# Language support in Slackware

by Eric Hameleers
http://slackware.com/~alien/
alien@slackware.com

Presented at:
Universidade Presbiteriana Mackenzie
for the Slackware Show
Encontro Nacional de Usuarios Slackware
http://slackshow.slackwarebrasil.org/

August 21–22, 2008

# Overview of this presentation

Language support and input methods

- Introduction
- Computers and texts
- Internationalization and localization
- Linux for non-english people
- Unicode in the console
- Unicode in X-Window
- Other resources

# Overview of this presentation

Language support and input methods

- Introduction
- Computers and texts
- Internationalization and localization
- Linux for non-english people
- Unicode in the console
- Unicode in X-Window
- Other resources

We will see how “SCIM” can be used to input complex characters in your X applications.

# Introduction

Eric Hameleers - Slackware user since 1996

# Computers and texts

- In early days, computers used 7-bit ASCII for text
- Other European fonts required 8 bits per character
- Unicode was invented to work for all language texts

# Unicode

- Unicode does not define how characters are displayed
- Unicode allows for more than a million characters
- Unicode characters are called “code points”
- Several standards of encoding exist which define how to store characters. UTF-8 is the most well-known
- In UTF-8, ASCII characters use up 1 byte; Latin-1 characters 2 bytes; complex characters 3 bytes
- This makes UTF-8 compatible with ASCII texts (no size difference)

# Locales

- Locale is the set of parameters that define how a computer is localized
- Locale parameters:
  language, country, currency, number formats and more

**List your available locales:**

```
locale -a
```

**find out which locale is currently active:**

```
locale
```

# Locales (cont'd)

```
LANG=en_US.UTF-8
LC_CTYPE="en_US.UTF-8"
LC_NUMERIC="en_US.UTF-8"
LC_TIME="en_US.UTF-8"
LC_COLLATE=C
LC_MONETARY="en_US.UTF-8"
LC_MESSAGES="en_US.UTF-8"
LC_PAPER="en_US.UTF-8"
LC_NAME="en_US.UTF-8"
LC_ADDRESS="en_US.UTF-8"
LC_TELEPHONE="en_US.UTF-8"
LC_MEASUREMENT="en_US.UTF-8"
LC_IDENTIFICATION="en_US.UTF-8"
LC_ALL=
```

# Locales (cont'd)

- Locate settings are defined in "/etc/profile.d/lang.sh" (globally) or `/.profile` (per-user setting)
- Example: I want my desktop to display messages and menus in English but use Dutch localization (measurements, currency, time display, word sort, paper sizes and such):

# Locales (cont'd)

I will have to add this to my startup scripts:

```
LANG="en_US"
LC_MESSAGES="en_US"
LC_CTYPE="nl_NL@euro"
LC_COLLATE="nl_NL@euro"
LC_TIME="nl_NL"
LC_NUMERIC="nl_NL"
LC_MONETARY="nl_NL@euro"
LC_PAPER="nl_NL"
LC_TELEPHONE="nl_NL"
LC_ADDRESS="nl_NL"
LC_MEASUREMENT="nl_NL"
LC_NAME="nl_NL"
```

# Internationalization and localization

Internationalization and localization are related to the ability of software to be used with different languages and regions.

- Internationalization of software (abbreviated to i18n) means that it uses libraries which enable support for new languages without having to rewrite the complete application.
- Localization of software (abbreviated to L10n, note the capital "L") means that you add text translations and locale-specific elements.

# Internationalization and localization

- All major Free and Open Source (FOSS) initiatives support the *i18n* and *L10n* standards:
- **KDE, Gnome, OpenOffice, Mozilla, ...**
- and many of the smaller FOSS projects implement support as well.

# Linux for non-english people

With locales, UTF-8, i18n and L10n, we have the basics covered. We can fully customize our Slackware computer for use in our own country.

We start with enabling Unicode.

- define a UTF-8 locale for at least the language and messages.
- Easiest way is to edit /etc/profile.d/lang.sh and change:

```
export LANG=en_US
#export LANG=en_US.UTF-8
```

to

```
#export LANG=en_US
export LANG=en_US.UTF-8
```

and login again.

# Linux for non-english people

- Use your own language instead of "en_US" of course. For this presentation, we will assume "en_US.UTF-8".
- **Note**: the extension to your language setting must be ".UTF-8". When you look at the output of the `locale -a` command, you will see "en_US.utf8" listed.
  Be careful not to copy and paste this value from the **locale -a** output! Your Unicode support will not be enabled.

The next sections describe:

- How to display Unicode in the Linux console
- How to display and input Unicode in X Window

# Unicode in the console

Using non-latin text in the console is made easier starting with the Linux 2.6.24 kernel. The linux console is Unicode-enabled by default. In earlier days, you had to run

```
unicode_start
```

in a console to enable Unicode support.

Slackware disables unicode in `/etc/lilo.conf` (user selectable during installation):

```
append="vt.default_utf8=0"
```

# Unicode in the console

You will need a console font with decent Unicode coverage. Example: **terminus font** (not part of Slackware).
To load this font after you installed it, you need to add the following to your `/.profile` file:

```
if [ "$TERM" == "linux" ]; then
setfont ter-v16n
fi
```

The default Slackware console font will be loaded with UTF-8 extensions if you run the command:

```
setfont -v
```

which also gives good results.

# Unicode in the console

There are good reasons why Slackware does not come with a Unicode-enabled console by default.

- The collection of tools on your system expects input in plain roman text (ASCII text). Entering a Unicode text string on the command prompt can have unexpected results if the shell interprets this string and executes the command.
- Several terminal-based (curses) applications will not display correctly - especially in case of graphical characters. This is caused by the double size of these Unicode characters.

**Show characters of your console font:**

```
showconsolefont
```

# Unicode in X-Window

Slackware has always been able to show non-english text, thanks to the available locales and the internazionalization of many of the applications that come with it. However:

- The installer is available only in english.
  This will not change.
- The consoles are not Unicode-enabled by default.
  This may change in future, when all of the console applications are able to display their text and interface correctly.
- No decent support existed for non-european languages. Specifically, CJK (Chinese/Japanese/Korean, or just 'Asian language') support.
  This has changed in Slackware 12.1.

# Unicode in X-Window (cont'd)

What was the status before Slackware 12.1?

- Slackware did not have high-quality fonts with full CJK coverage
- TrueType font rendering of double-width fonts like Chinese was messy and lacked proper bold-face.
- Text input for these languages was difficult.
- Several initiatives to add CJK language support existed which required extensive patching of core Slackware packages (X, fontconfig, freetype)
- The "best" looking fonts (or even, good-looking fonts at all) were all produced by Microsoft and thus, non-free.

# Unicode in X-Window (cont'd)

What happened during the development of Slackware 12.1?

- In Slackware 12.1 we made an effort to converge all the missing pieces into the core distribution.
- By 2007, CJK font rendering in freetype, fontconfig and the X libraries had advanced to a point where patching of the vanilla sources was no longer required.
- Several excellent free TrueType Unicode fonts were added.
  - Redhat Liberation fonts (MS core font replacement)
  - Zen Hei (Wen Quan Yi family - Chinese coverage)
  - Sazanami (Japanese coverage)
  - Sinhala (for display of Sanskrit/Sri Lankan texts)
  - TibMachUni (Tibetan Machine Unicode font)
- Crucial was the addition of keyboard input methods!

# SCIM

- Keyboard input methods allow the user to enter extended (non-latin) characters using a standard keyboard.
- Several projects exist that implement Input Methods, like UIM and IIIMF, fcitx, and SCIM.
- We adopted the SCIM (Smart Common Input Method) platform because it is widely adopted among other distributions, is actively developed and covers a lot more than just CJK input.
  These factors make it the safe choice.

How do we use SCIM for working with Unicode texts?

Read HINTS_AND_TIPS.TXT on the Slackware CD!

# SCIM (cont'd)

- The first requirement is to use a UTF-8 locale. We covered this earlier on. The scim daemon will not start if it detects lack of UTF-8 support.
- Make the scim profile scripts executable. These will setup your environment correctly for the use of scim with X applications. Run this command as root:

```
chmod +x /etc/profile.d/scim.*
```

# SCIM (cont'd)

- Start the scim daemon as soon as your X session starts. The scim daemon must be active before any of your X applications start.
  - In KDE, you can add a shell script to the `/.kde/Autostart` folder that runs the command "scim -d".
  - In XFCE you can add "scim -d" to the Autostarted Applications.
  - If you boot your computer in runlevel 4 (the graphical XDM/KDM login) you can add the line "scim -d" to your `/.xprofile` file. This method is independent of the Desktop Environment you use.
- GTK apps like firefox will crash in case you remove or forget to install the **scim-bridge** package, so take care which packages you leave out. A full Slackware installation is always recommended if your hard drive has the available space.

# Using SCIM

- When SCIM is up and running, you will see a small keyboard icon in your system tray.
- The key sequence to activate SCIM input is: <**Cntrl**><**Space**>
- SCIM offers input methods for many other languages like Greek, Russian, even accented German is possible.
- In addition to the use of SCIM, GTK is Unicode friendly and allows to enter unicode characters directly.
  If you know the number of a code point (Unicode character) you can input this character into any GTK input field, using the key combination of <**Ctrl**>-**U** + **number**.

# Other resources

- UTF-8 explained on Wikipedia
  `http://en.wikipedia.org/wiki/UTF-8`
- Gentoo's HOWTO on using Unicode
  `http://gentoo-wiki.com/HOWTO_Make_your_system_use_unicode/utf-8`
- Good article on locales
  `http://publib.boulder.ibm.com/infocenter/systems/index.jsp?topic=/com.ibm.aix.nls/doc/nlsgdrf/locale_env.htm`

# About the author

I am a Linux user since the 0.99 kernel releases and ran into Slackware Linux around 1996. The first need to make modifications to Slackware occurred when I joined IBM and was doomed to use Token Ring for which there was no support in the network scripts :-)
I still help with the Slackware develoment in 2008. I also maintain a repository of non-official Slackware packages and scripts, I am one of the admins of the SlackBuilds.org website, and sometimes like to write articles for my own Wiki. I was born in the Netherlands and still live there, with wife, son and several small animals.

Feedback and suggestions are welcome
`alien@slackware.com`

A copy of this presentation can be found at:
`http://www.slackware.com/~alien/slackshow2008/`