thinkpad-scripts Documentation

Release 4.10.0

Martin Ueding and Jim Turner

Mar 19, 2017

Contents

Welcome to the documentation for thinkpad-scripts, a set of scripts to automate a variety of tasks on the ThinkPad X220 Tablet.

If you want to get thinkpad-scripts up an running, then check out the *Getting Started* guide.

In case you already have thinkpad-scripts set up and just want a quick reference, see the *manual pages*.

Should there be something working unlike it should on your system and you can't figure out a solution from the documentation, check the issue tracker to see if it is a known problem. If it's not there, please create a new issue on the issue tracker.

See also:

Project website

- Tar archives with source checkouts

GitHub page

- Issue tracker
- git repository

CHAPTER 1

Short introduction

This collection of scripts is intended for the Lenovo ThinkPad X220 Tablet. You can still use them with the regular X220 machine, but only `thinkpad-rotate` will probably be useless for you then. I think that most scripts will also be handy for other ThinkPad models, I have not tested them though.

In short, this script fixes or improves the following:

1. Rotation of the internal screen and any Wacom touch and pen input devices using the bezel buttons or physical screen rotation
2. Get the microphone mute button to work.
3. Automatically use any external monitor, speakers and LAN connection when docking onto an UltraBase or similar.
4. Ability to disable touch pad or touch screen.

CHAPTER 2

Contents of the documentation

Guides

This section contains a few guides for thinkpad-scripts and related topics. If you're new to thinkpad-scripts, start with the *Getting Started* guide.

Getting Started

Installation

The easiest way to install thinkpad-scripts on Ubuntu, Arch Linux, Fedora and openSUSE is with your package manager, as described in *From Package*. If you are on another distribution, then you can build and install it manually using the instructions in *Build Manually*.

From Package

Ubuntu

On Ubuntu and its derivatives, you can install from Martin's PPA:

```
$ sudo -s
# add-apt-repository ppa:martin-ueding/stable
# apt-get update
# apt-get install thinkpad-scripts
```

Arch Linux

On Arch Linux, you can install the `thinkpad-scripts` package from the AUR.

Fedora

I, Martin Ueding, just (2015-01-24) tried out to package this for Fedora using the Open Build Service.

As far as I have understood and tested this, you have to download home:martinueding.repo into `/etc/yum.repos.d/`. Then you can use `yum` to install thinkpad-scripts.

```
$ sudo -s
# wget http://download.opensuse.org/repositories/home:/martinueding/Fedora_21/
↪home:martinueding.repo -O /etc/yum.repos.d/home:martinueding.repo
# yum install thinkpad-scripts
```

Warning: I only have a Fedora 21 VM right now, so I could only test it there. The virtual screen of the virtual machine could not be rotated, so I got an error message by `xrandr`. It seems like it would work on a real machine. I would be very grateful if somebody could test this and tell me about it.

- Repository overview page
- Repository download page

openSUSE

Same as *Fedora*, I tried to create an RPM package for openSUSE with the Open Build Service. There is a repository on my home project. Feedback is greatly appreciated.

Build Manually

First install all the dependencies, listed in *Dependencies*. Then, you can build and install with:

```
$ make
# make install
# ./setup.py install
```

To make the ACPI hooks take effect, you will need to restart `acpid` with the following on SysVinit/Upstart systems:

```
# service acpid restart
```

or on systemd systems:

```
# systemctl restart acpid
```

Packagers will also need to add the following line, run as root, to their post installation hook to update the udev hardware database with the information in `90-X2x0T-keyboard.hwdb`:

```
# udevadm hwdb --update
```

Alternatively, you can use `make full-install` which does that restarting for you. However, this does not work when `DESTDIR` is set to something! For a direct installation, use `make full-install`, for packaging, just use `make install`.

Dependencies

These dependencies refer to Debian and Arch Linux packages, but should have similar names in other distributions. `yum` in Fedora and `zypper` in openSUSE have a search for "provides". In openSUSE, you could use the `cnf` tool to find out the package.

Build

These programs are needed during the build process.

Needed Program	Debian package	Arch Linux package	Fedora package	openSUSE package
msgfmt	gettext	gettext	gettext	gettext
python3	python3	python	python3-devel	python3-devel
setuptools	python3-setuptools	python-setuptools	python3-setuptools	python3-setuptools
sphinx-build	python3-sphinx	python-sphinx	python3-sphinx	python3-Sphinx
xgettext	gettext	gettext	gettext	gettext

Run

These programs are required for the execution of the scripts.

Needed Program	Debian package	Arch Linux package	Fedora package	openSUSE package	Version
acpid	acpid	acpid	acpid	acpid	
amixer	alsa-utils	alsa-utils	alsa-utils	alsa-utils	
linux					>= 3.11.0-17[1]
python3	python3	python			
setuptools	python3-setuptools	python-setuptools	python3-setuptools	python3-setuptools	
udev	udev	systemd		systemd	>= 196
xinput	xinput	xorg-xinput	xinput	xinput	
xrandr	x11-xserver-utils	xorg-xrandr	xorg-x11-server-utils	xrandr	

Optional

These programs enhance the functionality of the scripts, but are not strictly required.

[1] The Ubuntu Kernel with version `3.11.0-17` has a patched `thinkpad-acpi` module which allows it to control the LED in the microphone mute button. Previous versions of thinkpad-scripts would flash the power LED to signal a muted microphone. This branch of thinkpad-scripts does not flash the power LED anymore, therefore requiring that version of the kernel.

openSUSE and other distributions are not patching the 3.?.0 kernel, but ship a 3.?.? kernel. So users of distributions other than Ubuntu (maybe even Debian) would have to check whether their kernel has the acpi patch.

Needed Program	Debian package	Arch Linux package	For
gsettings	libglib2.0-bin	glib2	subpixel anti-alias order with GNOME/XFCE
kvkbd	kvkbd	kvkbd	virtual keyboard
lsusb	usbutils	usbutils	docking detection with a USB device
nmcli	network-manager	networkmanager	changing wifi
pactl	pulseaudio-utils	libpulse	volume control when docking
xbacklight	xbacklight	xorg-xbacklight	adjusting brightness
xsetwacom	xserver-xorg-input-wacom	xf86-input-wacom	Wacom device rotation

Setup

thinkpad-scripts includes files that hook into various hardware events:

- a udeb hwdb file that allows proper operation of the bezel buttons on ThinkPad X220 and X230 Tablet computers
- udev rules to automatically run thinkpad-dock when docking and undocking
- ACPI hooks to automatically call thinkpad-rotate when the screen is rotated/unrotated

All of these files should be installed as part of the installation process. If acpid is not enabled by default on your computer (which is the case for Arch Linux), you need to enable and start it for the ACPI hooks to work. Additionally, after installing thinkpad-scripts, you may need to restart udev and acpid for the new rules and hooks to take effect.

Usage

After following the configuration instructions above, you generally will not need to call any of the scripts manually. However, in case you do, this is a synopsis of each command:

```
thinkpad-dock [on|off]
thinkpad-mutemic
thinkpad-rotate [direction]
thinkpad-touch [on|off]
thinkpad-touchpad
```

See the *Manual Pages* for more details.

Configuration

You can modify the default configuration for things such as the screen brightness to set when docking, the relative positions of displays, and the direction of screen rotation by placing configuration scripts in `$HOME/.config/thinkpad-scripts`. See the *Manual Pages* for more details.

You may need to modify some of the parameters depending on your hardware. See *Hardware-Specific Configuration* for more details.

You can also add scripts that will be called before/after docking or rotating the display. See the man pages for *thinkpad-dock* and *thinkpad-rotate* for more details.

Tips

thinkpad-scripts fixes the bezel buttons so that they work, but it does not bind anything to them by default. If you'd like, you can bind the `thinkpad-rotate` script (or any other program for that matter) to one of the bezel buttons using your desktop environment. For example, under GNOME, go to “Settings” → “Keyboard” → “Shortcuts” → “Custom Shortcuts” and add a new “shortcut”.

thinkpad-scripts includes a script, `thinkpad-touch`, to make it easy to toggle the touchscreen of the X220 Tablet on/off. If you want to disable your touch screen on startup, use your desktop environment to call `thinkpad-touch off` when starting.

Under KDE, it is convenient to place all of the scripts in a drawer so that you can access them quickly. See *Script Drawer For KDE Plasma Panel* for instructions to do this.

Find hardware events

The ThinkPad X220 Tablet has a hardware sensor that registers when the screen is turned around. To find the code of the event, use `acpi_listen`:

```
$ acpi_listen
video/tabletmode TBLT 0000008A 00000001
video/tabletmode TBLT 0000008A 00000000
```

I started the command, turned the screen around, flipped it onto the keyboard and back again.

This then goes into an ACPI hook file like so:

```
event=video/tabletmode TBLT 0000008A 0000000[01]
action=/usr/bin/thinkpad-rotate-hook %e
```

If you give us the output of `acpi_listen`, we can try to get the hardware event working for you. The hook in `tps/hooks.py` needs to be made aware of the hardware keys as well in order to decide which action to take.

Configuring Additional Hardware Keys

Introduction

Keys are identified on multiple levels in Linux: *scancode*, *keycode*, and *keysym*. A *scancode* is the sequence of bytes that a keyboard sends to a computer when a key is pressed. A *keycode* corresponds to a specific function. A *keysym* corresponds to a symbol typed by the keyboard and mappings of *scancodes* to *keysyms* depend on the keyboard layout. The progression of mapping goes *scancode* → *keycode* → *keysym*.[1] By default, some unusual hardware keys, such as those on the bezels of some tablets, are not mapped. This guide explains how to map *scancodes* to *keycodes* using the udev hwdb, which is part of udev versions 196 and later.

Note that thinkpad-scripts includes a udev hwdb file that fixes the bezel key mappings for Lenovo X220 and X230 Tablets. This guide is useful if you have different hardware. If you find something that works for your hardware, please feel free to submit a pull request to the GitHub project.

Directions

Determine the scancodes of the keys

Determine your Linux kernel version with:

```
$ uname -r
```

[1] https://wiki.archlinux.org/index.php/Extra_Keyboard_Keys

For kernels v2.6 and later, you need reboot with the kernel parameter `atkbd.softraw=0` in order for the following step to work[1]. Detailed instructions on how to add kernel parameters are provided in[2].

Switch to a virtual console with a text terminal with `Ctrl-Alt-F2`, login, then run as root[1]:

```
# showkey --scancodes
```

When you press a key, it should send the scancode to stdout. Sometimes, pressing and releasing a key have two different scancodes, and both scancodes will show up in the output of `showkey --scancodes`. For example, pressing and releasing the screen rotation bezel key on the X220 tablet gives the keycodes `0x67` and `0xe7`. Just choose the one that occurs when you initially press the key. For the keys that you want to map, write down which key corresponds to which scancode. Mappings for older ThinkPad tablets are available at[3].

You can switch back to your graphical environment with a key combination somewhere between `Ctrl-Alt-F1` and `Ctrl-Alt-F7` and then reboot to restore your default kernel parameters.

Determine which keycodes you want to map them to

Look in `/lib/udev/hwdb.d/60-keyboard.hwdb` for mappings of some other ThinkPad models[4]; these provide a guide for which key should correspond to which keycode. (The mappings are up to you, but it's a good idea to pick mappings similar to already established ones.)

A complete list of possible keycodes is in `/usr/include/linux/input.h`[4]. Look for the definitions with the names `KEY_<KEYCODE>`.

Determine the modalias string of your keyboard

Use this command to list all of the modalias entries on your system[5]:

```
$ find /sys -name modalias -print0 | xargs -0 cat | sort -u
```

Determine the modalias string that corresponds to your keyboard. The relevant one will probably start with `dmi`. One example from an X220 tablet is:

```
dmi:bvnLENOVO:bvr8DET46WW(1.16):bd05/18/
↪2011:svnLENOVO:pn42962WU:pvrThinkPadX220Tablet:rvnLENOVO:rn42962WU:rvrNotAvailable:cvnLENOVO:ct10:c
```

Write and install a udev hwdb configuration file

Create a hwdb file with the mappings that you want. Here is the file from `thinkpad-rotate`, named `90-X220T-keyboard.hwdb`:

```
# Thinkpad X220_Tablet
keyboard:dmi:bvn*:bvr*:bd*:svnLENOVO*:pn*:pvrThinkPadX220Tablet*
KEYBOARD_KEY_67=cyclewindows                          # bezel circular arrow
KEYBOARD_KEY_6c=scale                                 # rotate screen
```

- Anything after a # is a comment and is ignored.

[2] https://wiki.archlinux.org/index.php/Kernel_parameters
[3] http://www.thinkwiki.org/wiki/Tablet_Hardware_Buttons
[4] https://wiki.archlinux.org/index.php/Map_scancodes_to_keycodes
[5] http://people.skolelinux.org/pere/blog/Modalias_strings___a_practical_way_to_map__stuff__to_hardware.html

- The second line is a pattern that should match the modalias string of your keyboard. This example matches the modalias string in the previous section.
- The following lines are the mappings. Each line is in the form `KEYBOARD_KEY_<scancode>=<keycode>`. The `<scancode>` should be the value you obtained earlier without the `0x` at the front, and the `<keycode>` should be the keycode you selected earlier but in all lowercase.

Give the file an appropriate name, such as `90-X220T-keyboard.hwdb`, and place it in `/lib/udev/hwdb.d/`.

See[4] for more details.

Update the udev hwdb

Run the following to update the udev hwdb:

```
# udevadm hwdb --update
```

You may need to reboot for the changes to take effect.

Where to go from here

Now that you have properly mapped keycodes, you need to bind functionality to them. You can do this with your desktop environment's settings manager.

Note that some keycodes may not be mapped to keysyms, so your desktop environment may not recognize them. In this case, the easiest thing to do is to choose a different keycode for that key. (This is what I did for the X220 screen rotation button in `thinkpad-rotate`: based on other ThinkPad models in `/lib/udev/hwdb.d/60-keyboard.hwdb`, the `direction` keycode would be the better choice than `scale`. However, `direction` was not mapped in my desktop environment, so it was easier just to choose a different keycode that wasn't mapped to anything.) The alternative is to use a utility like xmodmap to perform the mapping of keycode to keysym[6].

You can find some interesting tricks at this (somewhat out-of-date) page:[7].

References

Hardware-Specific Configuration

Introduction

Most of the configuration parameters (described in the individual *man pages*) depend on your own personal preferences. However, some parameters depend on your specific model of computer. This page provides recommended values for various computers.

Unlisted Hardware

If your particular model of computer is not listed here, please submit a pull request with the correct configuration or create a report on the issue tracker with the name of your computer and output of:

```
$ xinput
$ xrandr
```

[6] https://wiki.archlinux.org/index.php/Xmodmap
[7] http://www.thinkwiki.org/wiki/How_to_get_special_keys_to_work

Recommended Configurations

Functional configurations for computers that users have reported so far are listed below.

Lenovo Thinkpad X200 & Lenovo Thinkpad X200 Tablet

Some of the hardware devices are named differently from the defaults. These are the necessary configuration parameters:

```
[input]
touchscreen_device = Serial Wacom Tablet touch

[touch]
regex = Serial Wacom Tablet.*id=(\d+)
```

Lenovo Thinkpad X220 & Lenovo Thinkpad X220 Tablet

The defaults should work fine.

Lenovo Thinkpad X230 & Lenovo Thinkpad X230 Tablet

The defaults should work fine.

Lenovo Thinkpad Yoga

Some of the hardware devices are named differently from the defaults. These are the necessary configuration parameters:

```
[input]
touchscreen_device = ELAN Touchscreen

[touch]
regex = (?:Wacom ISD|ELAN Touchscreen).*id=(\d+)
```

Script Drawer For KDE Plasma Panel

Since there are more scripts than buttons, I added a drawer with all the programs to my KDE Panel. It looks like this:

Fig. 2.1: The script collection in a folder right next to the system clock.

Add a new "folder view" to your panel and set the following options:

(1) Go into the first tab and (2) set the folder to `/usr/share/applications`.

(3) Then go to the "Filter" tab and (4) set `thinkpad-*.desktop` as the filter. That will only list scripts from this collection.

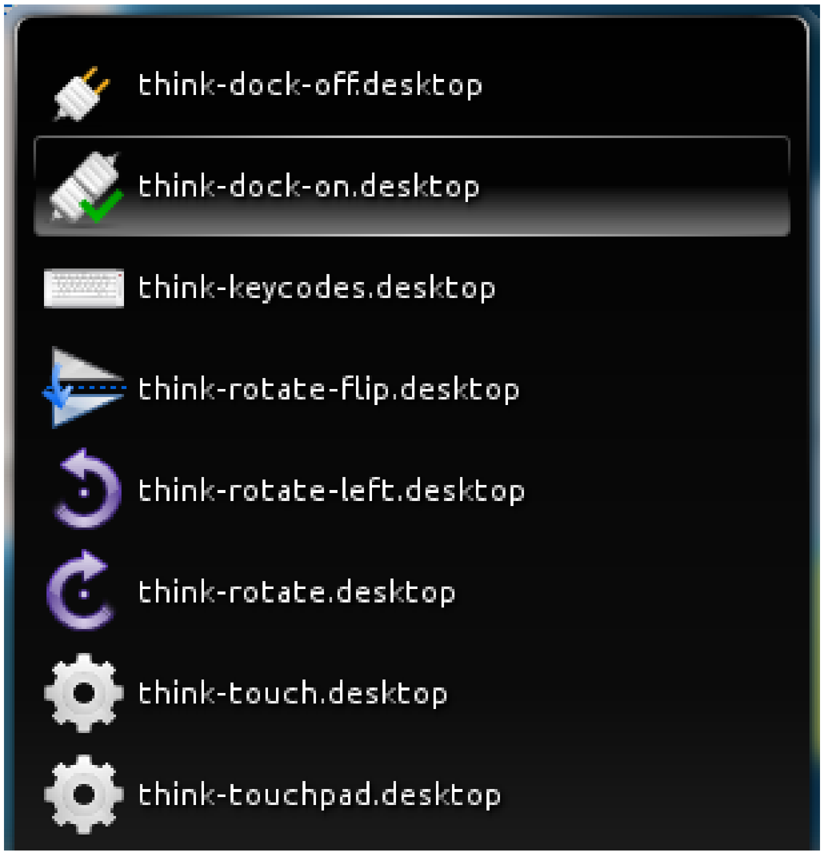
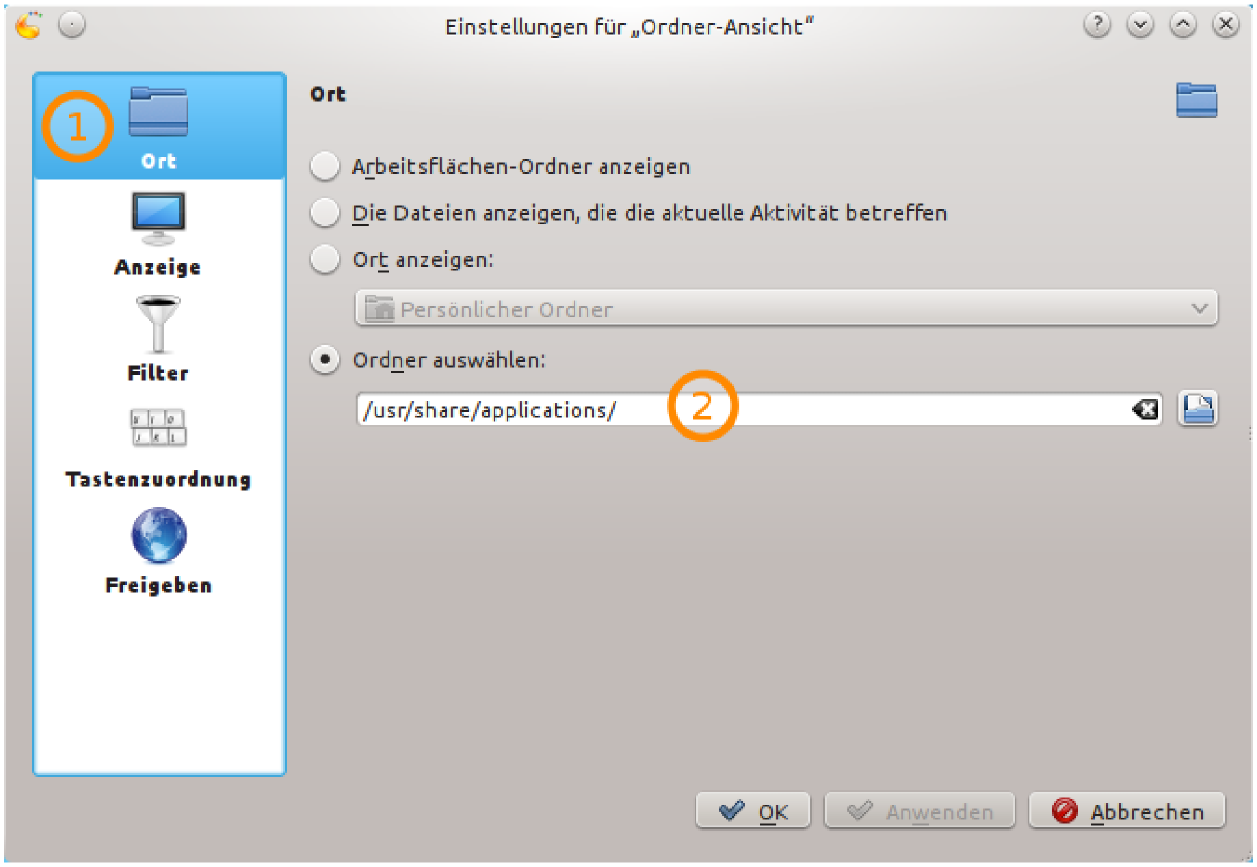


Fig. 2.2: KDE Plasma Panel drawer with all `thinkpad-` scripts.



Manual Pages

Once you have installed thinkpad-scripts, you can use `man <program-name>` to read the man page corresponding to `<program-name>`. The man pages are include here for your convenience:

thinkpad-config

Synopsis

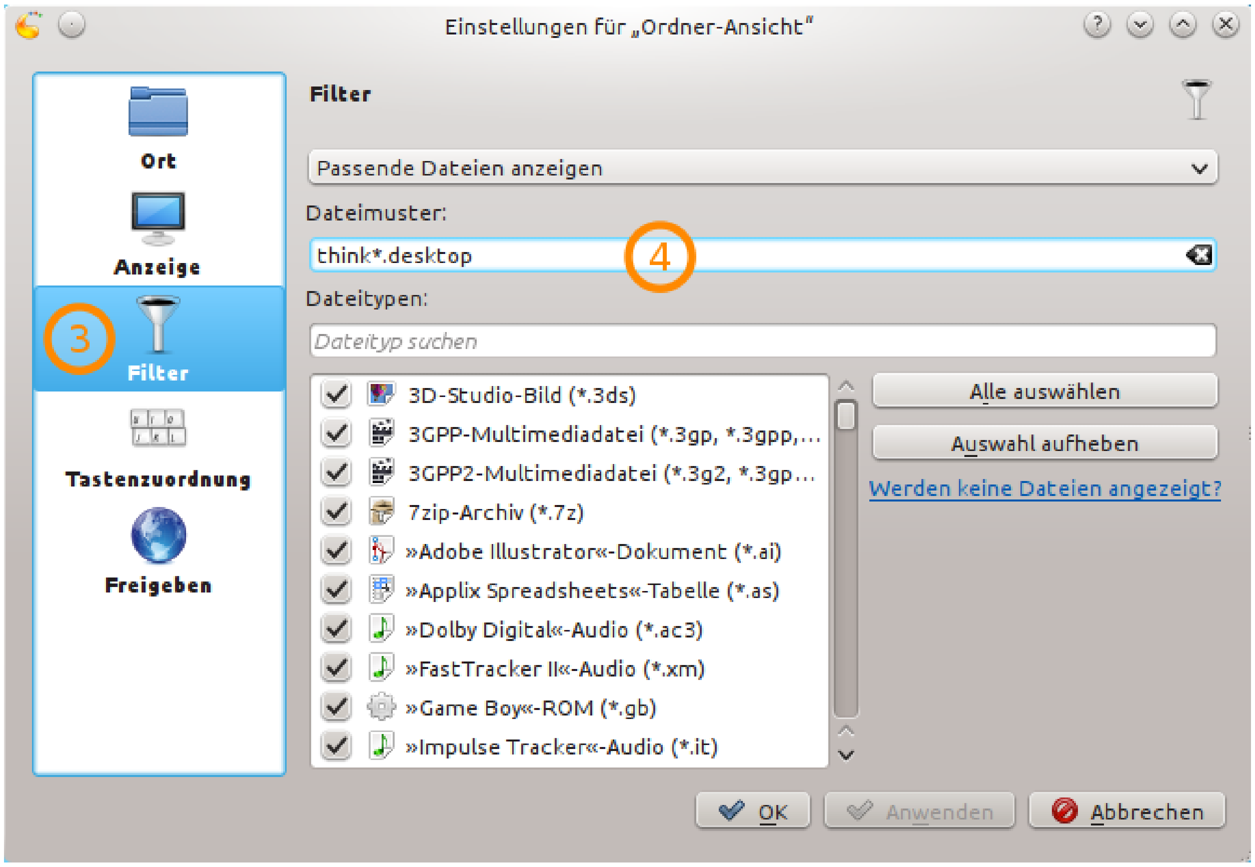
```
thinkpad-config
```

Description

The user configuration for thinkpad-scripts is stored in `~/.config/thinkpad-scripts/config.ini` in the INI format. There is a global configuration that thinkpad-scripts will use as a basis and apply your configuration over that, overriding default values. This program will show the config that will be used in the program.

Options

This program does not interpret any command line options.


Einstellungen für „Ordner-Ansicht"

Filter

Passende Dateien anzeigen

Dateimuster: think*.desktop

Dateitypen: Dateityp suchen

- ✓ 3D-Studio-Bild (*.3ds)
- ✓ 3GPP-Multimediadatei (*.3gp, *.3gpp,…
- ✓ 3GPP2-Multimediadatei (*.3g2, *.3gp…
- ✓ 7zip-Archiv (*.7z)
- ✓ »Adobe Illustrator«-Dokument (*.ai)
- ✓ »Applix Spreadsheets«-Tabelle (*.as)
- ✓ »Dolby Digital«-Audio (*.ac3)
- ✓ »FastTracker II«-Audio (*.xm)
- ✓ »Game Boy«-ROM (*.gb)
- ✓ »Impulse Tracker«-Audio (*.it)

Exit Status

0 Everything okay.

Epilogue

This file is part of thinkpad-scripts by Martin Ueding and Jim Turner.

We hope that this collection of scripts is useful to you. If you experience bugs, find the documentation lacking or have a new kind of hardware that we do not yet support, feel free to open an issue on GitHub or write an email to Martin Ueding.

See also:

- GitHub Repository
- project website.
- Hosted documentation (via Read the Docs)

thinkpad-dock

Synopsis

```
thinkpad-dock [on|off]
```

Description

This program sets the screen resolution correctly when putting the ThinkPad onto the docking station. It also sets the Wacom input devices to act on the internal screen only.

It deduces what to do automatically, if no option is given. If it is docked, it will perform the docking action. When you pressed the eject button on the docking station, it will un-dock.

There will be an udev rule installed that will automatically dock it when set onto the station and un-dock when you press the eject button. Technically, this rule calls the `thinkpad-dock-hook`.

What it does

When docking, the following things are done:

- Activating the external monitor.
- Setting the external monitor as primary monitor.
- Deactivate the wireless connection.
- Set the Wacom devices to the internal screen only.
- Set the brightness to a fixed value, currently 60%.
- Unmute the speakers and set the volume to 100%.

When undocking, the following things are done:

- Deactivating external monitor.
- Setting the internal monitor as primary monitor.
- Activating the wireless connection.
- Set the speakers to some medium volume, currently 50%.

Options

on|off If you have it sitting on the docking station and want it to dock, use `on`. Otherwise use `off` before you take the ThinkPad off the docking station.

You can omit this option and the script will guess what to do by checking whether a dock is docked in `/sys`.

Exit Status

0 Everything okay.

1 Some error.

Files

Config

You can create a config file in `$HOME/.config/thinkpad-scripts/config.ini`, which has standard INI format. The old config can be converted using the `thinkpad-scripts-config-migrate` script that was introduced in version 4.0.

A sample config would look like this:

```
[sound]
dock_loudness = 50%

[network]
disable_wifi = true

[screen]
relative_position = left-of
```

I will list all possible options in a moment. Since the INI format is hierarchical, I will denote the options with a dot. The first one would be `sound.dock_loudness` for example.

Those are the possible options:

`dock.lsusb_indicator_regex` Some docks might not have a docking indicator in the sysfs. In Issue 129 it has been discussed to use a particular USB device that is attached only at the dock to function as an indicator. If this option is set to a non-zero length string, it will be used as a regular expression. The output of `lsusb` is searched for that regular expression. If a match is found, the laptop is assumed to be on the docking station.

Example

The output of `lsusb` might contain lines like the following:

```
Bus 002 Device 003: ID 056a:00e6 Wacom Co., Ltd TPCE6
Bus 002 Device 002: ID 8087:0024 Intel Corp. Integrated Rate Matching Hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 04f2:b217 Chicony Electronics Co., Ltd Lenovo Integrated␣
↪Camera (0.3MP)
Bus 001 Device 006: ID 046d:c05a Logitech, Inc. M90/M100 Optical Mouse
Bus 001 Device 008: ID 273f:1007
Bus 001 Device 005: ID 0424:2514 Standard Microsystems Corp. USB 2.0 Hub
Bus 001 Device 004: ID 0424:2514 Standard Microsystems Corp. USB 2.0 Hub
Bus 001 Device 002: ID 8087:0024 Intel Corp. Integrated Rate Matching Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Some of these devices might be integrated in the docking station. One of the USB hubs is the one in my external screen. That does not help much because its ID is not unique. The unnamed device with ID `273f:1007` is only present on the docking station. Therefore I would set the configuration value to `273f:1007`.

At the office, I have a second docking station. There I have some other device, say ID `1234:1234`. Since this configuration option is a regular expression, I could specify the following: `273f:1007|1234:1234`. Then both devices can trigger the docking state.

`gui.kdialog` Please see the appropriate section in thinkpad-rotate(1), it has the same option. *Default:*.

`hooks.postdock` Full path to postdock hook. *Default: ~/.config/thinkpad-scripts/hooks/postdock*

`hooks.predock` Full path to predock hook. *Default: ~/.config/thinkpad-scripts/hooks/predock*

`logging.syslog` Whether to log everything to syslog. *Default: true*

`network.disable_wifi` Whether to set the wifi. *Default: true*.

`network.restart_connection` If this is set, the given network connection will be restarted on startup. I (Martin Ueding) have seen the issue where my default DHCP connection would not work right away. Restarting that connection helped. *Default: true*

`network.connection_name` If the connection should be restarted, you can specify which one in case there is more than one wired connection. The default case is to use the lexicographically first connection name in the list provided by `nmcli` that contains the case-insensitive string `'ethernet'`.

`screen.internal_regex` Regular expression to match the `xrandr` name for the internal monitor. *Default: LVDS-?1|eDP-?1*

`screen.primary` The `xrandr` name for the primary monitor when docked or an empty string to guess a reasonable monitor. *Default: (empty string)*.

`screen.secondary` The `xrandr` name for the secondary monitor when docked or an empty string to guess a reasonable monitor. *Default: (empty string)*.

`screen.set_brightness` Whether to change the brightness. *Default: true*.

`screen.brightness` Brightness to set to when docking. *Default: 60%*.

`screen.relative_position` Where to set the primary monitor relative to the secondary monitor when docking. Set it to `right-of` or `left-of` or anything else that `xrandr` supports with a `--*` argument. *Default: right-of*.

`screen.internal_docked_on` Whether to keep the internal screen on while docking. *Default: true*

`sound.unmute` Whether to change the volume. *Default: true*.

`sound.dock_loudness` Volume to set to when docking. *Default: 100%*.

`sound.undock_loudness` Volume to set to when undocking. *Default: 50%*.

`trigger.dock_triggers` Whitespace-delimited list of the enabled hardware triggers to execute docking/undocking. The available triggers are `udev1_on`, `udev1_off`, `acpi1_on`, `acpi1_off`, and `acpi2`. *Default:* `udev1_on udev1_off`

Hooks

There are hooks, called before and after the main script. It gets a single command line argument, `on` or `off`.

- `~/.config/thinkpad-scripts/hooks/predock`
- `~/.config/thinkpad-scripts/hooks/postdock`

You can change the path of those hooks in the configuration, see above.

Example

You can just call `thinkpad-dock` and it will do the right thing probably.

If you want, you can tell the script what to do: When you have it sitting on the docking station, call `thinkpad-dock on` to get the external screen going. When you are done, call `thinkpad-dock off` before you disconnect to get the internal screen back again.

Epilogue

This file is part of thinkpad-scripts by Martin Ueding and Jim Turner.

We hope that this collection of scripts is useful to you. If you experience bugs, find the documentation lacking or have a new kind of hardware that we do not yet support, feel free to open an issue on GitHub or write an email to Martin Ueding.

See also:

- GitHub Repository
- project website.
- Hosted documentation (via Read the Docs)

thinkpad-mutemic

Synopsis

```
thinkpad-mutemic
```

Description

This script will be called when you press the microphone mute button. It will mute the microphone and toggle the LED.

Epilogue

This file is part of thinkpad-scripts by Martin Ueding and Jim Turner.

We hope that this collection of scripts is useful to you. If you experience bugs, find the documentation lacking or have a new kind of hardware that we do not yet support, feel free to open an issue on GitHub or write an email to Martin Ueding.

See also:

- GitHub Repository
- project website.
- Hosted documentation (via Read the Docs)

thinkpad-rotate

Synopsis

```
thinkpad-rotate [direction]
```

Description

If you want to use your X220 Tablet as a tablet, you might want to rotate the screen. You can use this script for that and it will ensure that the pen and touch interface know about the rotated screen.

It will also disable the trackpoint (the xinput id is automatically queried) so that the back of the screen does not move your mouse if there is any force on the side of the screen.

Finally, it will start the virtual keyboard (`kvkbd` by default) when the screen is rotated and kill it when the screen is rotated back to normal.

If the screen is already rotated (say left) and you call `thinkpad-rotate left`, the screen will be reverted to the normal orientation. That way, you can use this script as a toggle.

A udev hook is installed as well that picks up the ACPI event when rotating the screen.

Options

`direction` The direction can be any of:

- ccw
- cw
- flip
- half
- left
- none
- normal
- right

Since the Wacom tools and `xrandr` have different names, this program accepts all of them, so that you do not have to learn yet another set of directions.

`-v` Enable verbose output. Can be supplied multiple times for even more verbosity.

`--force-direction` Do not try to be smart. Actually rotate in the direction given even it already is the case.

Exit Status

0 Everything went okay.

2 User specified a direction that is not known.

Environment

The script relies on `xrandr` to get the information, so this has to work.

Files

Config

You can create a config file in `$HOME/.config/thinkpad-scripts/config.ini`, which is a simple INI configuration file. The old config can be converted using the `thinkpad-scripts-config-migrate` script that was introduced in version 4.0. A sample config would look like this:

```
[rotate]
default_rotation = flip

[sound]
undock_loudness = 0%

[screen]
relative_position = left-of
```

You can set the following option:

`hooks.postrotate` Executable file to run after rotation. *Default: ~/.config/thinkpad-scripts/hooks/postrotate*

`hooks.prerotate` Executable file to run before rotation. *Default: ~/.config/thinkpad-scripts/hooks/prerotate*

rotate.default_rotation Default rotation if device is in normal rotation and no arguments are given. *Default: right*

rotate.subpixels Rotate subpixel orientation when rotating the screen. *Default: true*

rotate.subpixels_with_external Rotate the subpixel orientation if a second screen is attached. *Default: false*.

rotate.xrandr_bug_workaround On Ubuntu 15.04, XRandr has a bug which turns the screen black when rotating with no external screen attached.

This is problematic when the rotation is executed from a hardware event hook. Then the screen is physically laying on the keyboard and one cannot do anything. A workaround is to go to another terminal with [Ctrl][Alt][F1] and back to the graphical one with [Ctrl][Alt][F7].

As contributed by Cody Christensen, that can be automated with `chvt`. This way the hook will work in a useful way for users with that XRandr bug. However, this program needs superuser privileges. One can use `sudo` to allow oneself to call this program without a password entry. Add the following line in a file like `/etc/sudoers.d/chvt`:

```
myuser  ALL = NOPASSWD: /bin/chvt
```

Replace `myuser` with your username! Then check with `visudo -c` whether the syntax is fine.

thinkpad-scripts can figure out whether this line is implemented by querying `sudo -l` for a list of available commands with higher privileges. If you set this option to `true` and the line is configured, it will call `chvt 6; chvt 7` after the rotation and before the hook.

If `chvt` cannot be used, the hook will be disabled by enabling this option. That way you can manually rotate the contents of the display with `thinkpad-rotate`, press [Ctrl][Alt][F1] and [Ctrl][Alt][F7] and only then physically rotate the screen. The hook will not fire and rotate back.

Default: false.

screen.internal_regex Regular expression to match the `xrandr` name for the internal monitor. *Default: LVDS-?1|eDP-?1*

trigger.rotate_triggers Whitespace-delimited list of the enabled hardware triggers to execute rotation. The available triggers are `acpi1_normal`, `acpi1_rotated`, `acpi2_normal`, and `acpi2_rotated`. *Default:* `acpi1_normal acpi1_rotated acpi2_normal acpi2_rotated`

touch.regex Regular expression to match Wacom devices against. If your devices do not start with `Wacom ISD`, change this appropriately. *Default:* `Wacom ISD.*id=(\d+)`

unity.toggle_launcher The Unity Launcher on the left side is only shown if you excert pressure with the mouse. That means that you do not only have to put the mouse to the left edge of the screen, but push it beyond that edge. This is not possible to do with touchscreen or the pen, so you need to show the launcher by default.

With this option set to *true*, the hide mode will be toggled. That way, you have a hidden launcher on normal rotation, and a always-shown launcher with any rotation. *Default: false*

vkeyboard.program Command to start the virtual keyboard. Choices are (among others) `kvkbd` for KDE, `cellwriter`, `onboard`. *Default: kvkbd*

Hooks

You can add scripts to be called before and/or after rotation by placing them at the following paths. The `postrotate` hook gets the new rotation (`left`, `right`, `inverted`, or `normal`) as a command line argument.

The default paths are:

- `~/.config/thinkpad-scripts/hooks/prerotate`
- `~/.config/thinkpad-scripts/hooks/postrotate`

Example

To rotate the screen to the right (and later back again), use:

```
thinkpad-rotate
```

To specify the direction, you can use:

```
thinkpad-rotate left
thinkpad-rotate right
thinkpad-rotate inverted
thinkpad-rotate normal
```

Epilogue

This file is part of thinkpad-scripts by Martin Ueding and Jim Turner.

We hope that this collection of scripts is useful to you. If you experience bugs, find the documentation lacking or have a new kind of hardware that we do not yet support, feel free to open an issue on GitHub or write an email to Martin Ueding.

See also:

- GitHub Repository
- project website.
- Hosted documentation (via Read the Docs)

thinkpad-scripts-config-migration

Synopsis

```
thinkpad-scripts-config-migration
```

Description

The versions 3.x and before of thinkpad-scripts were Bash shell scripts. The configuration files also were shell scripts that the main script would `source` (execute within the main script) to set the configuration values.

With version 4.0, the complete project was rewritten in Python 3. The configuration format was changed to INI since the Python standard library ships the `configparser` module to handle those easily.

Users of the old version with configurations should be able to convert this into the new format with this tool. Since the 3.x configuration files could be programs really, there is no perfect way to parse them and turn them into INI files. This program tries to parse it and should work fine if your configuration file consists of simple variable assignments.

It will read the files `~/.config/thinkpad-scripts/rotate.sh` and `~/.config/thinkpad-scripts/dock.sh` and interpret them. All the errors will be shown as well as the configuration that is understood. You will be prompted whether to actually save the new configuration.

Options

This program does not take any options.

Epilogue

This file is part of thinkpad-scripts by Martin Ueding and Jim Turner.

We hope that this collection of scripts is useful to you. If you experience bugs, find the documentation lacking or have a new kind of hardware that we do not yet support, feel free to open an issue on GitHub or write an email to Martin Ueding.

See also:

- GitHub Repository
- project website.
- Hosted documentation (via Read the Docs)

thinkpad-touch

Synopsis

```
thinkpad-touch [on|off]
```

Description

This program enables/disables the touch screen of the ThinkPad tablet. If no option is given, it toggles the touch screen on/off.

Options

on|off If you want to enable the touch screen, use `on`. Otherwise use `off`.

If you omit this option, the script will toggle the touch screen on/off.

Exit Status

0 Everything okay.

1 Some error.

Config

In the configuration file, you can set the `xinput` name of the touch screen. The ThinkPad X220 Tablet has `Wacom ISDv4 E6 Finger touch` for instance:

```
[input]
touchscreen_device = Wacom ISDv4 E6 Finger touch
```

Examples

You can just call `thinkpad-touch` to toggle the touch screen; otherwise state on/off explicitly with `thinkpad-touch on` or `thinkpad-touch off`.

Epilogue

This file is part of thinkpad-scripts by Martin Ueding and Jim Turner.

We hope that this collection of scripts is useful to you. If you experience bugs, find the documentation lacking or have a new kind of hardware that we do not yet support, feel free to open an issue on GitHub or write an email to Martin Ueding.

See also:

- GitHub Repository
- project website.
- Hosted documentation (via Read the Docs)

thinkpad-touchpad

Synopsis

```
thinkpad-touchpad
```

Description

This scripts toggles the TrackPad. It is designed to work on ThinkPads, but it will probably work on almost all laptops.

Options

on|off If you want to enable the touchpad, use `on`. Otherwise use `off`.

If you omit this option, the script will toggle the touchpad on/off.

Config

In the configuration file, you can set the `xinput` name of the touchpad. The ThinkPad X220 Tablet has `SynPS/2 Synaptics TouchPad` for instance. The default configuration option is just `TouchPad` to be rather general. This is how you change it in the configuration:

```
[input]
touchpad_device = TouchPad
```

Epilogue

This file is part of thinkpad-scripts by Martin Ueding and Jim Turner.

We hope that this collection of scripts is useful to you. If you experience bugs, find the documentation lacking or have a new kind of hardware that we do not yet support, feel free to open an issue on GitHub or write an email to Martin Ueding.

See also:

- GitHub Repository
- project website.
- Hosted documentation (via Read the Docs)

thinkpad-trackpoint

Synopsis

```
thinkpad-trackpoint [on|off]
```

Description

This program enables/disables the TrackPoint of the ThinkPad. If no option is given, it toggles the TrackPoint on/off.

Options

on|off If you want to enable the touch screen, use `on`. Otherwise use `off`.

If you omit this option, the script will toggle the TrackPoint on/off.

Exit Status

0 Everything okay.

1 Some error.

Config

In the configuration file, you can set the `xinput` name of the TrackPoint. The ThinkPad X220 Tablet has `TPPS/2 IBM TrackPoint` for instance. The default configuration option is just `TrackPoint` to be rather general. This is how you change it in the configuration:

```
[input]
trackpoint_device = TrackPoint
```

Epilogue

This file is part of thinkpad-scripts by Martin Ueding and Jim Turner.

We hope that this collection of scripts is useful to you. If you experience bugs, find the documentation lacking or have a new kind of hardware that we do not yet support, feel free to open an issue on GitHub or write an email to Martin Ueding.

See also:

- GitHub Repository
- project website.
- Hosted documentation (via Read the Docs)

Changelog

v4.10.0 Released: 2017-03-19 18:40:09 +0100

- Add a configuration option to select hardware triggers
- Graphical user detection can now handle display `:1`

v4.9.1 Released: 2017-02-24 22:35:30 +0100

- Update documentation
- Improve determination of the username that is logged in using X.

v4.9.0 Released: 2017-02-19 18:03:31 +0100

- Add `lsusb` check for docking.

v4.8.1 Released: 2017-01-20 09:26:34 +0100

- Ignore an additional command line argument to the hooks (GH-127)
- Fix the missing import of `subprocess` (GH-128)
- Update the documentation (GH-126)
- Use `xsetwacom` for the rotation and screen mapping if the input device supports it (GH-124)

v4.8.0

- Ignore a failure by `xbacklight`. On Martin's laptop, the modesetting driver currently has no access to the brightness setting. Therefore the docking will always fail at the brightness step. This update converts the failure into a warning.
- Change the configuration option `screen.internal` to `screen.internal_regex` and give a sensible default value that matches the output name variant reported by the modesetting driver. This change should also help Yoga users.

v4.7.5 Released: 2017-01-14 20:40:37 +0100

- The program now does not crash in the case that the network connection could not be restarted. Instead, a warning is logged (GH-121).

v4.7.4 Released: 2016-07-13 10:35:07 +0200

- Update the name of the Wacom touch screen such that its name also works on Fedora 23 (GH-120).

v4.7.3 Released: 2016-04-15 13:14:40 +0200

- Fix location of `sphinx-build` on Fedora (GH-119). Contributed by Aruee.

v4.7.2 Released: 2016-04-14 21:07:41 +0200

- Ubuntu seems to ship with a version of XRandR which set the `Wacom Rotation` property of a few devices, but not all of them. As we have switched to the rotation matrix some versions ago, we and XRandR interfere with each other. Now we reset the rotation made by XRandR. Jim tested this on Ubuntu 15.10, so that should fix GH-117 and GH-112.
- Debug shell commands are pretty-printed using `shlex.quote`. That way, one can directly paste the log output into a shell and re-run a given command.

v4.7.1 Released: 2015-10-20 17:14:26 +0200

- Fix errors caused by hooks at boot time. Hooks are sometimes executed at boot when no user is logged in yet. This would cause error logs that are just annoying for the user as he/she is greeted with crash reports. This fixes GH-110 and GH-111.

v4.7.0 Released: 2015-10-16 10:10:16 +0200

- Add configuration option to disable the rotate and dock hook individually.
- In cases where `/dev/log` does not exist, it will use standard UDP to connect to the log.

v4.6.0 Released: 2015-10-15 20:30:31 +0200

- Add a `--force-direction` command line option for `thinkpad-rotate` such that this can be used in a script. When starting up the computer an autostart entry like

  ```
  thinkpad-rotate --force-direction normal
  ```

  could be very handy to normalize the setup.

v4.5.0 Released: 2015-10-15 16:50:57 +0200

- Add `chvt` workaround as suggested by Cody Christensen.

v4.4.2 Released: 2015-07-31 15:14:59 +0200

- Fix error in docking. I have broken it by assigning a temporary to a variable `output` which also happened to be the function argument. Sorry. I wish I had `const` in Python :-/.

v4.4.1 Released: 2015-07-29 08:23:28 +0200

- Check list of PulseAudio devices to get sound settings right. (Contributed by Jannis Stoppe, thank you!)
- Update regular expressions for `xinput`. We have been using the ones for `xsetwacom` until now. Since we have switched to `xinput` in version 4.2.3, this should fix bugs since then.

v4.4.0 Released: 2015-05-09 10:51:34 +0200

- Add a workaround for an XRandr bug that I have on my machine.

v4.3.0 Released: 2015-03-25 14:53:31 +0100

- Fix a bug that was introduced in 69ef6ea. This leads to premature exit and dump of a stacktrace. The screens got rotated, but the TrackPoint would not be disabled.
- Add an option to disable the internal screen on docking (GH-103).

v4.2.6 Released: 2015-03-15 22:53:34 +0100

- Update documentation for openSUSE package

v4.2.5 Released: 2015-03-15 19:28:52 +0100

- Remove icons from docking desktop files to get it to build on openSUSE Build Service

v4.2.4 Released: 2015-02-19 18:49:21 +0100

- Write transformation matrices to debug output.
- Small fixes in documentation: Remove dead navigation entry and use correct syntax highlighting for config snippet.

v4.2.3 Released: 2015-02-08

- Add documentation about Fedora package
- Add hardware specific documentation
- Replace `xsetwacom` with `xinput` in all cases and use transformation matrix (GH-91)
- Add a nice error message when a screen could not be found
- Remove `termcolor` as a dependency
- Always have at least one screen enabled
- Be more careful with `gsettings`, check whether the schema exists before writing to it

v4.2.2 Released: 2015-01-24

- Remove dependency on `termcolor` since that is not packaged for Python 3 in Ubuntu or Fedora. It was not needed heavily anyway, so I just got rid of it.
- Add manual page for `thinkpad-config`
- Add manual page for `thinkpad-trackpoint`
- Add manual page for `thinkpad-scripts-config-migration`
- Add a common epilogue for all manual pages
- Remove mailing list from README
- Replace hard coded strings with configuration options (GH-91)
- Toggle touch screen with `xinput` only (GH-91)
- Give a real error when rotation cannot be determined (GH-92)

v4.2.1 Released: 2015-01-20

- Fix errors in `.desktop` files
- Use built-in mocking for unit tests

v4.2.0 Released: 2015-01-15

- Log error when unsupported key is given to rotate hook.
- Fix `full-install` target in makefile.
- Add `test` target to makefile.
- Add support for multiple external monitors. See the manual page of `thinkpad-dock` for the details of the configuration options.

v4.1.5 Released: 2014-10-26

- Make selection of ethernet connection which is restarted predictable.
- Remove call to `nmcli con down` in the restarting of the network connection. This makes it compatible with nmcli 0.9.10. That closes GH-81, fixes GH-74 and closes GH-75,

v4.1.4 Released: 2014-10-25

- Fix GH-79 by catching the exceptions and logging warnings. Missing TrackPoint and TouchPad do not cause the program to abort now.

v4.1.3 Released: 2014-10-15

- Fix breakage of the rotation script when the subpixel order cannot be changed for some reason. An error is logged then.

v4.1.2 Released: 2014-10-05

- Fix hiding of Unity launcher (GitHub #72)
- Warn about `make install` (GitHub #76)

v4.1.1 Released: 2014-09-07

- Add `network.connection_name` configuration option.
- Add support for `nmcli` v0.9.10 command line interface.

v4.1 Released: 2014-07-12

- Add `tablet-normal` rotation. That will not rotate the screen but deactivate the trackpoint.
- Accept all rotation names again.

v4.0.8: Released: 2014-06-14

- Fix some errors in the manual pages

v4.0.7 Released: 2014-06-14

- Make triggering on hardware rotation slightly more robust against changes in the event that `acpid` gives.

v4.0.6 Released: 2014-06-02

- Toggle Wacom Touch property with `xsetwacom` as well as using `xinput`.

v4.0.5 Released: 2014-05-29

- Automatic determination of ethernet network connection
- `make install` does not restart any services. `make full-install` does that now.

v4.0.4 Released: 2014-05-29

- State Python termcolor dependency in the documentation
- Stop failing if `gsettings` is not installed
- Add subpixel rotation in Xfce
- Warn about missing screen when docking

v4.0.3 Released: 2014-05-28

- Replace unicode arrow because of Launchpad errors.

v4.0.2 Released: 2014-05-28

- Assert Python 3 everywhere. I suspect that the Launchpad Build System uses Python 2 for some reason. That causes some unicode errors.

v4.0.1 Released: 2014-05-28

- Fill in dependencies in the “Getting Started“ guide.
- Explicitly state the encoding in `getversion.py`.

v4.0 Released: 2014-05-27

- Complete rewrite in Python 3.
- INI style config. Run `thinkpad-scripts-config-migrate` to help you migrate your config.
- Remove the transitional scripts. If you have anything that still depends on having scripts starting with `think-`, **this will break!**
- v3.0.1 introduced more relative positions by putting the `-of` into your configuration variable. Old configurations that still had `left` or `right` still worked, since the script appended the `-of` for you. Those couple lines were removed, so **add a ‘‘-of‘‘ to your config, if you do not have already!**
- You can change the regular expression that matches the Wacom devices now in the config. That is `touch.regex` in the config.

v3.5.1 Released: 2014-02-22

- Small fixes in the manual pages

v3.5 Released: 2014-02-22

- **Added**: Set the option `toggle_unity_launcher` for *thinkpad-rotate* to un-hide the Unity launcher whenever the screen is rotated. This was previously an example hook in the guides, now it is part of the main suite of scripts.

v3.4 Released: 2014-02-21

- Rename all the scripts from `think-` to `thinkpad-` to match the new project name. To ease transition, there are transition scripts with the old names. **Be sure to adjust all your scripts and hooks accordingly!** The transition scripts will be dropped with version 4.0.
- Rename the configuration directory from `~/.config/think-rotate` to `~/.config/thinkpad-scripts`. There is an automatic upgrade script in place, so calling either `thinkpad-rotate` or `thinkpad-dock` will rename your configuration folder if it exists and there is no new one already existing.
- Put dates into the changelog, for all releases so far.

v3.3 Released: 2014-02-21

- Rename project to “thinkpad-scripts”
- Add subpixel anti-alias order change on rotation for Gnome

v3.2 Released: 2014-01-07

- Update copyright years in the documentation.
- Add a guard that prevents multiple execution of `think-dock` and `think-rotate`. For some reason, the `udev` hooks call the script twice, resulting in race conditions.

v3.1.2 Released: 2014-01-07

- Fix finding of external display. I tried to improve the syntax, but let the script fail whenever the number needed to be incremented.

v3.1.1 Released: 2014-01-05

- Clean all `*.pyc` files in makefile. This was causing errors with prisine tars and Debian packaging before.
- Add changelog to documentation

v3.1 Released: 2014-01-03

- Pass target orientation to postrotate hook
- Pass version number to Sphinx automatically from the changelog

v3.0.2 Released: 2013-12-19

- Manual pages with Sphinx

v3.0.1 Released: 2013-12-10

- Allow more relative positions by putting the `-of` into the value of the `relative_position` variable

v3.0 Released: 2013-12-01

- Settings of the keycodes is now done via a `.hwdb` file for `udev`. This requires `udev` to be of version 196 or greater. Therefore, it is marked as a major release, since it breaks Ubuntu 13.04 and earlier.

v2.11 Released: 2013-12-01

- Add some guides: "Additional Keys" and "KDE Script Drawer"
- Fix recursive make, pass `-j` down to child processes

v2.10.2 Released: 2013-10-30

- Actually return from function.

v2.10.1 Released: 2013-10-28

- Do not fail if `qdbus` does not work (like on vanilla Kubuntu 13.10)

v2.10 Released: 2013-10-28

- Print missing programs
- Do not fail if `qdbus` is missing

v2.9 Released: 2013-10-07

- **Added**: ACPI hook to call `think-rotate` (Jim Turner)
- **Added**: Support for systemd network inferface names (Jim Turner)
- **Removed**: `think-resume` (Jim Turner)
- Use syslog in `think-dock`
- Update documentation
- State all dependencies (Debian package names)
- Change indentation to four spaces instead of a single tab

v2.8.1 Released: 2013-09-30

- More logging to syslog
- Disable `kdialog` for ACPI hooks since that does now work well

v2.8 Released: 2013-09-24

- Translate to German

v2.7.1 Released: 2013-08-08

- Close KDialog progress bar when the script fails (via `trap`)

v2.7 Released: 2013-07-31

- **Added**: Hooks
- **Added**: `on|off` for the `think-touchpad` script

v2.6 Released: 2013-06-26

- Support for `kdialog` status.

v2.5.2 Released: 2013-05-10

- Update the ACPI hooks to find other docks as well

v2.5.1 Released: 2013-05-06

- Find other docks as well

v2.5 Released: 2013-02-03

- Get microphone mute button to work

v2.4.1 Released: 2012-12-29

- Actually install makefiles
- Implement required actions in `init.d` script to that Debian lintian does not complain

v2.4 Released: 2012-12-29

- Fix bezel keyboard codes, so that they are usable. (Jim Turner)
- Add script to toggle touch screen. (Jim Turner)
- Organize code in subdirectories, using recursive make.

v2.3.1 Released: 2012-11-02

- Map Wacom devices to the output when rotating in any case. Thanks to Jim Turner!

v2.3 Released: 2012-10-25

- Add support for other virtual keyboards. Thanks to Jim Turner!
- Use shorter redirection (`&>` instead of `2>&!`).

v2.2.1 Released: 2012-10-22

- Fix spelling typo in `relative_position`. Thanks to Jim Turner!

v2.2 Released: 2012-10-15

- Background most tasks so that they run in parallel. This should speed up docking.

v2.1 Released: 2012-10-06

- Only set Wacom screen devices. That way, any attached Wacom graphics tablet is not affected by the docking.

v2.0 Released: 2012-08-31

- Use the kernel to determine what the docking status is.
- Add `udev` rules to perform the docking action.

v1.5 Released: 2012-08-31

- Desktop files for think-dock.

v1.4.5 Released: 2012-07-21

- Revert too intelligent behavior.

v1.4.4 Released: 2012-07-21

- Even if the user calls `think-dock on`, do not dock if there is no external monitor attached. This might be the case when the `think-dock on` is called automatically without any prior checks. If the script would dock either way, it might disable wireless (although that is only done when `eth0` is connected) and set the volume to a wrong setting.

v1.4.3 Released: 2012-07-20

- Disable the wireless connection on docking.

v1.4.2 Released: 2012-07-20

- Fix commands in `.desktop` files.

v1.4.1 Released: 2012-07-20

- Install `.desktop` files.

v1.4 Released: 2012-07-20

- Query the state of the whole system automatically and determine the right action. You can still specify `on` or `off`, if you want to.

v1.3 Released: 2012-07-16

- Optional config file for `think-dock`.

v1.2.2 Released: 2012-07-16

- Fix flip direction.

v1.2.1 Released: 2012-07-16

- Disable wireless only when eth0 connected.
- Document options.

v1.2 Released: 2012-07-15

- Change display brightness on docking.

v1.1 Released: 2012-07-15

- Check whether programs are there before using them.
- Create directories on `make install`.
- Disable wifi when going onto the docking station.
- Enable sound on docking.
- Lower the volume after docking.
- Query Wacom devices automatically.

v1.0 Released: 2012-07-13

This is the first release with a version number. It contains a couple fixes and improvements compared to previous (before 2012-07-13) versions of these scripts.

- Accept other names for the rotation.
- Disable the trackpad as well.
- Start and stop the virtual keyboard.
- Try to go back automatically, if a rotation is already set.
- Use `--rotation` instead of `-o`. This will only rotate the internal screen and not any attached screens as well.

Way before 2012-07-13, those are significant changes in the history:

- Add desktop files.
- Also set Wacom hardware correctly.
- Determine resolution automatically.
- Disable trackpoint when switching.
- Dynamically find external display.
- Limit Wacom devices to internal screen.
- Set external monitor as primary.

Legal and License

I took the script, that served as a basis for *thinkpad-rotate* from a forums entry where the original author said:

> "Put this in a file blah.sh anywhere, and do whatever you want with it!"

The changes that I made to that script are licensed under the GPLv2.

All other scripts are just licensed under GPLv2.

The "ThinkPad" name

ThinkPad® is a trademark of Lenovo®. This project is not affiliated with, sponsored by, or endorsed by Lenovo. Our use of the term "ThinkPad" is purely descriptive since this collection of scripts is only applicable to said type of computers.

Developer documentation

This part of the documentation is meant for developers. If you just want to use thinkpad-scripts, you do not need to read this.

API

tps

Main module for thinkpad-scripts.

class `tps.Direction`(*xrandr*, *xsetwacom*, *subpixel*, *physically_closed*, *rot_mat*)

Holds the direction names of different tools.

`xrandr` and `xsetwacom` use different names for the rotations. To avoid proliferation of various names, this class holds the differing names. The module provides constants which have to be used within `tps`.

`physically_closed`
Alias for field number 3

`rot_mat`
Alias for field number 4

`subpixel`
Alias for field number 2

xrandr
Alias for field number 0

xsetwacom
Alias for field number 1

tps.**INVERTED = Direction(xrandr='inverted', xsetwacom='half', subpixel='bgr', physically_closed=True, rot_mat=[-1, 0, 1, 0**
Inverted

tps.**LEFT = Direction(xrandr='left', xsetwacom='ccw', subpixel='vrgb', physically_closed=True, rot_mat=[0, -1, 1, 1, 0, 0, 0, 0**
Left

tps.**NORMAL = Direction(xrandr='normal', xsetwacom='none', subpixel='rgb', physically_closed=False, rot_mat=[1, 0, 0, 0, 1,**
Normal

tps.**RIGHT = Direction(xrandr='right', xsetwacom='cw', subpixel='vbgr', physically_closed=True, rot_mat=[0, 1, 0, -1, 0, 1, 0,**
Right

tps.**TABLET_NORMAL = Direction(xrandr='normal', xsetwacom='none', subpixel='rgb', physically_closed=True, rot_mat=[1,**
Tablet normal

exception tps.**UnknownDirectionException**
Unknown direction given at the command line.

tps.**assert_python3**()
Asserts that this is running with Python 3

tps.**has_program**(*command*)
Checks whether given program is installed on this computer.

Parameters **command** (*str*) – Name of command

Returns Whether program is installed

Return type bool

tps.**print_command_decorate**(*function*)
Decorates a func from the subprocess module to log the *command* parameter.

Note that the wrapper adds an additional *local_logger* parameter following the *command* parameter that is used for the logging. All other parameters are passed to the wrapped function.

Parameters **function** – Function to wrap

Returns Decorated function

tps.**static_vars**(***kwargs*)
Attach static variables to a function.

Python does not have static variables. There is a workaround since all Python functions are objects really. Therefore one can attach attributes to it. This decorator conveniently does that.

Taken from a *Stack Overflow answer* by *Claudiu* and *ony*.

tps.**translate_direction**(*direction*)

Parameters **direction** (*str*) – Direction string

Returns Direction object

Return type *tps.Direction*

Raises ***tps.UnknownDirectionException*** –

tps.config

Config module.

Takes care of the INI style config file for global and user configuration.

`tps.config.`**`CONFIGFILE`** **= '/home/docs/.config/thinkpad-scripts/config.ini'**

Path of global config file

exception `tps.config.`**`ShellParseException`**

Bash code could not be parsed.

The parser here is very limited, it can only detect variable assignments. If something more complicated is found on a given line, this exception is raised.

`tps.config.`**`get_config`**`()`

Loads the config from the config files.

The global config file is read first, then the user config file is read. That way, options can be overwritten in the user config file.

Returns Config

Return type configparser.ConfigParser

`tps.config.`**`interpret_shell_line`**`(`*line*, *config*`)`

Interprets a single Bash line to parse for variable assignments.

The given line is searched for a config option was allowed in the 3.x series. It will use the `shlex` module to parse the value of the variable assignment. If it could be parsed correctly, it will add the value to the config. The keys are translated into the new config sections.

Parameters

- **`line`** (*`str`*) – Line to check
- **`config`** (*`configparser.ConfigParser`*) – Config to store parsed variables

Raise tps.config.ShellParseException

Returns None

`tps.config.`**`main`**`()`

Command line entry point.

Returns None

`tps.config.`**`migrate_shell_config`**`()`

Migrates the shell config in an interactive way.

The old shell based config will be imported and transfered into a configparser based one. Then that will be printed out. The user can accept it and the config will be saved.

`tps.config.`**`print_config`**`(`*config*`)`

Pretty prints config with colors.

Parameters **`config`** (*`configparser.ConfigParser`*) – Config to print

Returns None

`tps.config.`**`set_up_logging`**`(`*verbosity*`)`

Sets up the logging to console and syslog.

This is taken from the Python Docs – Logging Cookbook.

The `address` parameter for the syslog is taken from an answer from dr jimbob.

tps.dock

Logic related to the UltraBase® docks.

`tps.dock.`**`dock`**`(`*on*, *config*`)`
Performs the makroscopic docking action.

Parameters

- **on** (*bool*) – Desired state
- **config** (*configparser.ConfigParser*) – Global config

Returns None

`tps.dock.`**`is_docked`**`(`*config*`)`
Determines whether the laptop is on a docking station.

This checks for `/sys/devices/platform/dock.*/docked`. In issue 129 it became apparent that this is not a sufficient solution. Therefore a configuration option allows to alternatively check for USB devices that are present.

Returns True if laptop is docked

Return type bool

`tps.dock.`**`main`**`()`
Command line entry point.

Returns None

`tps.dock.`**`select_docking_screens`**`(`*internal*, *primary=''*, *secondary=''*`)`
Selects the primary, secondary, and remaining screens when docking.

If *primary* or *secondary* is not the name of a connected screen, then select an appropriate screen from the connected screens. External screens are prioritized over the *internal* screen, and *primary* is prioritized over *secondary*. Warn the user if *primary* or *secondary* is a non-empty string and the screen is not connected.

If no external screens are connected, then set *primary* to the *internal* screen, and set *secondary* to *None*.

Parameters

- **internal** (*str*) – Name of the internal screen
- **primary** (*str*) – Name of primary screen, or an empty string
- **secondary** (*str*) – Name of secondary screen, or an empty string

Returns (*primary*, *secondary*, [*other1*, ...])

Return type tuple

For example, when only `LVDS1` is connected:

```
>>> select_docking_screens('LVDS1', '', '')
('LVDS1', None, [])
```

When `LVDS1` and `VGA1` are connected:

```
>>> select_docking_screens('LVDS1', '', '')
('VGA1', 'LVDS1', [])
>>> select_docking_screens('LVDS1', 'LVDS1', '')
('LVDS1', 'VGA1', [])
```

When `LVDS1`, `VGA1`, and `HDMI1` are connected:

```
>>> select_docking_screens('LVDS1', '', '')
('HDMI1', 'VGA1', ['LVDS1'])
>>> select_docking_screens('LVDS1', 'VGA1', '')
('VGA1', 'HDMI1', ['LVDS1'])
>>> select_docking_screens('LVDS1', '', 'LVDS1')
('HDMI1', 'LVDS1', ['VGA1'])
```

Note that the default order of `VGA1` versus `HDMI1` depends on the output of `xrandr`. See `tps.testsuite.test_dock.SelectDockingScreensTestCase` for more examples.

tps.hooks

Functions that execute the appropriate hooks.

`tps.hooks.`**`get_graphicsl_user`**`()`

`tps.hooks.`**`main_dock_hook`**`()`

Entry point for `thinkpad-dock-hook`.

It interprets the key values from the caller and start up another interpreter with the actual `thinkpad-dock` script.

`tps.hooks.`**`main_rotate_hook`**`()`

Entry point for `thinkpad-rotate-hook`.

It interprets the key values from the caller and start up another interpreter with the actual `thinkpad-rotate` script.

`tps.hooks.`**`parse_graphical_user`**`(`*lines*`)`

Determine the graphical user from the output of `who -u`.

`tps.hooks.`**`postdock`**`(`*state*, *config*`)`

Executes postdock hook if it exists.

Parameters

- **`state`** (*`bool`*) – Whether new state is on
- **`config`** (*`configparser.ConfigParser`*) – Global config

Returns None

`tps.hooks.`**`postrotate`**`(`*direction*, *config*`)`

Executes postrotate hook if it exists.

Parameters

- **`direction`** (`tps.Direction`) – Desired direction
- **`config`** (*`configparser.ConfigParser`*) – Global config

Returns None

`tps.hooks.`**`predock`**`(`*state*, *config*`)`

Executes predock hook if it exists.

Parameters

- **`state`** (*`bool`*) – Whether new state is on
- **`config`** (*`configparser.ConfigParser`*) – Global config

Returns None

`tps.hooks.`**`prerotate`**(*direction*, *config*)

Executes prerotate hook if it exists.

Parameters

- **`direction`** (`tps.Direction`) – Desired direction
- **`config`** (*`configparser.ConfigParser`*) – Global config

Returns None

tps.input

Logic related to input devices.

exception `tps.input.`**`InputDeviceNotFoundException`**

`xinput` device could not be found.

`tps.input.`**`generate_xinput_coordinate_transformation_matrix`**(*output*, *orientation*)

Generates the coordinate transformation matrix that is needed for xinput to confine the input to one screen and rotate it properly.

0.415703, 0.000000, 0.584297, 0.000000, -0.711111, 0.711111, 0.000000, 0.000000, 1.000000

`tps.input.`**`get_wacom_device_ids`**()

Gets the IDs of the built-in Wacom touch devices.

This calls `xinput` to get the list and parses that with a regular expression. Only device names starting with `Wacom ISD` (default regex) are taken into account. If you have an external device, this will not be picked up.

Return type list

`tps.input.`**`get_xinput_id`**(*name*)

Gets the `xinput` ID for given device.

The first parts of the name may be omitted. To get “TPPS/2 IBM TrackPoint”, it is sufficient to use “TrackPoint”.

Raises ***`InputDeviceNotFoundException`*** – Device not found in `xinput` output

Return type int

`tps.input.`**`get_xinput_state`**(*device*)

Gets the device state.

Parameters **`device`** (*`int`*) – `xinput` ID of devicwe

Returns Whether device is enabled

Return type bool

`tps.input.`**`has_device_property`**(*device*, *property_*)

Checks whether a given device supports a property.

`tps.input.`**`has_xinput_prop`**(*device*, *prop*)

Checks whether the device has the given xinput propery.

`tps.input.`**`map_rotate_all_input_devices`**(*output*, *orientation*)

Maps all Wacom® devices.

`tps.input.`**`map_rotate_input_device`**(*device*, *matrix*)

Rotates an input device.

`tps.input.`**`map_rotate_wacom_device`**(*device*, *output*, *direction*)

`tps.input.`**`set_wacom_touch`**(*device_id*, *state*)
Changes the Wacom Touch property of the given device.

`tps.input.`**`set_xinput_state`**(*device*, *state*)
Sets the device state.

Parameters

- **`device`** (*`int`*) – `xinput` ID of devicwe
- **`state`** (*`bool`*) – Whether device should be enabled

`tps.input.`**`state_change_ui`**(*config_name*)
Change the state of the given device depending on command line options.

It parses the command line options. If no state is given there, it will be the opposite of the current state.

Parameters **`set_touch`** (*`bool`*) – Whether to also toggle the `Touch` property on this device.

Returns None

`tps.input.`**`wacom_rotate_reset`**(*device*)
Resets the "Wacom Rotation" property of devices.

In GH-117 we noticed that in Ubuntu the `xrandr` rotation command will also rotate some input devices. This is probably meant in a good way but interferes with our rotation here. Therefore we reset the "Wacom Rotation" after setting the transformation matrix.

tps.rotate

`tps.rotate.`**`can_use_chvt`**()
Checks whether `chvt` can be called with `sudo` without a password.

The `sudo` command has the `-n` option which will just make the command fail when the user does not have the appropriate permissions. The problem with `chvt` is that it does not have any intelligent command line argument parsing. If will return code 1 if no argument is given, the same code that `sudo` gives when no permission is available. Therefore I chose to use `` sudo -l` to get the whole list and see whether the full path to ``chvt `` is in there. This might break on Fedora where the `usr`-merge has been done now.

The following line is needed in a file like `/etc/sudoers.d/chvt`:

```
myuser  ALL = NOPASSWD: /bin/chvt
```

You have to replace `myuser` which your username. Giving too broad permissions to every other user account is probably not a good idea.

Return type bool

`tps.rotate.`**`has_external_screens`**(*config*)
Checks whether any external screens are attached.

`tps.rotate.`**`main`**()
Entry point for `thinkpad-rotate`.

`tps.rotate.`**`needs_xrandr_bug_workaround`**(*config*)
Determines whether xrandr bug needs to be worked around.

XRandr has a bug in Ubuntu, maybe even in other distributions. In Ubuntu 15.04 a workaround is to change the virtual terminal to a different one and back to the seventh, the graphical one. This can be automated using the `chvt` command which requires superuser privileges. An entry in the sudo file can let the normal user execute this program.

`tps.rotate.`**`new_rotation`**(*current*, *desired_str*, *config*, *force=False*)
Determines the new rotation based on desired and current one.

Parameters **`force`** (*`bool`*) – If set the function does not try to be too clever but

just uses the rotation given. If no rotation is given in `desired_str`, it still uses the default from the configuration.

`tps.rotate.`**`rotate_to`**(*direction*, *config*)
Performs all steps needed for a screen rotation.

`tps.rotate.`**`toggle_virtual_terminal`**()

`tps.rotate.`**`xrandr_bug_fail_early`**(*config*)
Quits the program if xrandr bug cannot be coped with.

tps.screen

Screen related logic.

exception `tps.screen.`**`ScreenNotFoundException`**
`xrandr` device could not be found.

`tps.screen.`**`disable`**(*screen*)
Disables the given screen using `xrandr`.

Parameters **`screen`** (*`str`*) – Name of the output to disable

Returns None

`tps.screen.`**`enable`**(*screen*, *primary=False*, *position=None*)
Enables given screen using `xrandr`.

Parameters

- **`screen`** (*`str`*) – Name of the output to enable
- **`primary`** (*`bool`*) – Set output as primary
- **`position`** (*`tuple`*) – Tuple with (0) relative position and (1) other output. This could be `('right-of', 'LVDS1')`.

Returns None

`tps.screen.`**`filter_outputs`**(*outputs*, *regex*)

`tps.screen.`**`get_available_screens`**(*output*)

`tps.screen.`**`get_externals`**(*internal*)
Gets the external screens.

You have to specify the internal screen to exclude that from the listing.

;param str internal: Name of the internal screen :returns: List of external screen names :rtype: str

`tps.screen.`**`get_internal`**(*config*, *cache=True*)
Matches the regular expression in the config and retrieves the actual name of the internal screen.

The names of the outputs that XRandR reports may be `LVDS1` or `LVDS-1`. The former happens with the Intel driver, the latter with the generic kernel modesetting driver. We do not know what the system will provide, therefore it was decided in GH-125 to use a regular expression in the configuration file. This also gives out-of-the-box support for Yoga users where the internal screen is called `eDP1` or `eDP-1`.

Parameters

- **config** – Configuration parser instance
- **cache** (*bool*) – Compute the value again even if it is cached

`tps.screen.`**`get_resolution_and_shift`**(*output*)

Retrieves the total resolution of the virtual screen and the position of the given output within that.

The X server seems to generate a huge screen which is then displayed by the physical displays. `xrandr` gives the size of that (virtual) screen as well as the positions of each display in that.

For example, I currently have the 12.5" 1366×768 ThinkPad X220 display on the right of a 23" 1920×1080 pixel display. `xrandr` tells me the following:

```
Screen 0: ... current 3286 x 1080 ...
LVDS1 ... 1366x768+1920+0
DP2 ... 1920x1080+0+0
```

This only shows the interesting parts. The size of the (virtual) screen is 3286×1080 and the position of the internal screen is 1366×768+1920+0. This allows to compute the transformation matrix for this.

`tps.screen.`**`get_rotation`**(*screen*)

Gets the current rotation of the given screen.

Parameters **screen** (*str*) – Find rotation of given output

Returns Current direction

Return type *tps.Direction*

`tps.screen.`**`rotate`**(*screen*, *direction*)

Rotates the screen into the direction.

Parameters

- **screen** (*str*) – Name of the output to rotate
- **direction** (`tps.Direction`) – New direction

Returns None

`tps.screen.`**`set_brightness`**(*brightness*)

Sets the brightness with `xbacklight`.

Parameters **brightness** (*str*) – Percent value of brightness, e. g. `60%`

Returns None

`tps.screen.`**`set_subpixel_order`**(*direction*)

Sets the text subpixel anti-alias order.

Parameters **direction** (`tps.Direction`) – New direction

Returns None

tps.unity

Logic for Ubuntu Unity.

`tps.unity.`**`set_launcher`**(*autohide*)

Sets the autohide property of the Unity launcher.

In the back, this uses `dconf`. If that is not installed, this just fails with a warning.

Parameters **autohide** (*bool*) – True if autohide is desired

tps.vkeyboard

Logic for virtual keyboard

`tps.vkeyboard.`**`toggle`**(*program*, *state*)

Toggles the running state of the given progam.

If state is true, the program will be spawned.

Parameters

- **`program`** (*`str`*) – Name of the program
- **`state`** (*`bool`*) – Desired state

Returns None

Environments

The environment differs between the normal user and root. This might be the cause for some bugs.

Difference user and root

See the following diff between the user and root environment, made with `$ env` and `# env`:

```
--- env-user.txt        2014-01-07 08:10:57.297768327 +0100
+++ env-root.txt        2014-01-07 08:10:57.297768327 +0100
@@ -1,76 +1,16 @@
-BROWSER=/usr/bin/firefox
-COLORFGBG=15;0
-DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-TK74xcpHuB
-DEBEMAIL=dev@martin-ueding.de
-DEBFULLNAME=Martin Ueding
-DEBIAN_PACKAGING_DIR=/home/mu/Packaging_Debian
-DEFAULTS_PATH=/usr/share/gconf/kde-plasma.default.path
-DESKTOP_SESSION=kde-plasma
 DISPLAY=:0
-EDITOR=/usr/bin/vim
-GDM_LANG=en
-GDMSESSION=kde-plasma
-GNOME_KEYRING_CONTROL=/run/user/1000/keyring-35U0Ew
-GNOME_KEYRING_PID=1784
-GPG_AGENT_INFO=/tmp/gpg-BQD8Wt/S.gpg-agent:1864:1
-GS_LIB=/home/mu/.fonts
-GTK2_RC_FILES=/etc/gtk-2.0/gtkrc:/home/mu/.gtkrc-2.0:/home/mu/.kde/share/config/
↪gtkrc-2.0
-GTK_RC_FILES=/etc/gtk/gtkrc:/home/mu/.gtkrc:/home/mu/.kde/share/config/gtkrc
 HOME=/home/mu
-IM_CONFIG_PHASE=1
-INSTANCE=
-JOB=dbus
-KDE_FULL_SESSION=true
-KDE_MULTIHEAD=false
-KDE_SESSION_UID=1000
-KDE_SESSION_VERSION=4
-KONSOLE_DBUS_SERVICE=:1.72
-KONSOLE_DBUS_SESSION=/Sessions/2
-KONSOLE_DBUS_WINDOW=/Windows/1
```

```
-KONSOLE_PROFILE_NAME=Shell
 LANG=de_DE.UTF-8
 LANGUAGE=en
-LESSCLOSE=/usr/bin/lesspipe %s %s
-LESS=-FRSXx8
-LESSOPEN=| /usr/bin/lesspipe %s
-LOGNAME=mu
-MAKEFLAGS=-j 4
-MANDATORY_PATH=/usr/share/gconf/kde-plasma.mandatory.path
-MATHEMATICA_HOME=/usr/local/Wolfram/Mathematica/9.0
-PAGER=less -FRSXx8
-PATH=/home/mu/.local/bin:/home/mu/bin:/usr/lib/lightdm/lightdm:/usr/local/sbin:/usr/
↪local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
-PROFILEHOME=
-PROJECTS_DIR=/home/mu/Projekte
-PROMPT_DIRTRIM=4
-PWD=/home/mu/Projekte/think-rotate/doc/environment
-QT_PLUGIN_PATH=/home/mu/.kde/lib/kde4/plugins/:/usr/lib/kde4/plugins/
-SESSION=kde-plasma
-SESSION_MANAGER=local/Martin-X220:@/tmp/.ICE-unix/2000,unix/Martin-X220:/tmp/.ICE-
↪unix/2000
-SESSIONTYPE=
+LOGNAME=root
+MAIL=/var/mail/root
+PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
 SHELL=/bin/bash
-SHELL_SESSION_ID=a7053eff52fd4000b6f58d0783a5b9ba
-SHLVL=2
-SSH_AGENT_PID=1867
-SSH_AUTH_SOCK=/tmp/ssh-KejIl272p39R/agent.1862
-TAR_ARCHIVE_DIR=/home/mu/Packaging_Debian
+SUDO_COMMAND=/usr/bin/env
+SUDO_GID=1000
+SUDO_UID=1000
+SUDO_USER=mu
 TERM=xterm-256color
-TEXTDOMAINDIR=/usr/share/locale/
-TEXTDOMAIN=im-config
-UPSTART_EVENTS=started xsession
-UPSTART_INSTANCE=
-UPSTART_JOB=startkde
-UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1797
-USER=mu
-_=/usr/bin/env
-WINDOWID=52428826
+USERNAME=root
+USER=root
 XAUTHORITY=/tmp/kde-mu/xauth-1000-_0
-XCURSOR_THEME=default
-XDG_CONFIG_DIRS=/etc/xdg/xdg-kde-plasma:/usr/share/upstart/xdg:/etc/xdg
-XDG_CURRENT_DESKTOP=KDE
-XDG_DATA_DIRS=/usr/share:/usr/share/kde-plasma:/usr/local/share/:/usr/share/
-XDG_RUNTIME_DIR=/run/user/1000
-XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
-XDG_SEAT=seat0
-XDG_SESSION_ID=c2
-XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
-XDG_VTNR=7
```

Difference user and `su`

If you look the differences to an environment made with `su`, there is less changed, but still a lot missing. I made the following with (where `mu` is my user account):

```
# su -c env mu
```

This is the output:

```
--- env-user.txt        2014-01-07 08:10:57.297768327 +0100
+++ env-su.txt        2014-01-07 08:10:57.225768328 +0100
@@ -1,76 +1,24 @@
-BROWSER=/usr/bin/firefox
-COLORFGBG=15;0
-DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-TK74xcpHuB
-DEBEMAIL=dev@martin-ueding.de
-DEBFULLNAME=Martin Ueding
-DEBIAN_PACKAGING_DIR=/home/mu/Packaging_Debian
-DEFAULTS_PATH=/usr/share/gconf/kde-plasma.default.path
-DESKTOP_SESSION=kde-plasma
 DISPLAY=:0
-EDITOR=/usr/bin/vim
-GDM_LANG=en
-GDMSESSION=kde-plasma
-GNOME_KEYRING_CONTROL=/run/user/1000/keyring-35U0Ew
-GNOME_KEYRING_PID=1784
-GPG_AGENT_INFO=/tmp/gpg-BQD8Wt/S.gpg-agent:1864:1
-GS_LIB=/home/mu/.fonts
-GTK2_RC_FILES=/etc/gtk-2.0/gtkrc:/home/mu/.gtkrc-2.0:/home/mu/.kde/share/config/
↪gtkrc-2.0
-GTK_RC_FILES=/etc/gtk/gtkrc:/home/mu/.gtkrc:/home/mu/.kde/share/config/gtkrc
 HOME=/home/mu
-IM_CONFIG_PHASE=1
-INSTANCE=
-JOB=dbus
-KDE_FULL_SESSION=true
-KDE_MULTIHEAD=false
-KDE_SESSION_UID=1000
-KDE_SESSION_VERSION=4
-KONSOLE_DBUS_SERVICE=:1.72
-KONSOLE_DBUS_SESSION=/Sessions/2
-KONSOLE_DBUS_WINDOW=/Windows/1
-KONSOLE_PROFILE_NAME=Shell
 LANG=de_DE.UTF-8
 LANGUAGE=en
-LESSCLOSE=/usr/bin/lesspipe %s %s
-LESS=-FRSXx8
-LESSOPEN=| /usr/bin/lesspipe %s
 LOGNAME=mu
-MAKEFLAGS=-j 4
-MANDATORY_PATH=/usr/share/gconf/kde-plasma.mandatory.path
-MATHEMATICA_HOME=/usr/local/Wolfram/Mathematica/9.0
-PAGER=less -FRSXx8
-PATH=/home/mu/.local/bin:/home/mu/bin:/usr/lib/lightdm/lightdm:/usr/local/sbin:/usr/
↪local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
-PROFILEHOME=
-PROJECTS_DIR=/home/mu/Projekte
-PROMPT_DIRTRIM=4
```

```
+MAIL=/var/mail/mu
+PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/
↪local/games
 PWD=/home/mu/Projekte/think-rotate/doc/environment
-QT_PLUGIN_PATH=/home/mu/.kde/lib/kde4/plugins/:/usr/lib/kde4/plugins/
-SESSION=kde-plasma
-SESSION_MANAGER=local/Martin-X220:@/tmp/.ICE-unix/2000,unix/Martin-X220:/tmp/.ICE-
↪unix/2000
-SESSIONTYPE=
 SHELL=/bin/bash
-SHELL_SESSION_ID=a7053eff52fd4000b6f58d0783a5b9ba
-SHLVL=2
-SSH_AGENT_PID=1867
-SSH_AUTH_SOCK=/tmp/ssh-KejIl272p39R/agent.1862
-TAR_ARCHIVE_DIR=/home/mu/Packaging_Debian
+SHLVL=1
+SUDO_COMMAND=/bin/su -c env mu
+SUDO_GID=1000
+SUDO_UID=1000
+SUDO_USER=mu
 TERM=xterm-256color
-TEXTDOMAINDIR=/usr/share/locale/
-TEXTDOMAIN=im-config
-UPSTART_EVENTS=started xsession
-UPSTART_INSTANCE=
-UPSTART_JOB=startkde
-UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1797
 USER=mu
+USERNAME=root
 _=/usr/bin/env
-WINDOWID=52428826
 XAUTHORITY=/tmp/kde-mu/xauth-1000-_0
-XCURSOR_THEME=default
-XDG_CONFIG_DIRS=/etc/xdg/xdg-kde-plasma:/usr/share/upstart/xdg:/etc/xdg
-XDG_CURRENT_DESKTOP=KDE
-XDG_DATA_DIRS=/usr/share:/usr/share/kde-plasma:/usr/local/share/:/usr/share/
 XDG_RUNTIME_DIR=/run/user/1000
-XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
 XDG_SEAT=seat0
+XDG_SESSION_COOKIE=550ca4757c0af12b457994cd526d188e-1389078657.209124-1472854378
 XDG_SESSION_ID=c2
-XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
 XDG_VTNR=7
```

Shared properties

In both cases, the variable `DBUS_SESSION_BUS_ADDRESS` is missing. This might be cause for bugs like #36.

Difference user and `su -c env`

Even if you use `env` to recreate the user's environment, it does not change anything, apparently:

```
su -c 'env env' mu
```

```
--- env-user.txt        2014-01-07 08:10:57.297768327 +0100
+++ env-su_env.txt        2014-01-07 08:10:57.277768327 +0100
@@ -1,76 +1,24 @@
-BROWSER=/usr/bin/firefox
-COLORFGBG=15;0
-DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-TK74xcpHuB
-DEBEMAIL=dev@martin-ueding.de
-DEBFULLNAME=Martin Ueding
-DEBIAN_PACKAGING_DIR=/home/mu/Packaging_Debian
-DEFAULTS_PATH=/usr/share/gconf/kde-plasma.default.path
-DESKTOP_SESSION=kde-plasma
 DISPLAY=:0
-EDITOR=/usr/bin/vim
-GDM_LANG=en
-GDMSESSION=kde-plasma
-GNOME_KEYRING_CONTROL=/run/user/1000/keyring-35U0Ew
-GNOME_KEYRING_PID=1784
-GPG_AGENT_INFO=/tmp/gpg-BQD8Wt/S.gpg-agent:1864:1
-GS_LIB=/home/mu/.fonts
-GTK2_RC_FILES=/etc/gtk-2.0/gtkrc:/home/mu/.gtkrc-2.0:/home/mu/.kde/share/config/
↪gtkrc-2.0
-GTK_RC_FILES=/etc/gtk/gtkrc:/home/mu/.gtkrc:/home/mu/.kde/share/config/gtkrc
 HOME=/home/mu
-IM_CONFIG_PHASE=1
-INSTANCE=
-JOB=dbus
-KDE_FULL_SESSION=true
-KDE_MULTIHEAD=false
-KDE_SESSION_UID=1000
-KDE_SESSION_VERSION=4
-KONSOLE_DBUS_SERVICE=:1.72
-KONSOLE_DBUS_SESSION=/Sessions/2
-KONSOLE_DBUS_WINDOW=/Windows/1
-KONSOLE_PROFILE_NAME=Shell
 LANG=de_DE.UTF-8
 LANGUAGE=en
-LESSCLOSE=/usr/bin/lesspipe %s %s
-LESS=-FRSXx8
-LESSOPEN=| /usr/bin/lesspipe %s
 LOGNAME=mu
-MAKEFLAGS=-j 4
-MANDATORY_PATH=/usr/share/gconf/kde-plasma.mandatory.path
-MATHEMATICA_HOME=/usr/local/Wolfram/Mathematica/9.0
-PAGER=less -FRSXx8
-PATH=/home/mu/.local/bin:/home/mu/bin:/usr/lib/lightdm/lightdm:/usr/local/sbin:/usr/
↪local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
-PROFILEHOME=
-PROJECTS_DIR=/home/mu/Projekte
-PROMPT_DIRTRIM=4
+MAIL=/var/mail/mu
+PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/
↪local/games
 PWD=/home/mu/Projekte/think-rotate/doc/environment
-QT_PLUGIN_PATH=/home/mu/.kde/lib/kde4/plugins/:/usr/lib/kde4/plugins/
-SESSION=kde-plasma
-SESSION_MANAGER=local/Martin-X220:@/tmp/.ICE-unix/2000,unix/Martin-X220:/tmp/.ICE-
↪unix/2000
-SESSIONTYPE=
 SHELL=/bin/bash
-SHELL_SESSION_ID=a7053eff52fd4000b6f58d0783a5b9ba
-SHLVL=2
-SSH_AGENT_PID=1867
-SSH_AUTH_SOCK=/tmp/ssh-KejI1272p39R/agent.1862
-TAR_ARCHIVE_DIR=/home/mu/Packaging_Debian
+SHLVL=1
+SUDO_COMMAND=/bin/su -c env env mu
+SUDO_GID=1000
+SUDO_UID=1000
+SUDO_USER=mu
 TERM=xterm-256color
-TEXTDOMAINDIR=/usr/share/locale/
-TEXTDOMAIN=im-config
-UPSTART_EVENTS=started xsession
-UPSTART_INSTANCE=
-UPSTART_JOB=startkde
-UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1797
 USER=mu
+USERNAME=root
 _=/usr/bin/env
-WINDOWID=52428826
 XAUTHORITY=/tmp/kde-mu/xauth-1000-_0
-XCURSOR_THEME=default
-XDG_CONFIG_DIRS=/etc/xdg/xdg-kde-plasma:/usr/share/upstart/xdg:/etc/xdg
-XDG_CURRENT_DESKTOP=KDE
-XDG_DATA_DIRS=/usr/share:/usr/share/kde-plasma:/usr/local/share/:/usr/share/
 XDG_RUNTIME_DIR=/run/user/1000
-XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
 XDG_SEAT=seat0
+XDG_SESSION_COOKIE=550ca4757c0af12b457994cd526d188e-1389078657.247557-1009355119
 XDG_SESSION_ID=c2
-XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
 XDG_VTNR=7
```

Why Python?

Author Martin Ueding <dev@martin-ueding.de>

Advantages

Why do I want to switch to Python 3 over Bash?

INI config format The Bash implementation just sourced its config file. So you could write a little shell script that assigned a couple variables and you were done with it.

Python has the `configparser` module which enables using INI style config files. There is a default configuration which can be overwritten. This offers sectioning in the config, which makes sense with the current amount of options.

Language offers module Bash has no notion of modules. It just has scripts that could be sourced. This was done with the `lib` folder, but it has felt like a hack from the very beginning. When I documented those functions, I realized that it was hard to do that since Bash functions do not have named parameter like real programming languages. The biggest issue are the missing return values.

Since modules are easy with Python, I can split the long scripts into multiple modules.

Scoping The Bash implementation used a lot of global variables. Like the `$external` that appeared magically when you would call the `find-external` function. I believe in the “explicit is better than implicit” statement of the Python Zen, so this bugged me.

With the scopes in the functions, I can create lot of simple, small functions that can be tested and reused better.

XML support With the XML config file for fontconfig coming up, I wanted to have a language that can work with XML files natively. Using `sed` on a XML file just seems wrong. Well, XML as configuration seems wrong as well.

Direct GUI So far, the GUI has been made with `kdialog` which received messages via `qdbus`. This works. But with Python, a binding like PyQt can be used to create a real GUI.

Better string processing There are lines like the following in the 3.x codebase:

```
external=$(xrandr | grep -Eo '(\S+) connected' | grep -Eo '^(\S+)' | grep -v "
↪$internal")
```

I think this can be done much nicer in Python.

```
def get_external(internal):
    lines = tps.check_output(['xrandr'], logger).decode().split('\n')
    for line in lines:
        if not line.startswith(internal):
            matcher = re.search(r'^(\S+) connected', line)
            if matcher:
                return matcher.group(1)
```

Yes, it is way more code. But I find it easier to read and more self explanatory.

Possibility of a daemon It would be possible to have this running as a daemon which gets messages via D-Bus. The hooks that get called with hardware events are run as root and without the `DISPLAY` variable set. The 3.x code uses `su` to run the code in the context of the user. With such a daemon, it would be possible to avoid that and invoke the action.

This has the disadvantage of an always running daemon, which is not really needed.

Better documentation Docstrings and Sphinx allows one to document the hell out of code. And it is quite fun. With that, it is really easy to document the various parts of the codebase in a standard way.

Disadvantages

I do see some disadvantages. They are not big issues, I think.

Requires Python knowledge It will require the developers to know Python. Or they will have to be willing to learn it for this project. I do not consider Python an uncommon language, not more uncommon than Bash, on Linux. Look at various Ubuntu scripts, they are written in Python.

Bash is pretty hard to get right with all its pitfalls. So Python might be an easier choice to get people to contribute.

Adds more dependencies Bash is included in virtually every distribution. Python should be as well, but there might be somebody without Python on his system.

Python Module Index

Index