Creating custom keyboard layouts for X11 using XKB

License

Disclaimer

Tips

Download ready-made keyboard layouts

Instead of manually modifying keyboard layouts, as described in the article, you can download ready-made layout descriptions and start right with the part which describes where they should be placed and what other files need to be modified. Two layouts are available here:

- `de_pl` — the layout used as an example in the article (physical layout: German, logical German & Polish)
- `pl_de` — physical layout: Polish (= US English), logical: Polish & German (Umlauts, ß and bound to AltGr+1, AltGr+2 etc.)

Differences between X.org and XFree86 X servers, differences in file path between distributions

A lot of people, including myself, have switched from XFree86 to X.org by now, but the article was originally written with XFree86 in mind. If you are using X.org X server instead of XFree86, file `/etc/X11/xorg.conf` should be modified instead of `/etc/X11/XF86Config` and `/etc/X11/xkb/rules/xorg.{lst,xml}` instead of `/etc/X11/xkb/rules/xfree86.{lst,xml}`.

Ubuntu and Kubuntu use slightly different paths and as of 2011 place XKB configuration in `/usr/share/X11/xkb` prefix. I haven't checked on Debian but I guess paths will be the same as in the 'buntus. Another difference from what the text below describes is that the keyboard definition files are not placed in `[XKBROOT]/symbols/pc` but directly in `[XKBROOT]/symbols`. This has an additional consequence. Layout files use includes - for example most European layouts include one of the basic Latin alphabet layouts. This is a line which looks like:

```
    include "pc/latin(type4)"
```

On systems that don't use the `pc/` subdirectory, this line in the sample layouts `pl_de` and `de_p` presented in this article, has to be changed to:

```
include "latin(type4)"
```

Otherwise, the layout won't work.

Assigning unnamed characters to keystrokes

It is possible to assign arbitrary Unicode characters, even those which don't have a name, to keystrokes. For example, a line such as:

```
 key <AC07> { [         j,          J,          U263A,          U263B ] };
```

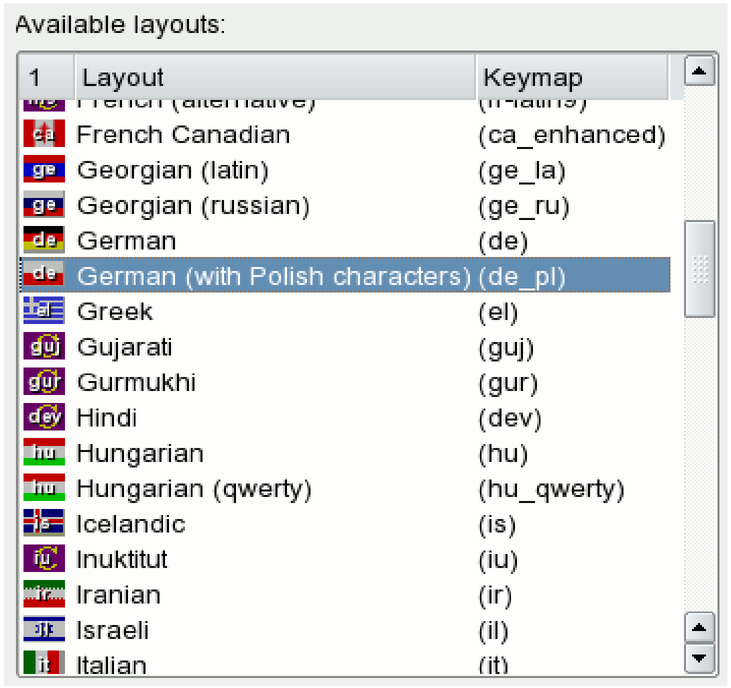
binds the Unicode characters 263A and 263B (white and black smiley faces) to AltGr+J an AltGr+Shift+J.
Thanks to Ossi Viljakainen for pointing this out.

Creating custom keyboard layouts for X11 using XKB

In most modern desktop environments there is a small applet that allows users to quickly switch between keyboard layouts when they need to type text in more than one language. However, there are situations where this solution is not quite satisfactory. Some writing jobs (that of a translator, for example) require writing text in which words from several languages are intermixed, which would make frequent switches between keyboard layouts necessary. This is a big inconvenience, even when keyboard shortcuts are used to switch between layouts. Another problem is that the logical keyboard layouts used may sometimes not fit the physical keyboard layout very well, resulting in a confusing setup.

One solution is to use layouts with so called dead keys. This is a way of generating accented characters by first pressing a key which corresponds to the accent (the dead key) and then a letter key (e.g. pressing ~ and then n generates ñ). This allows you to generate many accented characters using just a few keys, since a dead key need not be a single keystroke but can also be a combination of keys. For example, the backtick key is often used as a dead key and generates four different accents depending on whether it's pressed alone or with combinations of Shift and AltGr (the right Alt key). This is a clever trick, but writing using dead keys is usually less convenient than using a simple combination like AltGr+letter. Dead keys are a good solution for characters that are seldom used, but for characters that need to be typed often, we need a more convenient technique.

The X Window System used on most Unix-like systems today uses X Keyboard Extension (XKB) for translating keystrokes into character codes. Thanks to XKB's flexibility, one can easily create custom keyboard layouts that can help solve the problems mentioned in the preceding paragraphs. This tutorial describes how to create a custom keyboard layout combining diacritic characters from several standard layouts, thus eliminating the need to switch keyboard layouts while writing. On the right is an image showing the final result: a custom keyboard layout called `de_pl`, listed together with standard layouts in the KDE Control Center.



Available layouts:

1	Layout	Keymap
	French (alternative)	(fr-latin9)
ca	French Canadian	(ca_enhanced)
ge	Georgian (latin)	(ge_la)
ge	Georgian (russian)	(ge_ru)
de	German	(de)
de	German (with Polish characters)	(de_pl)
el	Greek	(el)
guj	Gujarati	(guj)
gur	Gurmukhi	(gur)
dev	Hindi	(dev)
hu	Hungarian	(hu)
hu	Hungarian (qwerty)	(hu_qwerty)
is	Icelandic	(is)
iu	Inuktitut	(iu)
ir	Iranian	(ir)
il	Israeli	(il)
it	Italian	(it)

Throughout this document I will use a keyboard layout combining Polish and German diacritic characters as an example. Standard layouts for Polish and German are quite different, and switching between them while writing is very inconvenient. The Polish layout is essentially the same as a standard US keyboard, the only difference being Polish diacritic characters (ĄĆĘŁŃÓŚŻŹ) which are generated using AltGr and the corresponding nonaccented letter (AltGr+X is used for Ź). The German layout is radically different, since it is a QWERTZ layout. The differing position of Y and Z alone is enough to make the writer's life really hard. But that's not all: separate keys are reserved for German umlauts ÄÖÜ and ß, causing almost all punctuation and special characters like &, +, /, <, >, etc., to have a different position than in the Polish layout. Writing any text where characters from both languages appear and having to switch between the two layouts is very inconvenient.

Our goal is to create a keyboard layout combining accented letters from both languages that fits well to the physical layout of a German keyboard. Thus we want it to essentially be a German layout (preserving the position of punctuation marks and umlauts as well as of Y and Z) but we want to bind additional AltGr+letter combinations to Polish accented characters.

XKB layouts are defined by text files living in `/etc/X11/xkb/`. On some systems they may be placed in `/usr/X11R6/lib/X11/xkb/` (on my machine, the later is a symlink to the former directory). Newer versions of Fedora and possibly other RedHat-based distributions use the location `/usr/share/X11/xkb`. There are several subdirectories in that folder, but for the simple task of creating a new layout we are interested only in the `symbols` subdirectory. Root permissions are necessary for most tasks described below, as all configuration files in this directory are owned by root. Note that some of the files we need to edit are read-only even for the superuser, so it may be necessary to change their access permissions before editing (most text editors allow saving read-only files after confirming the operation, though).

Files to describe the basic keyboard layouts are given the names of their language name abbreviations (e.g. `de` for German). Names such as `ge_la` are used for keyboard variants (`la` (Latin) variant of the `ge` (Georgian) keyboard). Thus, in order to follow this convention, we use `de_pl` for our new keyboard layout instead of e.g. `depl`. Using this naming scheme is not mandatory — the layout would also work under another name — but recommended. Some applications, such as the KDE Control Center, also rely upon this naming scheme in order to assign icons (national flags) to keyboard layouts.

We go to the `/etc/X11/xkb/symbols/pc/` directory (if a non-PC keyboard model is to be used, all actions should be performed in `/etc/X11/xkb/symbols/` instead) and copy the file `de` to `de_pl` in the same directory.

Let's make some modifications to our new layout. Open `de_pl` in a text editor and find the section that assigns character codes to keys. It should look similar to this (only a small part is quoted here for brevity):

```
---CUT---
key <AE02> { [          2,   quotedbl,    twosuperior,      oneeighth ] };
key <AE03> { [          3,    section,  threesuperior,       sterling ] };
key <AE04> { [          4,     dollar,     onequarter,       currency ] };
key <AE11> { [     ssharp,   question,      backslash,   questiondown ] };
key <AE12> { [dead_acute, dead_grave,   dead_cedilla,    dead_ogonek ] };
key <AD03> { [          e,          E,       EuroSign,       EuroSign ] };
key <AD06> { [          z,          Z,      leftarrow,            yen ] };
key <AD11> { [udiaeresis, Udiaeresis, dead_diaeresis, dead_abovering ] };
key <AD12> { [       plus,   asterisk,     dead_tilde,    dead_macron ] };
---CUT---
```

The meaning of these lines is obvious — they assign characters to key positions on the keyboard. Fo example <AE03> means the third alphanumeric key in the fifth row (the letters A, B, etc., denot key row numbers counted from the bottom of the keyboard). Special keys have their own names - e.g. <ESC>, <TAB>, <RTRN> etc. Character names are usually self-explanatory and in most case identical or very similar to Unicode names. A complete list of all available character names can b found in file `/usr/X11/include/X11/keysymdef.h` in the form of constant definitions for (programs.

Each line of the form `#define XK_xyz` corresponds to the character name `xyz` that can be used in keyboard layout definition files. However, usually it is easiest to just copy some lines directly from other keyboard layouts (in this case, from file `pl`, which defines the Polish layout). The four character names assigned to each key are characters generated when the key is pressed alone, with the Shift key pressed, with the Multi_key key (which is defined to be AltGr in this layout) pressed, and with both Shift and Multi_key pressed, respectively.

For example, we can use AltGr+Z to generate the character Ż (lower/uppercase depending on Shift state) instead of a left-pointing arrow and the yen symbol by changing the corresponding line to:

```
key <AD06> { [         z,          Z,    zabovedot,    Zabovedot ] };
```

In a similar manner, we can add or modify the corresponding characters' descriptions to get all the other Polish characters. (Ł is bound to AltGr+L in the German layout already so there's no need to add it.):

```
key <AD03> { [         e,          E,      eogonek,      Eogonek ] };
key <AD09> { [         o,          O,       oacute,       Oacute ] };
key <AC01> { [         a,          A,      aogonek,      Aogonek ] };
key <AC02> { [         s,          S,       sacute,       Sacute ] };
key <AB02> { [         x,          X,       zacute,       Zacute ] };
key <AB03> { [         c,          C,       cacute,       Cacute ] };
key <AB06> { [         n,          N,       nacute,       Nacute ] };
```

As the last touch, we bind the euro sign to AltGr+5, since the original AltGr+E binding is used for Ę now:

```
key <AE05> { [         5,    percent,     EuroSign,     EuroSign ] };
```

If you happen to once in a while use characters from many different languages (which you don't use often enough to bother creating separate keybindings for), you may find using the Multi Key (also called Compose Key) very useful. This is a special key, on PC keyboards usually assigned to the right Windows Key, which allows typing accented characters by entering a character corresponding to some accent and then an unaccented letter. For example, by typing Compose, then the comma and then C, you generate the C-cedilla (Ç). Note that you need to press the keys in sequence — Compose doesn't work like Alt or Shift which you press together with another key; you have to release Compose before typing in the next character. Similarily to creating C-cedilla, you can generate letters such as à, á, â and ä by composing a with the backtick, the apostrophe, the ^ character and the quotation mark ("), correspondingly. By composing the semicolon (;) with e or a you can get the Polish "ogonek" as seen in the letters Ę and Ą. Using Compose with the slash (/) you can type Ø, Ł or Ŧ. The tilde (~) can be composed with N to generate the Spanish Ñ while Compose followed by two question marks (?) gives the inverted question mark ¿. Many other ligatures and special characters, including punctuation, can be generated in a similar manner. Composing the letter O with R or C gives the registered trademark and copyright signs ® and ©. Composing S with another S gives Scharfes-S (ß) and composing A with E results in Æ. Composing two less-than or two greater-than signs produces opening or closing guillemets « and » while composing the percent sign with the letter O results in the permille sign ‰. Using this technique, the majority of all accented and special letters used in European languages can be typed. See this file for a complete list of compositions available in X.org X-server (look for lines starting with `<Multi_key>`). Other X servers may support slightly different combinations.

Some keyboard layouts make the right Windows Key function as Compose Key by default while others don't, so just to make sure you can use this functionality, add the following line to your keyboard layout definition:

```
key <RWIN> { [  Multi_key ] };
```

This makes our new keyboard layout description complete.

The last step is letting X know that a new layout was added. The file `/etc/X11/xkb/rules/xfree86.lst` contains a list of available layouts together with their descriptions. Adding the below line:

```
de_pl       German (with Polish characters)
```

makes our newly created layout usable in X. In order to make the keyboard layout available in GNOME, you should also modify `/etc/X11/xkb/rules/xfree86.xml` accordingly (this is an XML file and the format is mostly self-explanatory). If you want to make the new layout the default (which is probably the case), edit `/etc/X11/XF86Config` (sometimes `/etc/X11/XF86Config-4` is used) and modify the `InputDevice` section for your keyboard to contain the line:

```
  Option "XkbLayout" "de_pl"
```

After modifying these files, restart the X server by pressing Ctrl+Alt+Backspace. The new layout should now be loaded if you made it the default in `XF86Config`. If not, you can select it from the list of layouts in your favorite keyboard layout switching applet.

History

- 2011-05-25 — version 1.6; added some information about Ubuntu.
- 2007-01-02 — version 1.5; added information about configuration file paths used by nev versions of Fedora.
- 2006-09-24 — version 1.4; added information about using the Compose (Multi) key; the file `de_pl` and `pl_de` were modified by adding a Multi Key binding.
- 2006-01-15 — version 1.3; added information about modifyin `/etc/X11/xkb/rules/xfree86.xml` in order to make the layout visible to GNOME. Thanks t mirror (postm9am (at sign) plk (dot) mff (dot) cuni (dot) cz) who provided me with thi information.
- 2004-11-07 — version 1.2; article is double-licensed under the Creative Commons Attribution ShareAlike License and the GNU Free Documentation License
- 2004-09-08 — version 1.1; article is published on author's homepage
- 2004-06-10 — version 1.0; article is first published on linux.com

See also:

- *Creating custom keyboard layouts for X11 using XKB* as published on linux.com
- Computing main page
- A more detailed guide to XKB configuration
- Basic XKB configuration
- How to further enhance XKB configuration
- A list of characters that can be generated using the Compose Key in X.org (look for line starting with `<Multi_key>`)