible]

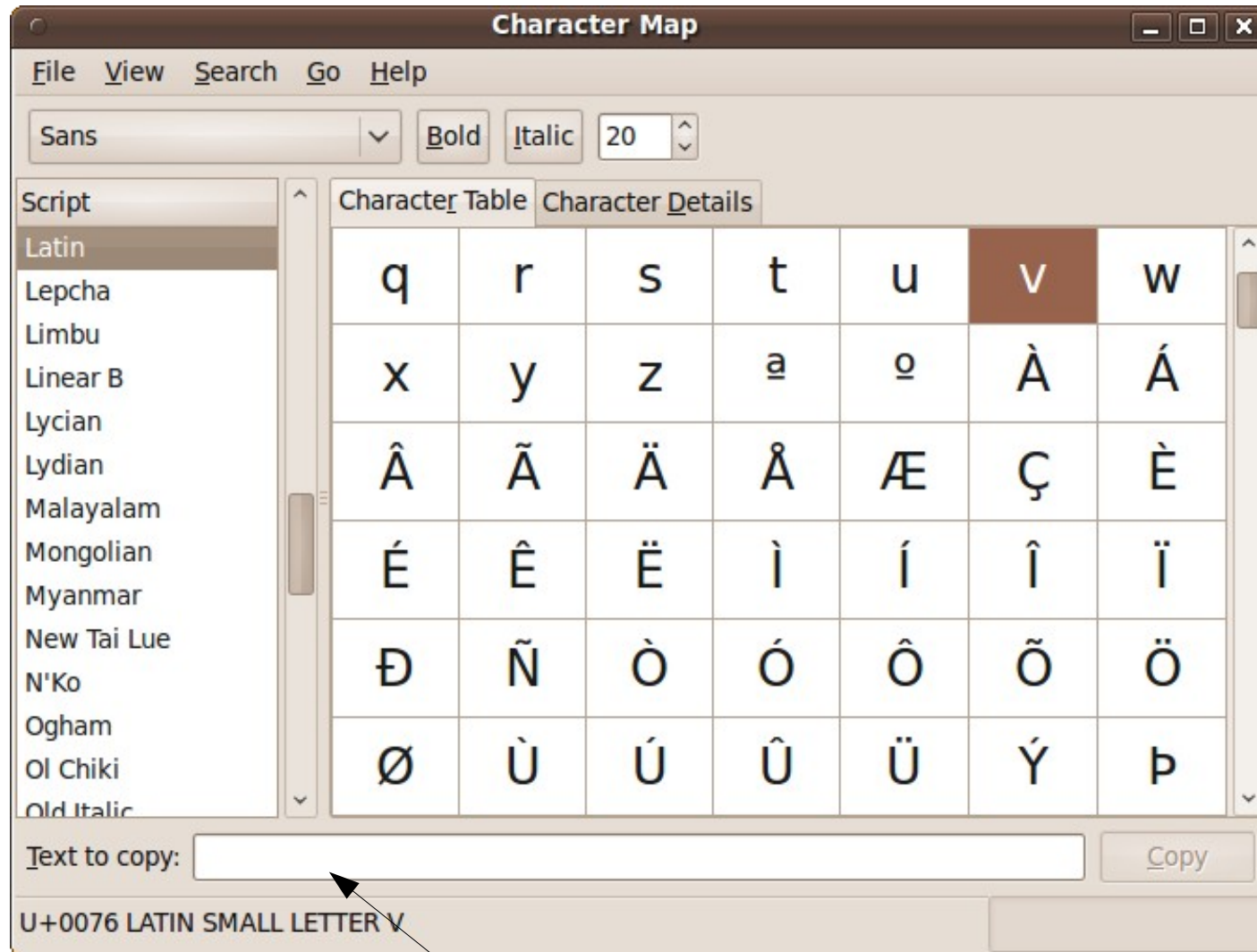
OSS Project Leverage

- Wikipedia
 - ~ 3.2 M **English** Entries
 - ~ 1 M **French** Entries
 - ~ 600k Polish Entries
- OLPC
 - Implementing UI and data input for third world nations in tens of dozens of foreign languages.
 - Just in N.A there are hundreds of official recognized spoken languages.

Where do these characters come from?

- Documents / Files that already have them
- Cut & Paste
- Special Helper Programs (character maps)
- Keyboards
- Source Code

Gnome Character Map – cut & paste any character



Manually enter any hex code

Keyboard Input Options

- Key Remapping
 - Sometime relabeling keys
 - X11 allows for multiple keyboards each with different mappings
- International Keyboards
 - Again multiple simultaneous keyboards possible
- Special Key Combinations (most common)

Unicode Input Key Combinations

- All Gtk+ applications: ctrl+shift+u, then hexadecimal code (full 32bit support), then enter
- Windows: alt+decimal code on the keypad, then release alt
 - Works most of the time
 - Must start with 0 (otherwise it's the old DOS behavior)
- Many application specific exist
- Qt (KDE) ?

One Type of Japanese Keyboard



One type of Hebrew Keyboard



One type of Russian Keyboard



Quick History of Unicode Encodings

- UCS-2 (Windows NT Era) limited to the first 2^{16} characters of Unicode; obsolete but still used
- UTF-16 – replaced UCS-2; almost identical except characters D800 – DFFF can be “swapped out” for different ranges of the remaining 4 billion possible characters
- Both of these are host ordered 2 byte integers resulting in the LE/BE variants
- Used internally by Windows, Qt, Java, .Net, Python, & more

Quick History of Unicode Encodings

- UTF-32 / UCS-4 (32bit variant of the last page)
--Most of these can safely be ignored---
- UTF-7 (“7 bit clean” version of UTF-8 [more on that next])
- UTF-1 (pre UTF-8 variable length scheme with compatibility with nothing)
- SCSU (odd ball compression scheme)
- BOCU-1 – merger of UTF-8, SCSU, & MIME (gross)

...and then there is...

UTF-8, *helping you to forget!*

- Greatest thing ever.
- All characters are supported
- Byte Order (endianess) independent
- Variable Length (bad, but technically all Unicode can be also more on that latter)
- First 128 characters are remapped to ASCII
- Compatible with the *nix world
- Compatible with the *nix style C programming – stdio, POSIX system calls, etc
- Allows for a great deal of Unicode support with out doing anything
- Conventionally, the only valid way to put Unicode directly in source of any kind. (without escape codes)
- “cruise control” for UCS



UTF-EBCDIC



- Similar Concept to UTF-8
- Store Unicode on EBCDIC systems

Unicode in DBMS

Collation and Encoding

MySQL localhost

 **Create new database** 

 MySQL connection collation: 

Always ensure your client is set to the same

```
markgreene@Mule: ~  
File Edit View Terminal Help  
markgreene@Mule:~$ mysql -u root -p --default-character-set=ascii  
Enter password:
```

Unicode in DBMS

```
mysql> Select * From UTF8_ExampleDB.ShopInventory;
```

tool	desc	onLoan
Hammer?	Used to fix "PC Load Letter" errors.	0

```
mysql> SET NAMES UTF8;
```

```
mysql> Select * From UTF8_ExampleDB.ShopInventory;
```

tool	desc	onLoan
Hammer®	Used to fix "PC Load Letter" errors.	0

Unicode on Mobile Devices

- Blackberry
- Android
- iPhone

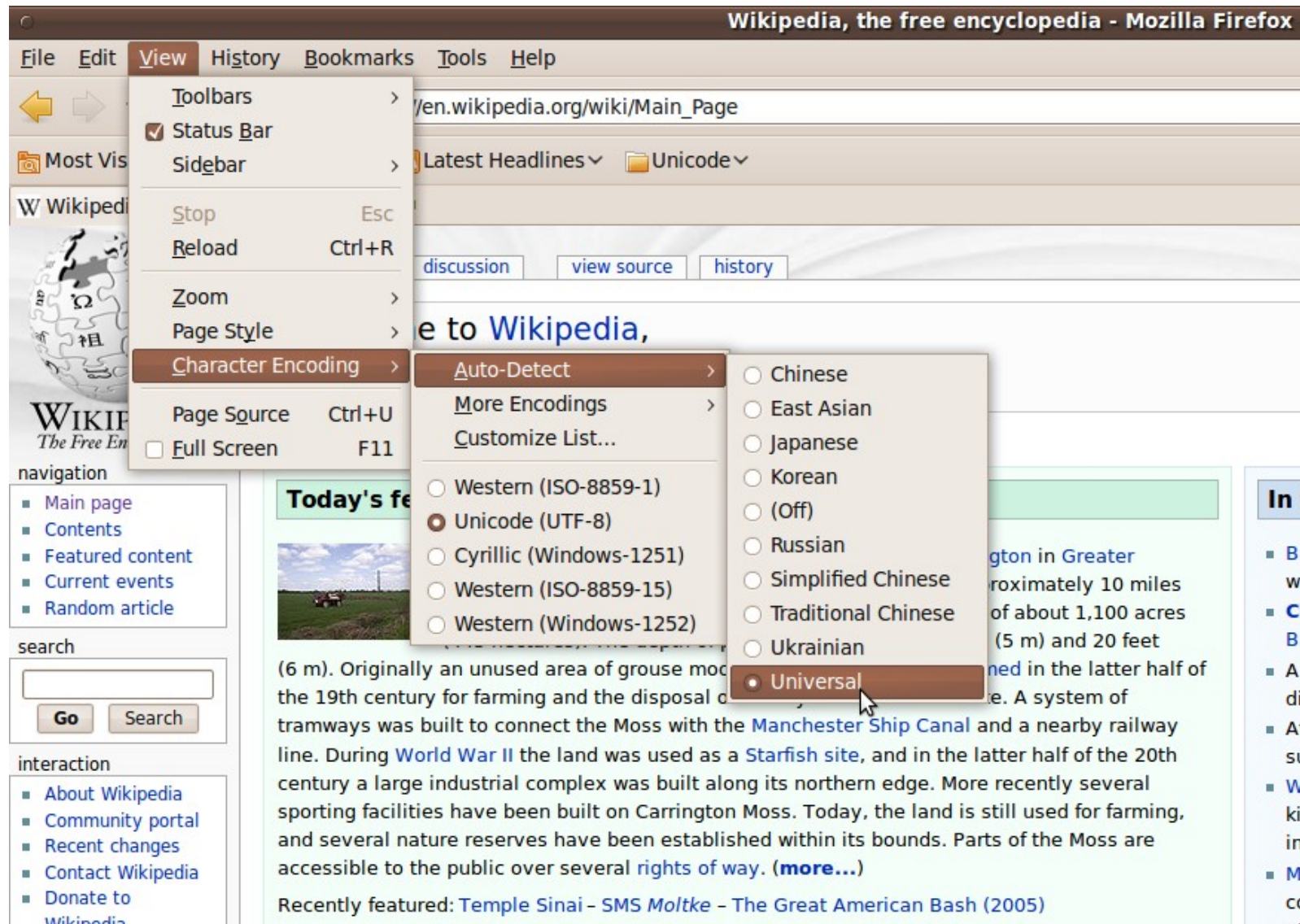
Unicode in Web Browsers

- UTF-8 Works Great
- You MUST specify the “charset”

```
<title>Wikipedia, the free encyclopedia</title>  
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />  
<meta http-equiv="Content-Style-Type" content="text/css" />
```

- Most JavaScript Functions will work with UTF-8
- ~ 50% of websites are using some encoding other than UTF, like Latin1

Unicode in Web Browsers





u222B



$\int 2$



$\int 2$



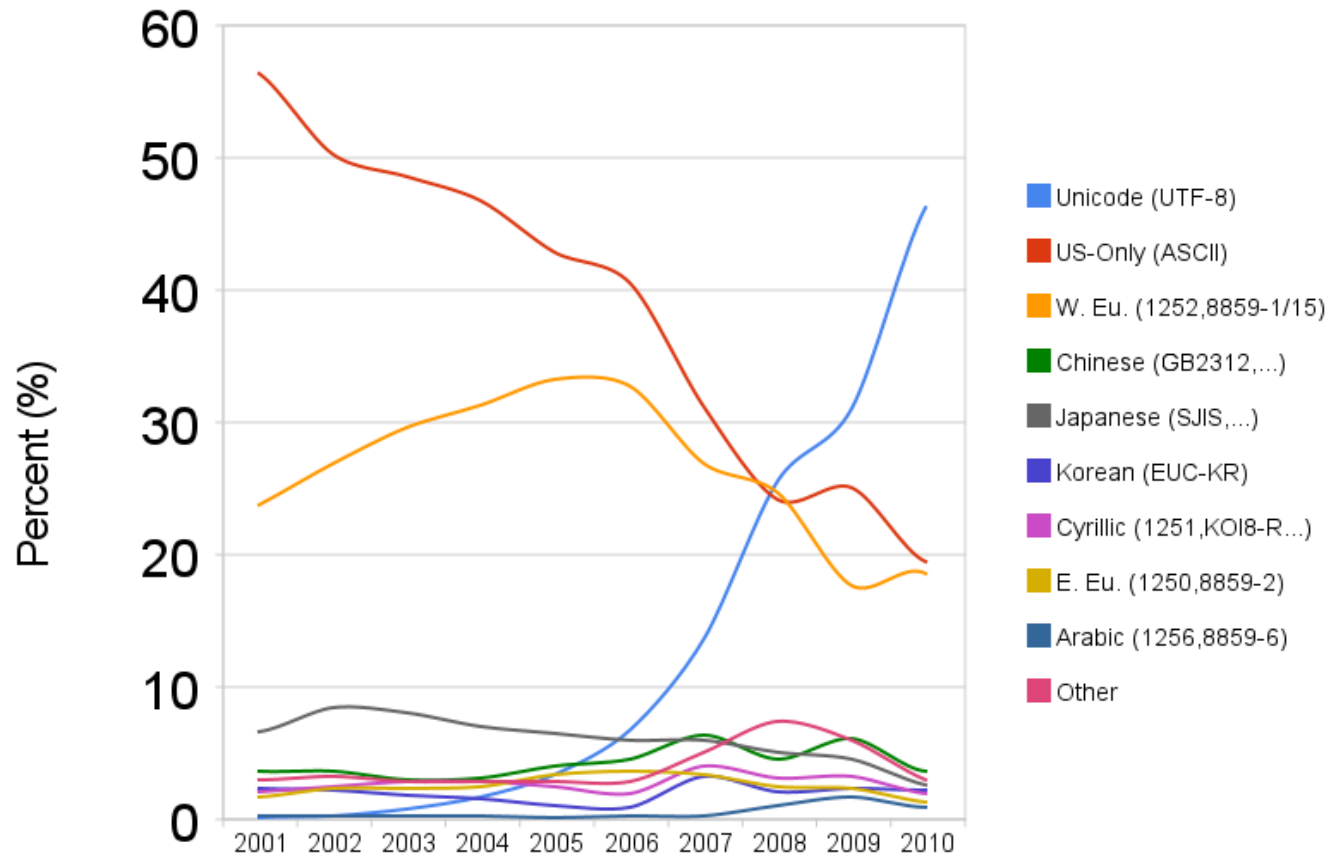
Indefinite integral:

Show steps

$$\int 2 \, dx = 2x + \text{constant}$$

Sites Using Unicode

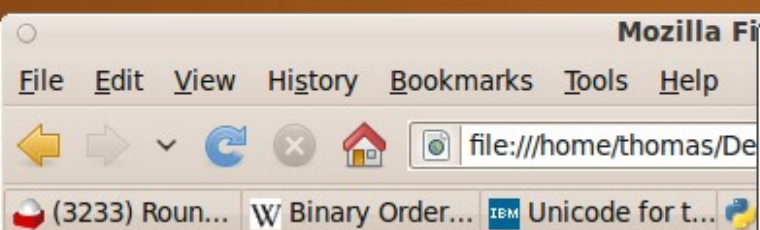
Growth of Unicode on the Web



UTF is only recently being implemented on a wide scale in website design.

Source:

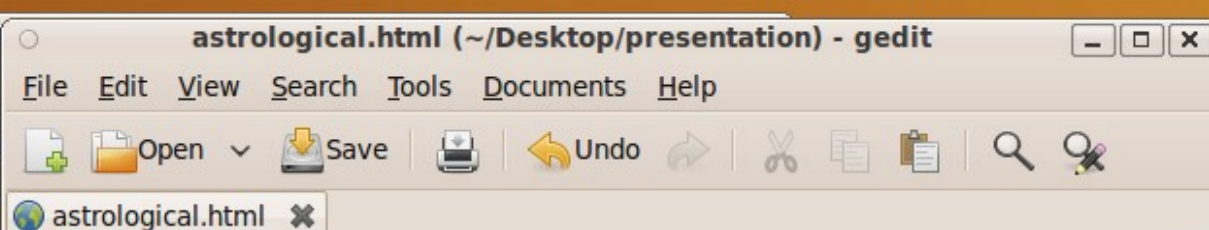
<http://googleblog.blogspot.com/2010/01/unicode-nearing-50-of-web.html>



Some Asctrologic Unicode Characte

â~½	263D	FIRST QUARTER MOON
â~¾	263E	LAST QUARTER MOON
â~¿	263F	MERCURY
â™€	2640	FEMALE SIGN
â™¸	2641	EARTH
â™‚	2642	MALE SIGN
â™Œ	2643	JUPITER
â™¸	2644	SATURN
â™¸...	2645	URANUS
â™¸†	2646	NEPTUNE
â™¸‡	2647	PLUTO
â™¸^	2648	ARIES
â™¸‰	2649	TAURUS
â™¸Š	264A	GEMINI
â™¸<	264B	CANCER
â™¸Œ	264C	LEO

Done



```
<body>
<h1>Some Asctrological Symbols as Unicode Characters</h1>
<table>
  <tr>
    <td>½</td><td>263D</td><td>FIRST QUARTER MOON</td>
  </tr>
  <tr>
    <td>¾</td><td>263E</td><td>LAST QUARTER MOON</td>
  </tr>
  <tr>
    <td>¿</td><td>263F</td><td>MERCURY</td>
  </tr>
  <tr>
    <td>€</td><td>2640</td><td>FEMALE SIGN</td>
  </tr>
  <tr>
    <td>⌘</td><td>2641</td><td>EARTH</td>
  </tr>
  <tr>
    <td>♂</td><td>2642</td><td>MALE SIGN</td>
  </tr>
  <tr>
    <td>♃</td><td>2643</td><td>JUPITER</td>
  </tr>
  <tr>
    <td>♄</td><td>2644</td><td>SATURN</td>
  </tr>
  <tr>
    <td>♅</td><td>2645</td><td>URANUS</td>
  </tr>
  <tr>
    <td>♆</td><td>2646</td><td>NEPTUNE</td>
  </tr>
  <tr>
    <td>♇</td><td>2647</td><td>PLUTO</td>
  </tr>
  <tr>
    <td>♈</td><td>2648</td><td>ARIES</td>
  </tr>
```


Some Asctrological Symbols as Unicode Characters

☾ 263D FIRST QUARTER MOON

☾ 263E LAST QUARTER MOON

☿ 263F MERCURY

♀ 2640 FEMALE SIGN

♁ 2641 EARTH

♂ 2642 MALE SIGN

♃ 2643 JUPITER

♄ 2644 SATURN

♅ 2645 URANUS

♆ 2646 NEPTUNE

♇ 2647 PLUTO

♈ 2648 ARIES

♉ 2649 TAURUS

♊ 264A GEMINI

♋ 264B CANCER

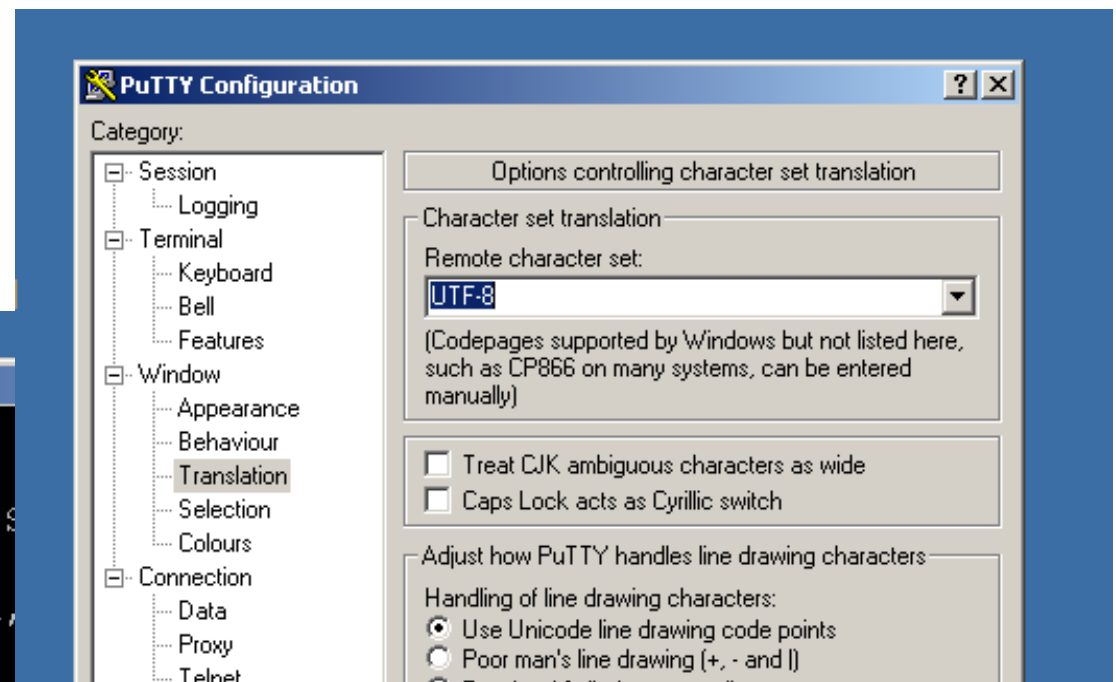
♌ 264C LEO

Done

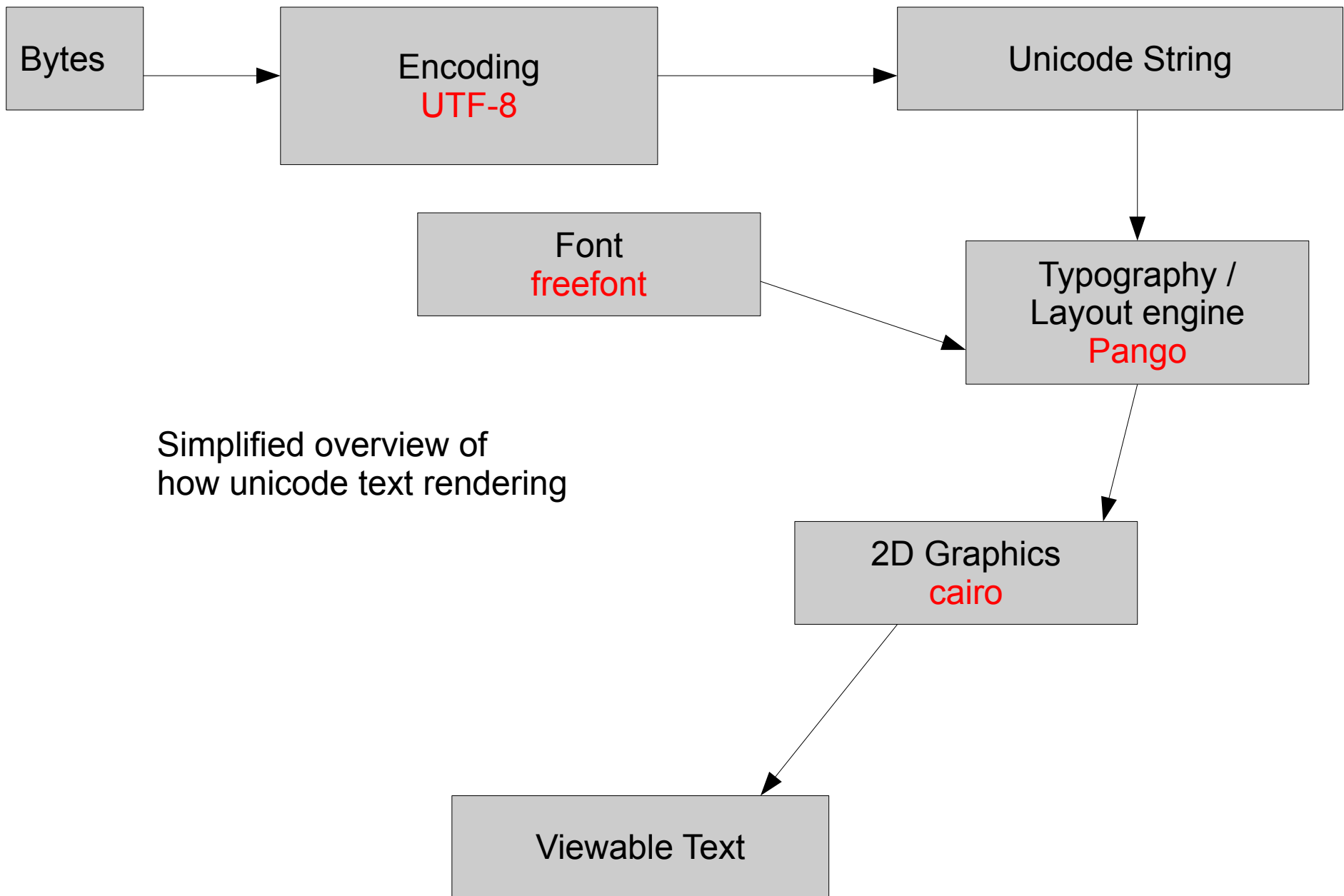
```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body>
  <h1>Some Asctrological Symbols as Unicode Characters</h1>
  <table>
    <tr>
      <td>☾</td><td>263D</td><td>FIRST QUARTER MOON</td>
    </tr>
    <tr>
      <td>☾</td><td>263E</td><td>LAST QUARTER MOON</td>
    </tr>
    <tr>
      <td>☿</td><td>263F</td><td>MERCURY</td>
    </tr>
    <tr>
      <td>♀</td><td>2640</td><td>FEMALE SIGN</td>
    </tr>
    <tr>
      <td>♁</td><td>2641</td><td>EARTH</td>
    </tr>
```

Unicode in the console

```
thomas@K-9: ~  
File Edit View Terminal Help  
thomas@K-9:~$  
thomas@K-9:~$  
thomas@K-9:~$  
thomas@K-9:~$ echo "☢ ☣ ☤ ☥"  
☢ ☣ ☤ ☥  
thomas@K-9:~$ echo "وط یررض الٹب مرولب هی ماچ"  
وط یررض الٹب مرولب هی ماچ  
thomas@K-9:~$ echo "زمروق"
```



```
thomas@K-9: ~  
login as: thomas  
thomas@192.168.12.192's password:  
Linux K-9 2.6.31-20-generic #58-Ubuntu s  
To access official Ubuntu documentation,  
http://help.ubuntu.com/  
Last login: Sun Apr  4 12:54:07 2010 from softtaco.local  
thomas@K-9:~$ echo "جام بيه بلورم بيا ضررى طوقونمز"  
جام بيه بلورم بيا ضررى طوقونمز  
thomas@K-9:~$
```



Simplified overview of
how unicode text rendering

Pango Typesetting Engine

- Advanced Unicode Features
- Integrates with Cairo
- Vertical Text Support
- Used in MathML implementations

$$a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{a_4}}}}$$

$$\binom{n}{k/2}$$

$$\binom{p}{2} x^2 y^{p-2} - \frac{1}{1-x} \frac{1}{1-x^2}$$

$$\sum_{\substack{0 \leq i \\ 0 < j}}$$

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) |\varphi(x + i y)|^2 = 0$$

$$2^{2^{2^x}}$$

$$\int_1^x \frac{dt}{t}$$

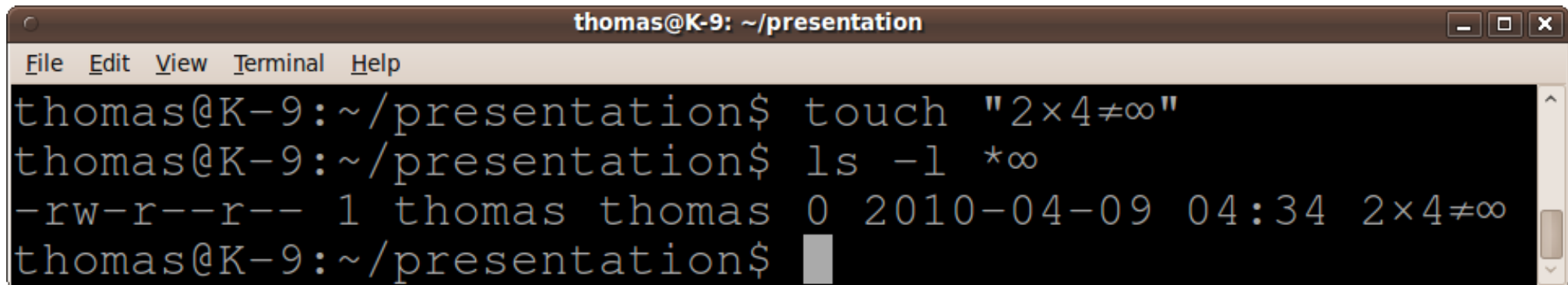
$$\iint_D dx dy$$

*images taken from www.pango.org/ScriptGallery



Unicode Filenames

- UTF-8 file names are valid in Linux
- Linux Kernel actually has to convert file names to UTF-8 from file systems that use other Unicode encodings (NTFS)

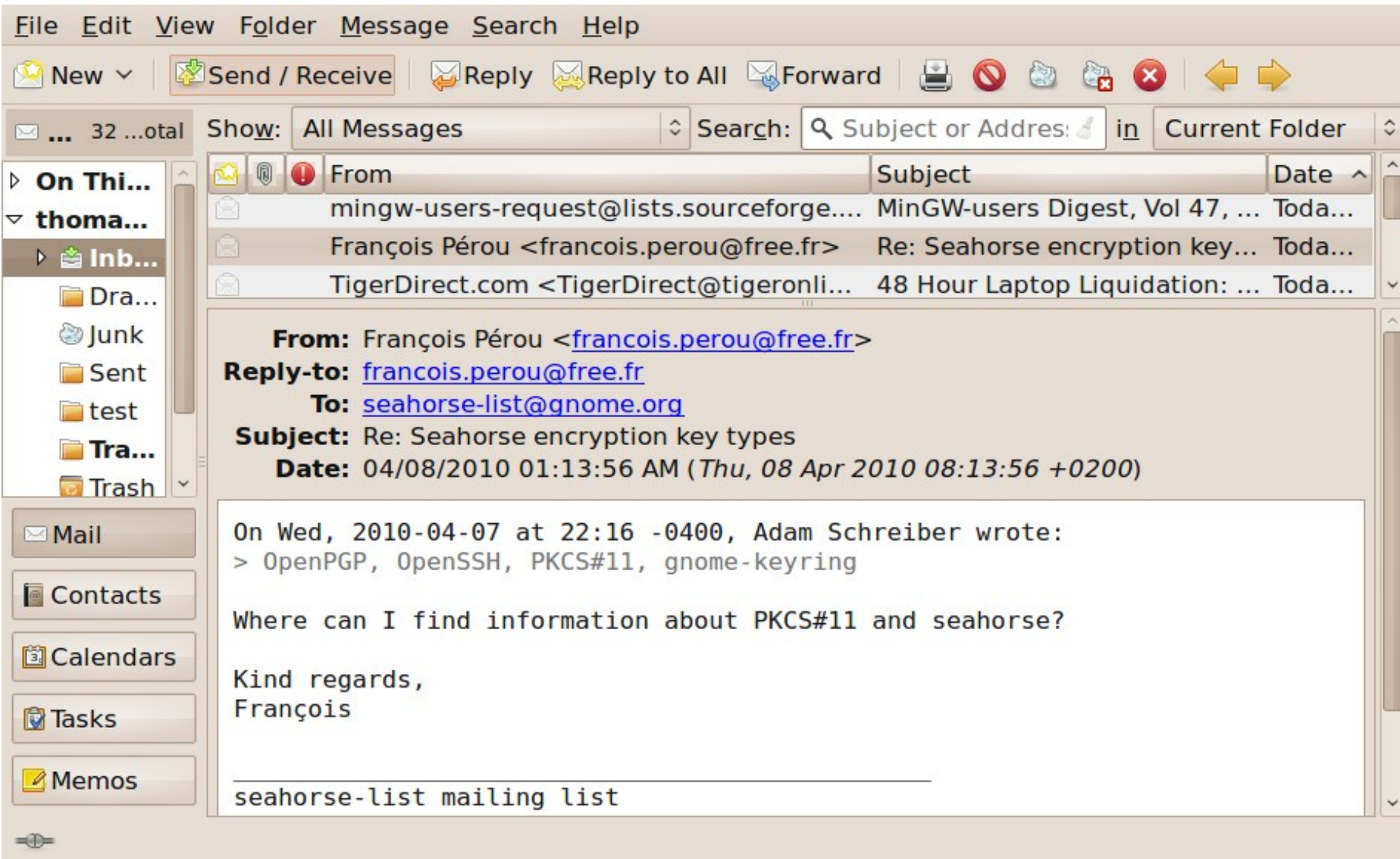


```
thomas@K-9: ~/presentation
File Edit View Terminal Help
thomas@K-9:~/presentation$ touch "2×4≠∞"
thomas@K-9:~/presentation$ ls -l *∞
-rw-r--r-- 1 thomas thomas 0 2010-04-09 04:34 2×4≠∞
thomas@K-9:~/presentation$
```

A terminal window titled "thomas@K-9: ~/presentation" with a menu bar (File, Edit, View, Terminal, Help). The terminal shows the execution of two commands: `touch "2×4≠∞"` and `ls -l *∞`. The output of the second command shows a file named "2×4≠∞" with permissions `-rw-r--r--`, owned by `thomas`, with a size of `0`, dated `2010-04-09 04:34`. The prompt is currently at `thomas@K-9:~/presentation$`.

Unicode in Email

- UTF-8 works in message bodies
- **Content-type: text/plain; charset="utf-8"**
- Attachments can of course be anything
- Base SMTP is ASCII only
- Possible use case of UTF-7 (resist the urge)
- Unicode in message headers is done with the RFC 2047 trick, that switches out of ASCII mode to specific character sets



Unicode in DNS

- Probably will break the Internet.
- Just kidding - of course it uses UTF-8!
- Could not find anyone to allow us to register ☢.us
- RFC-3490; Only lowercase letters
- Max name length by byte count not char count

```
thomas@K-9: ~  
File Edit View Terminal Help  
thomas@K-9:~$ ping ±2¢  
PING ±2¢ (127.0.2.1) 56(84) bytes of data.  
64 bytes from ±2¢ (127.0.2.1): icmp_seq=1 ttl=64 time=0.033 ms  
64 bytes from ±2¢ (127.0.2.1): icmp_seq=2 ttl=64 time=0.034 ms  
64 bytes from ±2¢ (127.0.2.1): icmp_seq=3 ttl=64 time=0.031 ms  
^C  
--- ±2¢ ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 1998ms  
rtt min/avg/max/mdev = 0.031/0.032/0.034/0.006 ms  
thomas@K-9:~$
```

Quick Concepts

Internal Encoding –

The low level format your program uses internally for strings.

- Sometimes more than one is needed
- May or may not be the way strings are presented in source code

External Encoding –

The format a string must be for a given input, output, library usage.

- Zero to Many might be needed
- Usually requires the most attention

Internal Encodings

UTF-16

- Various Native Windows
- Qt
- normal* Python
- wxWidgets
- .Net
- Java

UTF-8

- Gtk+
- Posix style C
- shell scripts
- PHP (we think)

UTF-32

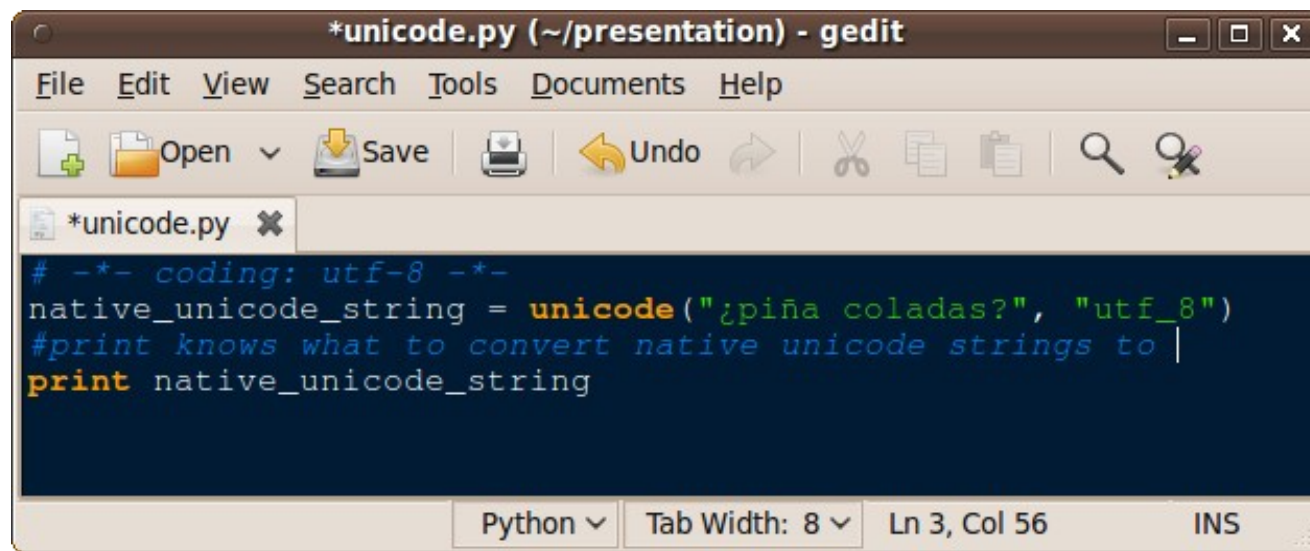
- Advanced text programs
- special build of Python

Conversion Options

- Built in Functions (most non-C environments)
- Iconv
- ICU (IBM)
- Manually

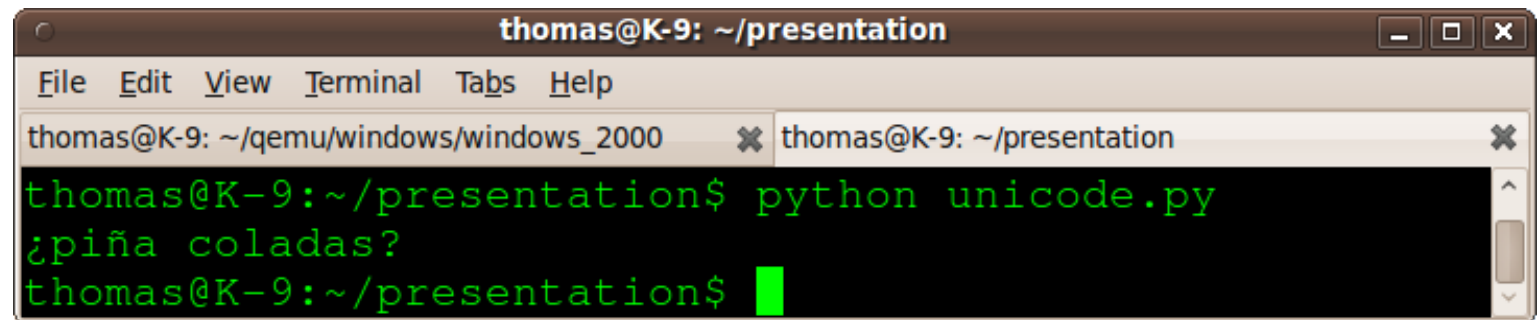
External Encodings

- EBDIC
- ASCII
- UTF-* LE/BE
- Old ISO-*codepages

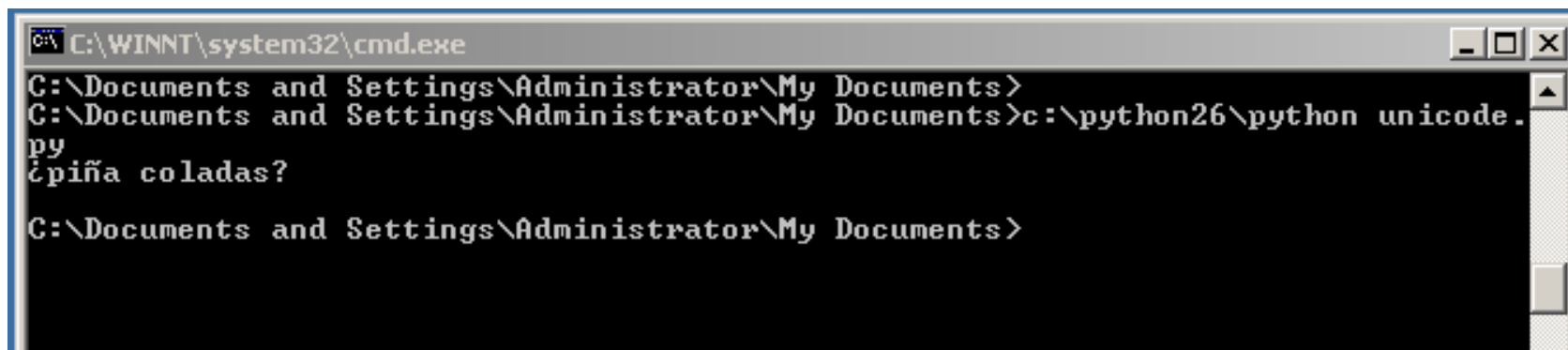


```
*unicode.py (~/.presentation) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
*unicode.py x
# -*- coding: utf-8 -*-
native_unicode_string = unicode("¿piña coladas?", "utf_8")
#print knows what to convert native unicode strings to
print native_unicode_string
Python Tab Width: 8 Ln 3, Col 56 INS
```

Unicode in Python



```
thomas@K-9: ~/.presentation
File Edit View Terminal Tabs Help
thomas@K-9: ~/qemu/windows/windows_2000 x thomas@K-9: ~/.presentation x
thomas@K-9:~/presentation$ python unicode.py
¿piña coladas?
thomas@K-9:~/presentation$
```



```
C:\WINNT\system32\cmd.exe
C:\Documents and Settings\Administrator\My Documents>
C:\Documents and Settings\Administrator\My Documents>c:\python26\python unicode.py
¿piña coladas?
C:\Documents and Settings\Administrator\My Documents>
```

unicode-gtk.py (~/.presentation) - gedit

File Edit View Search Tools Documents Help

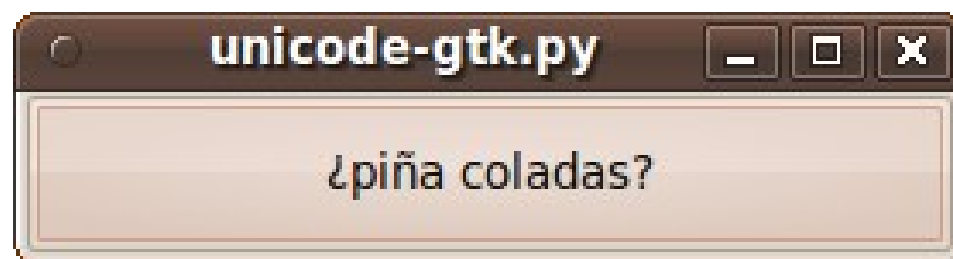
Open Save Undo

unicode-gtk.py

```
import pygtk
pygtk.require('2.0')
import gtk

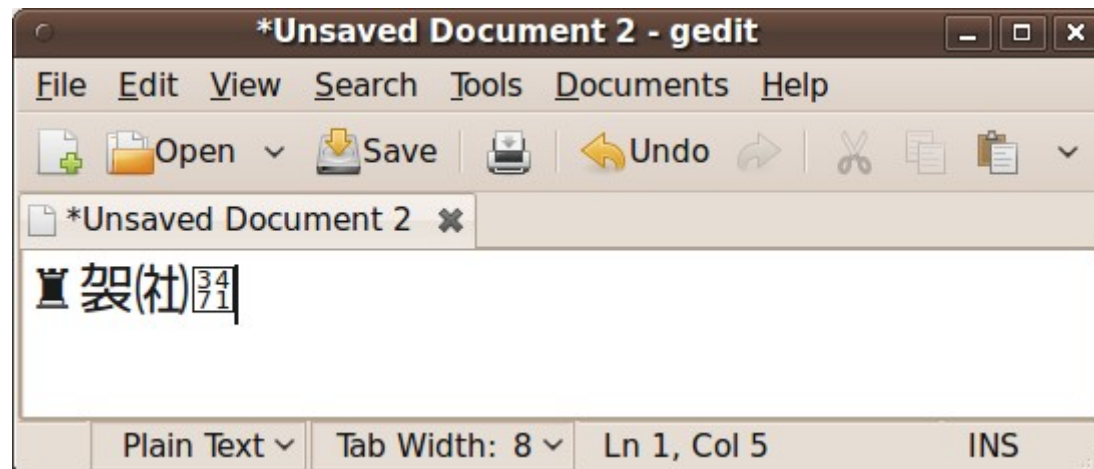
native_unicode_string = unichr(191) + "pi" + unichr(241) + "a coladas?"
window = gtk.Window(gtk.WINDOW_TOPLEVEL)
#native unicode strings transparrently are trasnscoded as needed
button = gtk.Button(native_unicode_string)
window.add(button)
window.show_all()
gtk.main()
```

Python Tab Width: 8 Ln 4, Col 1 INS



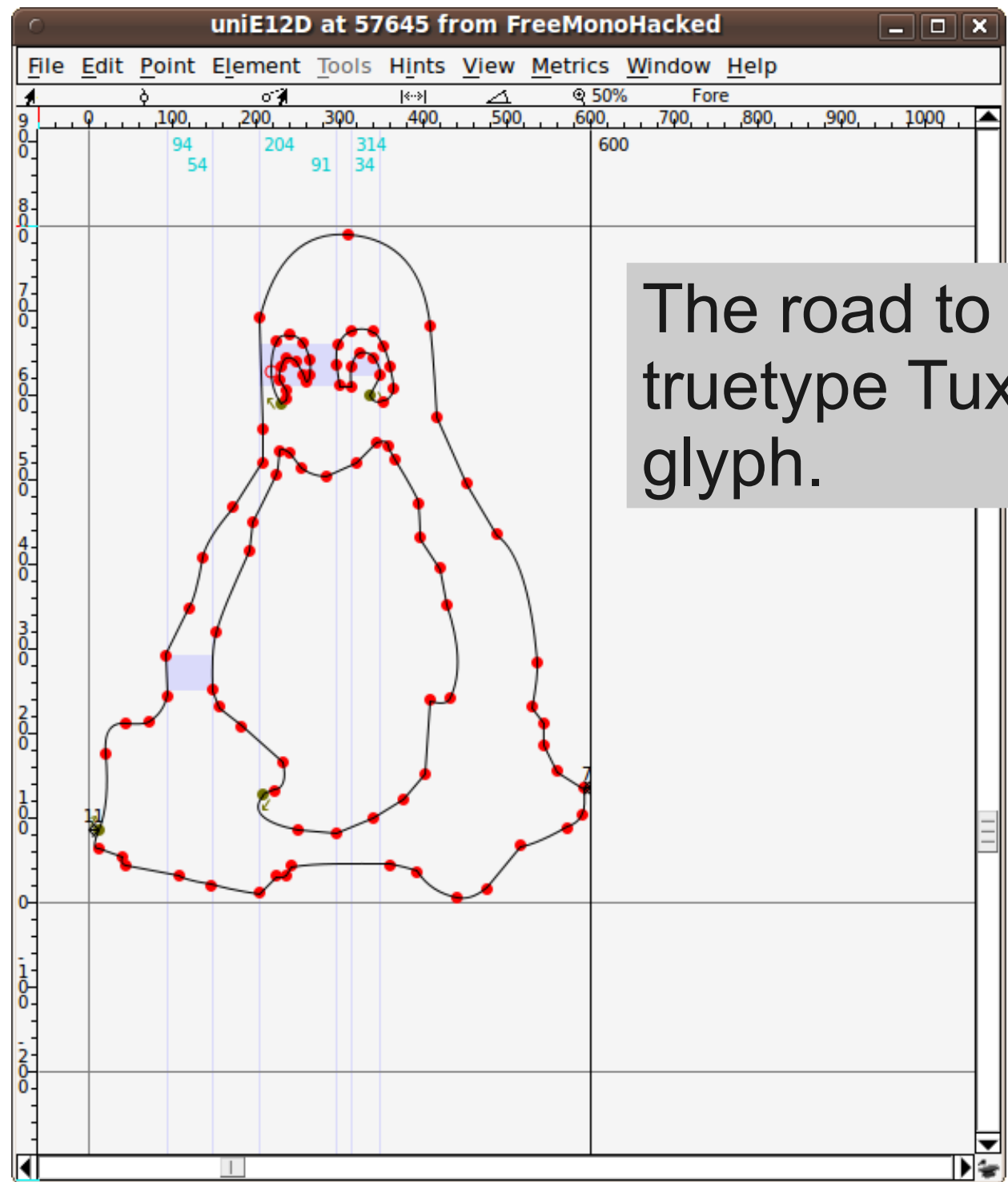
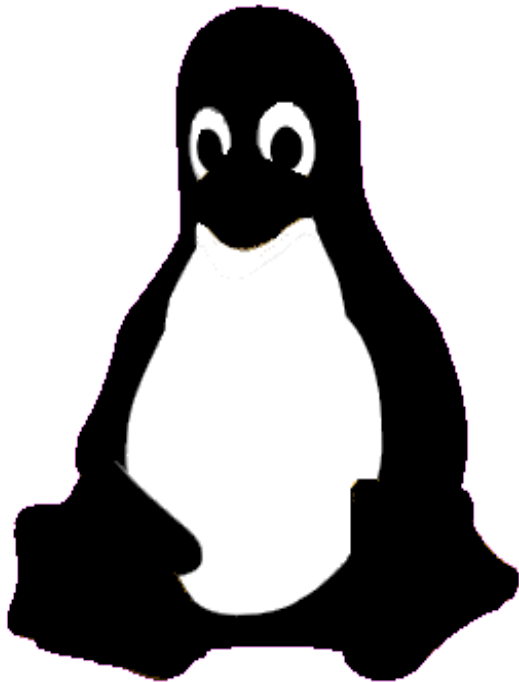
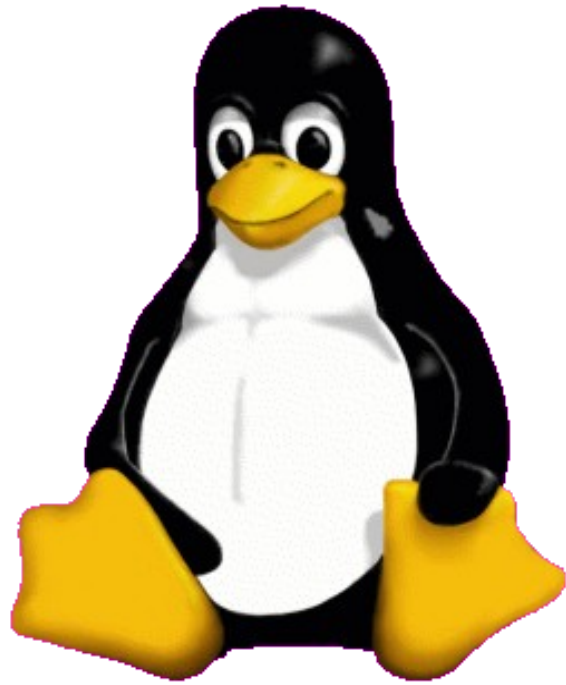
“unicode fonts”

- GNU FreeFont / Free UCS Outline Font (GPLv3 with “Font Exemption”)
- GNU Unifont (GPL) 27,000 characters
- No existing font file format can support all characters

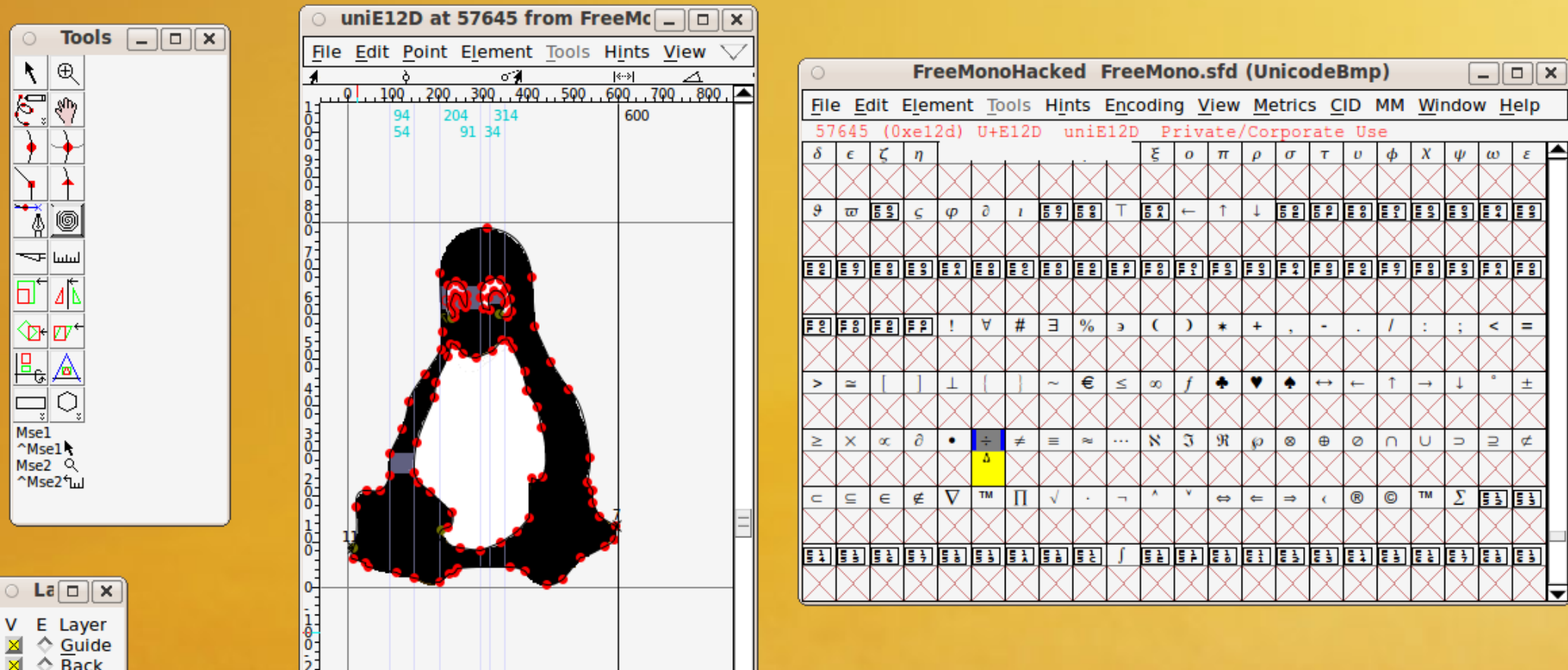


Private Use Area (PUA)

- Reserved sections of the address space for personal / corporate / freestyle use.
- One of these sections is in the more compatible “code page 0” / “multilingual plane” range (U+E000 to U+F8FF)
- Many unofficial standard uses:
 - Klingon
 - Medieval characters
 - Other math symbol efforts
 - Personal names in Asian Glyphs
 - Not yet accepted languages



Font Forge F/OSS Font Editing Suite

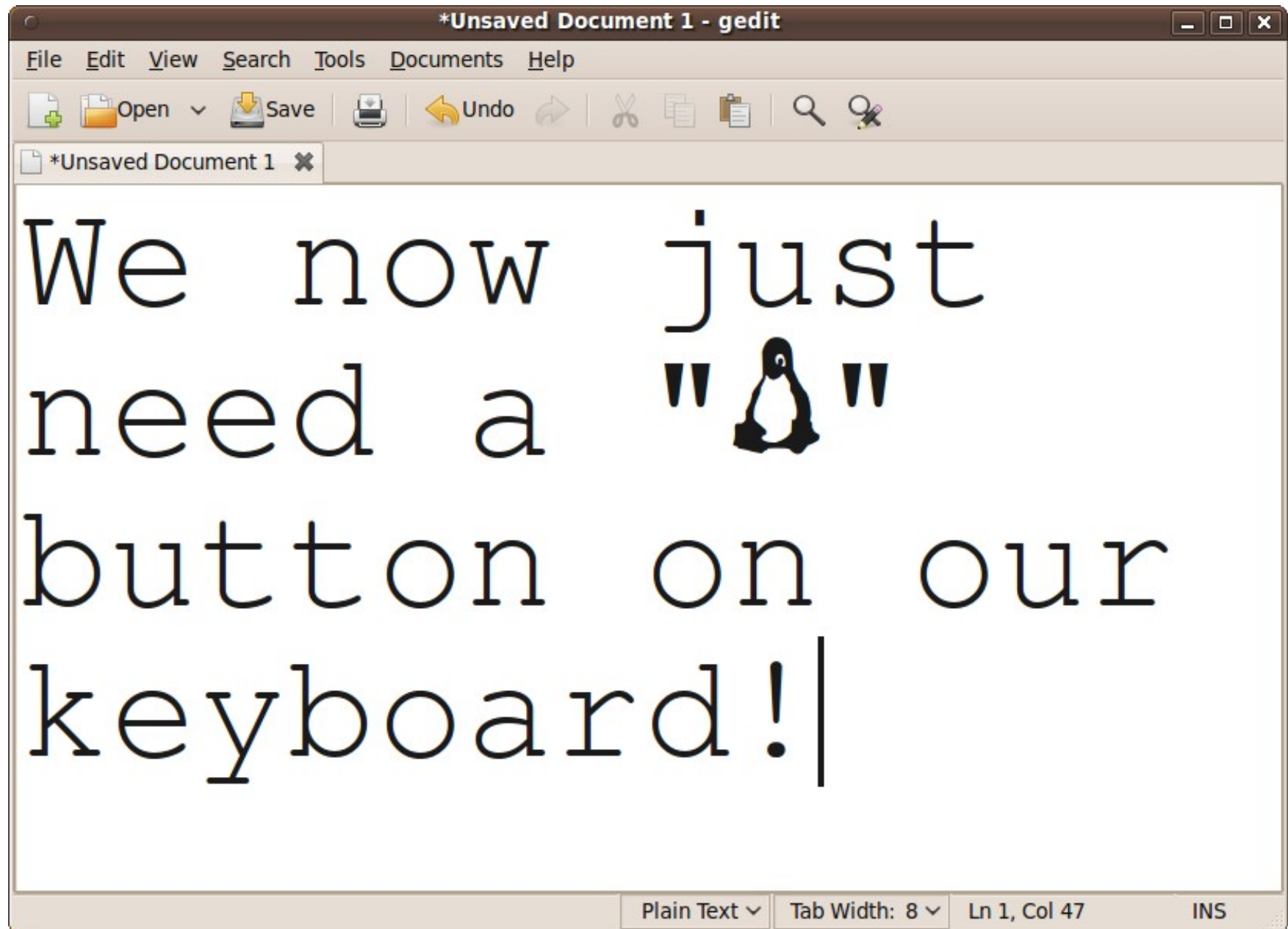


fontforge.sourceforge.net

```
thomas@K-9: ~$ echo "🐧🐧"  
🐧🐧  
thomas@K-9: ~$ ue12d
```

Remember, in Gtk+ apps:
ctrl+shift+u, then hex code

Where is tux's right eye?
Complexities of true type?



“Advanced” Unicode Operations

- Case Conversions / Collations
- Combining Characters
- Control Characters
- Variable Spaces
- Bi-directional text control
- Fraction Slash
- Script Specific (ie Music Notation Format)
- Surrogates (UTF-16 retrofitting)
- Simple String Length (not so simple)
- Encoding Sanity Check / Verification
- Numeric Character to Numerical Value