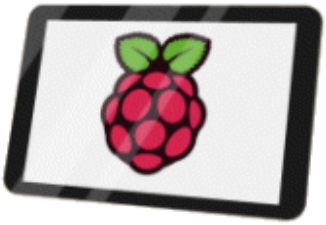


Raspberry Pi: Infoscreen- und Kiosksystem

Prof. Jürgen Plate

Raspberry Pi: Infoscreen- und Kiosksystem

Sie kennen sicher Computersysteme, die Informationen zu Museums-Exponaten bieten und interaktiv abrufbar machen. Auch wenn Sie am Bankautomaten Geld abheben oder eine Fahrkarte kaufen, inzwischen auch bei der Heimautomatisierung, die Arbeit leistet ein für genau diesen Zweck eingerichteter Computer. Die übliche Bezeichnung solcher öffentlichen Terminals lautet "infoscreen", "Infoboard" oder "Kiosksystem". Durch die Selbstbedienung der Kunden sparen die Betreiber Kosten und reduzieren Wartezeiten.

Viele Kiosksysteme verkaufen nichts, sondern informieren nur, z. B.in Museen oder bei Veranstaltungen. Dazu erlauben solche Internetterminals oder Surfstationen nur den Zugriff auf den Browser, das System selbst muss dem Nutzer verborgen bleiben. Es geht also darum, den Raspberry Pi so zu konfigurieren, dass

- die grafische Oberfläche (GUI) automatisch gestartet wird,
- der User "pi" (oder ein anderer Standard-User) automatisch angemeldet wird,
- der Screensaver abgeschaltet wird und
- ein Programm (Browser oder was auch immer) gestartet wird.

Zum Aufbau eines Kiosksystems eignet sich der Raspberry Pi bestens. Er besitzt genügend Rechenleistung, um Webseiten ausreichend schnell darzustellen, und verbraucht kaum Strom. Für ein Infoboard wird benötigt:

- Raspberry-Pi mit Netzteil (1,2 - 2 A)
- SD-Card: mind. 4 GB, max. 32 GB
- Monitor mit DVI- oder HDMI-Eingang (DVI über HDMI-DVI-Adapter) und Kabel
- Zur Erst-Installation: USB-Maus und USB-Tastatur
- Gehäuse für den Pi

Im Folgenden wird davon ausgegangen, dass das Raspbian-Linux auf dem RasPi installiert und so eingerichtet ist, dass die grafische Oberfläche LXDE automatisch startet. Es muss ausserdem eine SSH-Verbindung zum RasPi aufgebaut werden können. Auch der schlanke Webbrowser Midori muss verfügbar sein. Die Informationen werden als HTML-Seiten über den Webbrowser angeboten. Dr Raspberry Pi soll nach dem Booten automatisch den Browser starten. Alle Aktionen dazu werden über Shellscripte auf der Kommandozeile gesteuert. Zur Einrichtung des Kiosksystems sind die folgen Schritte notwendig. **Alle Arbeiten müssen als Benuzter "root" ausgeführt werden**, daher geben Sie am besten gleich das Kommando

```
sudo su
```

ein, um nicht ständig `sudo` verwenden zu müssen.

Damit der Browser im Kiosk-Mode läuft und zum Beispiel der Mauszeiger nicht dargestellt wird, müssen noch weitere Programme installiert werden:

```
apt-get install x11-xserver-utils unclutter
```

Die X11-Serverutilities brauchen Sie für `xset` und `unclutter` lässt den Mauszeiger verschwinden. Optional könnte man noch ein schlankes X-Window-System, beispielsweise mittels `apt-get install matchbox-window-manager` installieren, was aber bei den aktuellen Modellen des Pi nicht nötig ist.

Benutzer Pi automatisch anmelden

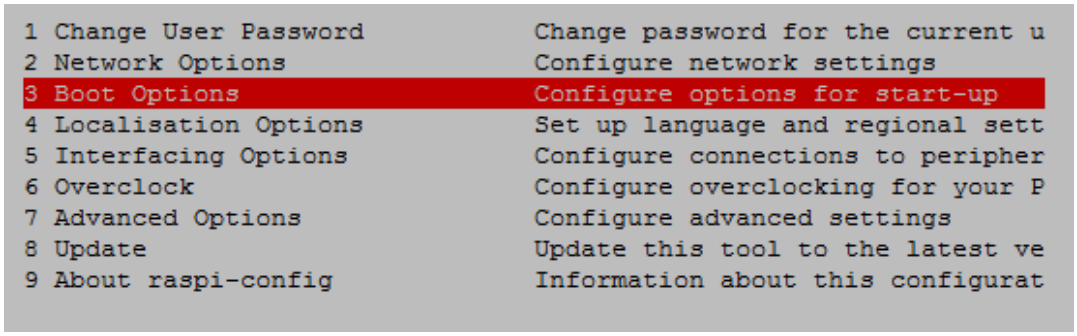
Da der Browser und alles andere über die Kommandozeile gestartet werden sollen, muss der Benutzer Pi nach dem Booten des Systems an der Kommandozeile angemeldet sein. Bei früheren Raspbian-Versionen musste dazu die Datei `/etc/inittab` angepasst werden:

```
1:2345:respawn:/sbin/getty --noclear 38400 tty1
2:23:respawn:/sbin/getty 38400 tty2
3:23:respawn:/sbin/getty 38400 tty3
4:23:respawn:/sbin/getty 38400 tty4
5:23:respawn:/sbin/getty 38400 tty5
6:23:respawn:/sbin/getty 38400 tty6
```

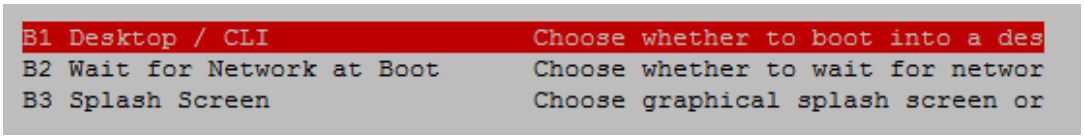
Für `tty1` wird der User pi automatisch eingeloggt, alle anderen TTYs werden gesperrt. Kommentieren Sie alle Zeilen mit # aus und fügen die folgenden beiden Zeilen darunter ein.

```
1:2345:respawn:/sbin/getty --autologin pi --noclear 38400 tty1
#2:23:respawn:/sbin/getty 38400 tty2
#3:23:respawn:/sbin/getty 38400 tty3
#4:23:respawn:/sbin/getty 38400 tty4
#5:23:respawn:/sbin/getty 38400 tty5
#6:23:respawn:/sbin/getty 38400 tty6
```

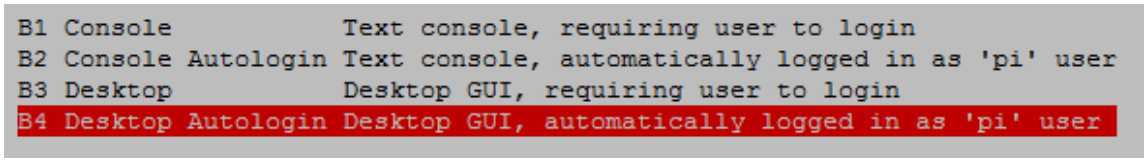
Ab der Version "Jessie" von Raspbian geht das alles viel einfacher, nämlich über das Standard-Tool `raspi-config`. Hier wählen Sie zunächst im Hauptmenü den Punkt 3 "Boot Options":



Danach wählen Sie den obersten Punkt "B1 Desktop / CLI":



Nun gibt es vier Auswahlmöglichkeiten, Login auf der Kommandozeile oder in die GUI, jeweils automatisch oder mittels Passwort. Hier wird der letzte Punkt, "B4 Desktop Autologin", gewählt:



Falls es Sie interessiert, was dabei jeweils hinter den Kulissen geschiet, könnten Sie einen Blick in das `raspi-config` werfen. Da es sich um ein Shellscript handelt ist alles schön sichtbar. Ich habe zur Info unten die relevanten Befehle für den Autologin mal herausgezogen (und wegen der Optik etwas umbrochen). Zunächst der Autologin auf der Kommandozeile (CLI):

```
   ...
systemctl set-default multi-user.target
sed /etc/systemd/system/autologin@.service -i \
  -e "s#^ExecStart=-/sbin/agetty --autologin [^[:space:]]*#ExecStart=-/sbin/agetty --autologin pi#"
ln -fs /etc/systemd/system/autologin@.service /etc/systemd/system/getty.target.wants/getty@tty1.service
   ...
```

Der Rechner wird auf den Standard-Multiusermodus geschaltet und dann eine Autologin-Steuerdatei für den User "pi" erstellt (im Original wird der User genommen, der das `raspi-config`-Script aufgerufen hat. Für die GUI

sieht es ähnlich aus. Hier wird zunächst geprüft, ob der Window-Manager `lightdm` auch verfügbar ist und erst dann der Autostart eingerichtet. Andernfalls erfolgt eine Fehlermeldung:

```
   ...
if [ -e /etc/init.d/lightdm ]; then
  systemctl set-default graphical.target
  ln -fs /etc/systemd/system/autologin@.service \
    /etc/systemd/system/getty.target.wants/getty@tty1.service
  sed /etc/lightdm/lightdm.conf -i \
    -e "s/^\(#\|\)autologin-user=.*/autologin-user=$SUDO_USER/"
  disable_raspi_config_at_boot
else
  whiptail --msgbox "Do 'sudo apt-get install lightdm' ..." 20 60 2
  return 1
fi
   ...
```

Den X-Server automatisch starten

Normalerweise wird die GUI automatisch gestartet - außer Sie haben das definitiv abgeschaltet oder die Light-Version von Raspbian installiert. Deshalb nur der Vollständigkeit halber: Falls Sie nicht wollen, dass die grafische Oberfläche automatisch beim Booten startet, kann der Start des X-Servers auch an den User Pi deligiert werden. In der Datei `/home/pi/.bash_profile` (wahlweise auch `/home/pi/.profile` tragen sie nur die nachfolgenden Befehle (am besten ganz am Ende vor dem `exit 0`) ein:

```
# X-Server starten
startx
```

Für den Start des Kiosk-Systems können Sie schon mal den Aufruf des Skriptes in der Datei `/home/pi/.xinitrc` eintragen:

```
/home/pi/infoscreen_start
```

Anmerkung: Gegebenenfalls muss man die Grafikauflösung an den gewählten Monitor anpassen. Das ist oft dann notwendig, wenn kleine HDMI-Displays mit 5 bis 7 Zoll Diagonale verwendet werden. Die Monitorauflösung kann man in der Datei `/boot/config.txt` einstellen. Um beispielsweise eine Auflösung von 1680x1050 Pixeln, 60Hz vorzugeben, trägt man ein:

```
...
hdmi_group=2
hdmi_mode=58
hdmi_force_hotplug=1
```

Die Grafik-Modi können der Dokumentation von `/boot/config.txt` entnommen werden.

Startskript für das Kiosksystem

Bei den früheren Raspbian-Versionen (vor "Jessie") war nun Handarbeit angesagt, wobei die im Folgenden geschilderte Methode immer noch funktionieren dürfte und in manchen Fällen immer noch ihre Berechtigung hat, etwa bei wirklich minimalistischen Systemen. Je nach Anwendung könnte man ja auch noch auf das Desktop-Environment und den Windowmanager verzichten und eine einzige X-Applikation starten. Deshalb und zur Info hier der Weg über ein Startscript.

Kiosk per Startscript

Alles Weitere erledigt das Startskript, das je nach Gegebenheit beliebig erweitert werden kann. In diesem Fall endet es mit dem Start des Browsers `midori`, in dem dann alles Weitere abläuft. Es steht (wie oben angegeben) in der Datei `/home/pi/infoscreen_start`:

```
#!/bin/bash
# dem System etwas Zeit geben (z. B. nach einen Restart)
sleep 10

export DISPLAY=:0
```

```
# ggf. Monitorauflösung setzen
#xrandr -s 1024x768 &

# Bildschirmschoner ausschalten I
xset s off
xset s noblank

#  Bildschirmschoner ausschalten II
perl -pi -e "s/BLANK_TIME=[0-9]+/BLANK_TIME=0/g" /etc/kbd/config
perl -pi -e "s/POWERDOWN_TIME=[0-9]+/POWERDOWN_TIME=0/g" /etc/kbd/config

# Energiesparfunktionen ausschalten
xset -dpms

# Mauszeiger und Cursor verstecken
unclutter -grab -visible &

# Optional: Run window manager
matchbox-window-manager -use_cursor no -use_titlebar no  &

# Endlosschleife, falls midori abkackt, neu starten
while :; do
  ps -ef | grep 'midori -e Fullscreen' | grep -v grep > /dev/null 2>&1
  if [ $? -eq 1 ] ; then
    midori -e Fullscreen -a <Startseite> &
  else
    midori -e Reload
  fi
  sleep 1800
done
```

Das Script endet in einer Endlosschleife, in der regelmäßig geprüft wird, ob der Browser noch läuft. Andernfalls wird der Browser im Vollbildmodus und als Hintergrundprozeß gestartet. Falls der Browser noch läuft, wird sicherheitshalber nach einen halben Stunde (1800 Sekunden) ein Reload der Webseite durchgeführt. Bei `<Startseite>` muss natürlich die Start-Webseite des Infoboards eingetragen werden. Vergessen Sie nicht, das Script ausführbar zu machen:

```
chmod 755 /home/pi/infoscreen_start
```

Die Option "-e" kenn noch weitere Argumente, die Sie mit `midori --help-execute` auflisten können.

Was mir manchmal nicht so gefällt, sind Scripten, die in einer Endlosschleife laufen wie das obige. Man kann die Schleife aber loswerden, wenn man die Teile, die in der Schleife laufen, an einen Auftrag für den Cron-Daemon deligiert. Das obige Script wird geteilt. Ein Teil wird wie oben einmal aufgerufen und startet alles. Über das `crontab`-Kommando wird ein zweites Script regelmäßig aktivieret, dass dann den Midori-Browser überwacht. Hier das geänderte Script `/home/pi/infoscreen_start`:

```
#!/bin/bash
# dem System etwas Zeit geben (z. B. nach einen Restart)
sleep 10

export DISPLAY=:0

# ggf. Monitorauflösung setzen
#xrandr -s 1024x768 &

# Bildschirmschoner ausschalten I
xset s off
xset s noblank

#  Bildschirmschoner ausschalten II
perl -pi -e "s/BLANK_TIME=[0-9]+/BLANK_TIME=0/g" /etc/kbd/config
perl -pi -e "s/POWERDOWN_TIME=[0-9]+/POWERDOWN_TIME=0/g" /etc/kbd/config
perl -pi -e "s/BLANK_DPMS=[a-zA-Z]+/BLANK_DPMS=off/g" /etc/kbd/config

# Energiesparfunktionen ausschalten
xset -dpms

# Mauszeiger und Cursor verstecken
unclutter -grab -visible &
```

```
# Midori starten
midori -e Fullscreen -a <Startseite> &
```

Der Aufruf des zweiten Scripts wird per `crontab -e` eingetragen. Es wird alle halbe Stunden gestartet:

```
0,30 * * * * /home/pi/midoriwatch
```

Das Script `/home/pi/midoriwatch` erledigt nun den Job, der vorher in der Schleife stand - nur wegen `crontab` eben ohne Schleife und `sleep`:

```
#!/bin/bash
# falls midori abkackt, neu starten
ps -ef | grep 'midori -e Fullscreen' | grep -v grep > /dev/null 2>&1
if [ $? -eq 1 ] ; then
  midori -e Fullscreen -a <Startseite> &
else
  midori -e Reload
fi
```

Vergessen Sie nicht, das Script ausführbar zu machen:

```
chmod 755 /home/pi/midoriwatch
```

In beiden Scripten muss auch noch die aktuelle Startseite eingetragen werden.

Alternativ kann bei den leistungsfähigeren Raspberries auch der Browser "Chromium" eingesetzt werden. Der Aufruf im KIosk-Mode lautet dann:

```
chromium --kiosk --incognito <Startseite>
```

Autostart mit LXDE

Der Standard-Desktop bei Raspbian ist LXDE (Lightweight X11 Desktop Environment). Auch beim Starten bzw. Anmelden eines Benutzers am Desktop lassen sich Programme per Autostart ausführen. Dazu wird die Datei

```
/etc/xdg/lxsession/LXDE-pi/autostart
```

bearbeitet, die für Systemweite Einstellungen zuständig ist (für einen einzelnen Benutzer kann die Datei `~/.config/lxsession/LXDE/autostart` entsprechend bearbeitet werden). Jede Zeile enthält ein auszuführendes Kommando. Wenn einem Kommando mit einem "@" beginnt, wird dieses automatisch erneut gestartet, wenn es aus irgendeinem Grund terminiert. Damit ist auch die oben skizzierte Überwachung nicht mehr notwendig. Die originale Datei (bei "Stretch") hat den Inhalt:

```
@lxpanel --profile LXDE-pi
@pcmanfm --desktop --profile LXDE-pi
@xscreensaver -no-splash
@point-rpi
```

Dieser wird nun geändert, indem einige Programmaufrufe durch Voranstellen eines "#" auskommentiert werden und man weitere Programme hinzufügt. Damit wirde der gleiche Zweck wie beim vorstehenden Script erfüllt.

```
#Disable screensaver:
@xset s noblank
@xset s off
@xset -dpms
#@lxpanel --profile LXDE-pi
#@pcmanfm --desktop --profile LXDE-pi
#@xscreensaver -no-splash
#@point-rpi
```

Ans Ende der Datei kommt nun der Aufruf des Kiosk-Programms. Gegebenefall können auch mehrere Programme gestartet werden, zum Beispiel für den Midori-Browser:

```
@midori -e Fullscreen -a <Startseite>
```

Für Shell- oder Python-Scripte wird analog verfahren, beispielsweise:

```
@lxterminal -e /home/pi/bin/shellscript.sh
@python /home/pi/bin/pythonscript.py
```

Update/Spezielles

Christoph Naethbohm vom Zentrum für Datenverarbeitung der Uni Mainz schrieb zu einem speziellen Problem:

Habe es nun Tricky geschafft, die Fehlermeldung von Midory abzufangen, wenn die Netzwerkverbindung nicht klappt oder der Webserver kaputt ist. Wir haben hier in der Uni Mainz schon einige Zeit einige Bildschirme mit `opennms` und Kopierüberwacher laufen.

Um die Fehlermeldungen abzufangen, wird ein Screenshot vom aktuellem Bildschirm genommen (Programm `scrot`). Mit `convert` wird der benötigte Auschschnitt extrahiert (damit `tesseract` schneller ist) und mittels Texterkennung `tesseract` geprüft, ob das Wort "Error" auftritt. Midori hat die Fehlermeldung glücklicherweise immer an der selben Stelle. Alles steckt nun im folgenden Cronjob. Das Programm `scrot` benutze ich auch, um mit dem Webfrontend den Status-Bildschirm anzuzeigen. Aber manchmal schaut man nur alle Wochen nach und dann wird es peinlich, wenn der Errorschirm zu sehen ist. Den Midori-Restart erledigt immer morgens der Cron, wenn der Bildschirm noch dunkel ist (tvservice). Somit werden benötigt:

- `scrot` (Bildschirmhardcopy)
- `imagemagick` (`convert` für Ausschnitt)
- `tesseract` (OCR Reader)

Der Cronjob wird dann durch das folgende Script realisiert. Die Daten werden in einer RAM-Partition gespeichert (siehe auch RasPi_Install.html, Schreibzugriff auf SD-Karte reduzieren):

```
#!/bin/bash
# tesseract kann scheinbar die Datei test.txt nicht überschreiben
rm /home/pi/ram/text.txt
/bin/sleep 1
# Screenshot
export DISPLAY=:0 && /usr/bin/scrot /home/pi/ram/current.png
/bin/sleep 1
# Ausschnitt erzeugen
convert /home/pi/ram/current.png -crop 200x100+410+60 /home/pi/ram/convert.png
# Texterkennung
tesseract /home/pi/ram/convert.png /home/pi/ram/text
# Suche nach "Error"
cat /home/pi/ram/text.txt | grep "Error"
if grep "Error" /home/pi/ram/text.txt
then
    echo  "Fehler gefunden" >/home/pi/ram/error.log
    url=$(cat /home/pi/ram/url.txt)
    echo $url
    export DISPLAY=:0
    kill $(pgrep midori)
    /usr/bin/midori -e Fullscreen -a $url &
else
    echo  "keinen Fehler gefunden do nothing" >/home/pi/ram/error.log
fi
```

Das System härten

[Strg]-[Alt]-[Del] abschalten

Natürlich soll niemand per Tastatur den Raspberry Pi neu starten könnne. Früher war das Verhalten des "Affengriffs" zum Rebooten/Herunterfahren des Systems auch in der dastei `/etc/inittab` verankert und es genügt ein Kommentar-# zum Abschalten. Inzwischen ist die Aufgabe der Tastenkombination an den `sytemd` übergegangen. Um nun die Taste zu deaktivieren, muss eine passende Steuerdatei, genauer eine symbolischer Link angelegt werden:

```
ln -sf /dev/null /etc/systemd/system/ctrl-alt-del.target
```

Ab da wird beim Drücken von `[Strg]-[Alt]-[Del]` nichts mehr getan.

[Strg]-[Alt]-[Backspace] abschalten

Auch die Taste zum Beenden der Grafikoberfläche muss dekativiert werden. Das kann man schon beim Einrichten des Systems mit `raspi-config` erledigen. Andernfalls legen Sie eine Datei namens `/etc/X11/xorg.conf.d/10-dontzap.conf` an (der Name ist eigentlich egal) und geben folgendes in die Datei ein:

```
Section "ServerFlags"
  Option "DontZap"  "true"
EndSection
```

System-Request-Taste [Alt]-[S-Abf] abschalten

Das ist die gefährlichste Taste überhaupt, denn mit ihr lassen sich verschiedene Funktionen, z. B. auch ein Neustart des Computers, ausführen. Das funktioniert sogar, wenn der Computer auf andere Eingaben nicht mehr reagiert, etwa wenn die Grafikoberfläche "hängt". Häufig wird so ein Neustart durchgeführt, ohne Schäden am Dateisystem zu verursachen. Auf einem Linux-System hält man die Tastenkombination [Alt][S-Abf] gedrückt und drückt nacheinander die Tasten [R] (unraw), [E] (term), [I] (kill), [S] (sync), [U] (Umount) und [B] (boot). Daneben gibt es noch andere Kombinationen, etwa [O] für Power Off.

Um die Taste zu temporär zu deaktivieren, kann man eine Null in die Pseudodatei `/proc/sys/kernel/sysrq` schreiben:

```
echo 0 > /proc/sys/kernel/sysrq
```

Diese Zeile könnten Sie auch in das Script `/home/pi/infoscreen_start` schreiben. Um die System-Request-Taste dauerhaft zu deaktivieren, legen Sie eine Steuerdatei für `sysctl` an:

```
echo "kernel.sysrq=0" > /etc/sysctl.d/10-magic-sysrq.conf
sysctl -p /etc/sysctl.d/10-magic-sysrq.conf
```

Beim Booten wird diese Datei dann mit allen anderen `syctl`-Configs ausgewertet. Siehe auch Wikipedia: Magische_S-Abf-Taste.

Deaktivieren Rechtsklick und Alt

Der Befehl `xmodmap` in der Autostart-Datei deaktiviert die rechte Maustaste durch Neuzuordnung der mittleren und rechten Tasten. Bei der Wahl des Tastaturtreibers wurde nun jeder Taste (also jedem "keycode") ein gewisses Zeichen bzw. Funktion in Form von keysyms zugewiesen. Mit `xmodmap` kann man diese Einstellungen individuell anpassen. Das Programm modifiziert die jeweils aktuelle Keymap (die Zuweisungen des Signals, das die Taste sendet (= keycode) und die Aktion, die durch dieses Signal ausgelöst werden soll). Dies ist unabhängig vom gewählten Tastaturlayout. Um nun Zuweisungen zu ändern, wird eine eigene Datei `.Xmodmap` im heimatverzeichnis erzeugt und dann geändert:

```
cd /home/pi
xmodmap -pke > .Xmodmap
```

In der Datei stehen die Zuordnungen und Aktionen zu den Tastencodes. Das kann z. B. folgendermassen aussehen:

```
keycode   8 =
keycode   9 = Escape
keycode  10 = 1 plus plusminus infinity
```

Hinter dem Tastencode können bis zu sechs Keywords stehen.

- Die Aktion des ersten Keywords ist der Text bzw. die Funktion, welche die Taste bei einfachem Drücken erzeugt.
- Das zweite Keyword gilt, wenn zusätzlich die Shift-Taste gedrückt ist.
- Das dritte Keyword gilt, wenn zusätzlich die Alt-Taste (Mode_Switch) gedrückt ist.
- Das vierte Keyword gilt, wenn zusätzlich die Shift-Taste und die Alt-Taste gedrückt sind.
- Das fünfte Keyword gilt, wenn zusätzlich die AltGr-Taste gedrückt ist.
- Das sechste Keyword gilt, wenn zusätzlich die Shift-Taste und die AltGr-Taste gedrückt sind.

Fehlt eines oder mehrere Keywords, gibt es die entsprechende Funktion nicht. So ist im Beispiel oben die Taste mit Code 8 unbelegt und die Escape-Taste nur einfach belegt. Um nun Tasten zu deaktivieren oder auch umzudefinieren, bearbeiten wir die private `.Xmodmap` des Users pi. Um die Tasten [Alt] und [AltGr] zu deaktivieren, ändern Sie den Keycode:

```
keycode 64 = NoSymbol NoSymbol NoSymbol NoSymbol
keycode 108 = NoSymbol NoSymbol NoSymbol NoSymbol
```

oder ganz brutal

```
keycode 64 =
keycode 108 =
```

Ähnlich können Sie auch mit den anderen Keycodes verfahren. Rufen Sie das Kommando `xmodemap` ohne Parameter auf, liefert das Programm eine Liste der aktuellen Zuordnungen. Nach Aktivierung durch Ab- und Anmelden des Users oder durch ausführen von

```
xmodmap ~/.Xmodmap
```

wird die Zuordnung aktualisiert.

Keyboard-Shortcuts deaktivieren

Für die Shotcuts ist die Datei `/home/pi/.config/openbox/lxde-rc.xml` zuständig. Von dieser Datei machen Sie zunächst ein Backup für den Fall, dass Sie die aktuelle Datei schrotten:

```
cd /home/pi
cp .config/openbox/lxde-rc.xml .config/openbox/lxde-rc.xml.original
```

Nun öffen Sie die Datei und suchen Sie die Sektion "Keybindings". Hier können Sie alle Shortcuts umdefinieren, löschen oder (defensiver) deaktivieren. Bei der Tastendefinition gilt übrigens:
S = Shift
C = Control
A = Alt
W = Super key (Win)

Der folgende Eintrag würde beispielsweise beim Drücken der Tastenkombination `[Strg]-[Shift]-|Alt]-[e]` den Editor `gedit` aufrufen.

```
    <keybind key="A-C-S-e">
      <action name="Execute">
        <command>gedit</command>
      </action>
    </keybind>
```

Zum Deaktivieren der jeweiligen Taste ändern Sie einfach die Kommando-Sektion, indem Sie die entsprechende Kommandodofinition durch

```
<command>false</command>
```

ersetzen. Aktiviert werden die Änderungen durch das Kommando

```
openbox --reconfigure
```

So können Sie auch Definition für Definition schrittweise anpassen und testen.

Browser konfigurieren

Der Browser "midori" muss natürlich auch an einigen Stellen abgespeckt werden, was aber relativ leicht über die Browser-Oberfläche selbt geht ("Einstellungen"). Dotz werden dann unerwünschte Features wie Menüleisten, Plugins etc. abgeschaltet. Für Puristen kommt auch noch das Bearbeiten der Konfigurationsdateien selbst unter `/home/pi/.config/midori/` infrage.

Bei Bedarf kann ein angepasstes Profil angelegt und mit dem Kommandozeilenargument `-c /Pfad/zum/Ordner` an Midori übergeben werden. Dazu kopieren Sie einfach das aktuelle Profil und bearbeiten dies dann

entsprechend.

Um externe Seiten zu blockieren, können Sie einen regulären Ausdruck als Blacklist verwenden. Um unerwünschte Websites zu blockieren, erzeugen Sie beispielsweise die Zeile:

```
-b 'youtube|youporn'
```

Durch Negieren des Ausdrucks können Sie auch eine Whitelist erzeugen:

```
-b '^(?!.*?(gmx|gmail|google)).*'
```

Alle Links auße der aufgeführten landen auf einer Fehlerseite. Alle Bilder und andere Dateien werden nicht geladen.

Einige Websites verhalten sich seltsam, wenn sie nicht einen Browser wie Chrome, Firefox, IE, usw. erkennen. Falls Sie also nicht nur lokale Seiten anzeigen, sondern auch auf andere Websites verlinken (z. B. auch in einem `iframe`), können Sie den Browser-Namen auch ändern: Preferences → Network → Identify As. Falls Ihnen da kein schöner Name einfällt, schauen Sie doch mal in diese Liste.

Neben den Pfeiltasten kann man auch über die guten alten `vi`-Tasten navigieren: h=links, l=rechts, j=runter, k=rauf. Standardmäßig initiiert die rechte Maustaste Gestensteuerung. Sie können jedoch den Butten wechseln (beispielsweise auf die mittlere Maustaste) indem Sie eine versteckte Option erzeugen. Erstellen Sie dazu eine Textdatei `/home/pi/.config/midori/extensions/libmouse-gestures.so/config` und geben Sie den folgenden Text ein:

```
[settings]
  button=2
```

Ab Midori 0.5.0 können individuelle Maus-Gesten frei in der Datei `/home/pi/.config/midori/extensions/libmouse-gestures.so/gestures` konfiguriert werden. Der Aufruf "midori --help-execute" liefert eine Liste der verfügbaren Aktionen, die auf der linken Seite des Gleichheitszeichens gesetzt werden. Auf der rechten Seite geht eine Folge von Anweisungen wie (W)est, (E)ast, (N)orth, (S)outh oder Kombinationen der Buchstaben mit einem Semikolon als Trennzeichen, z. B.:

```
[gestures]
  Quit=W;E;
  TabPrevious=SW;
  TabNext=SE;
```

Auch hier können Sie individuelle Gesten setzen - oder auch löschen.

Weitere Dateien mit Konfigurationsdaten sind:

```
/home/pi/.local/share/applications/defaults.list  # Zuordnung der MIME-Typen zu Plugins
/home/pi/.local/share/midori/styles               # Default-Styles
/home/pi/.local/share/midori/scripts              # User-Scripts (vorzugsweise leer)
```

Konfigurationsdateien sichern

Mit der Bearbeitung der Midori-Konfiguration ist hoffentlich aller Unsinn blockiert, den jemand mit dem Browser anstelle könnte. Trotzdem kann auch hier noch etwas mehr getan werden. Wenn ale Konfigurationsdateien bearbeitet sind, machen Sie ein Backup davon. Das geht ganz einfach für den kompletten Verzeichnisbaum mit dem `tar`-Kommando:

```
cd /home/pi
# Archiv anlegen
tar cvpf midori-config.tar .config/midori/*
```

Beim Booten des Systems kopieren Sie das Backup wieder über die Midori-Konfiguration und können sicher sein, dass alles wieder wie neu ist. Dazu packen Sie die folgende Kommandos in die oben schon erwähnte Datei `/home/pi/.bash_profile` vor den `startx` und den anderen Kommandos:

```
cd /home/pi
# Archiv auspacken
tar xpf midori-config.tar
```

Tipp 1: Im Prinzip können Sie auf die gleiche Weise auch mit anderen Konfigurationsdateien und anderen Dateien, die unverändert bleiben sollen, verfahren - z. B. auch der Datei `coockies.txt`.

Tipp 2: Wenn's noch einen Tick sicherer werden soll, könnten Sie das Regenerieren der Konfigurationsdateien auch regelmäßig per `crontab`-Kommando anstossen.

Das gesamte System abhärten

Um weiters Ungemach durch Fehlbedienung und Hacker zu vermeiden, sollten Sie alles deinstallieren, was nicht unbedingt gebraucht wird. Dieser Schritt ist leider mit nicht unbeträchtlichen Aufwand verbunden, denn es reicht nicht, nur ein unerwünschtes Kommando zu löschen (ausser bei den einfachen Shell-Kommandos), sondern es muss das jeweilige Paket deinstalliert werden. Das bedeutet, dass Sie für jedes Paket, dass nicht gebraucht wird, das Kommando

```
apt-get -y purge <Paketname>
```

aufgerufen werden muss. Sinnvollerweise würden Sie sich zunächst eine Datei mit den Paketnamen der zu löschenden Pakete anlegen (ggf. mit Hilfe von `apt-cache search ...` nach den richtigen Namen suchen). Die Datei `delpaket` kann dann per Script verarbeitet werden:

```
#!/bin/bash
while read PAKET
do
  apt-get -y purge $PAKET
done < /home/pi/delpaket
```

So eine Datei hat den Vorteil, dass Sie den Vorgang auch ratz-fatz erledigt haben, wenn Sie später mal ein ganz frisches System härten müssen.

Refresh von HTML-Seiten

Sie haben oben beim Script schon gesehen, dass der Browser mit dem Kommando `midori -e Reload` auf der Kommandozeile dazu veranlasst werden kann, den angezeigten Inhalt aufzufrischen. Oft soll der Refresh aber von der HTML-Datei, also der angezeigten Webseite aus geschehen. Dafür gibt es diverse Möglichkeiten. Für alle aber gilt, dass bei gleichem Namen der HTML-Datei und gleicher Größe, der Browser den Refresh aus seinem Cache und nicht von der ggf. geänderten Datei vollzieht. Daher muss in der Browser-Konfiguration (Einstellungen) der Browser-Cache abgeschaltet sein.

Refresh über META-Statement

Benutzt wird diese Angabe in der Praxis oft für das automatische Weiterleiten zu einer neuen Adresse. Beim Kiosk-System geht es aber nur um das erneute Laden einer Webseite. Mit dem Eintrag `<meta http-equiv="refresh" content="...">` im Header einer HTML-Datei veranlassen Sie die Weiterleitung zu einer anderen Adresse. DieContent-Angabe besteht aus zwei durch Semikolon getrennten Angaben, der Wartezeit bis zum nächsten refresh und der URL der zu ladenden Seite. Bei einer Wartezeit von 0 wird die angegebene URL sofort geladen. Beispiel:

```
<head>
  ...
<meta http-equiv="expires" content="0">
<meta http-equiv="pragma" content="no-cache">
<meta http-equiv="refresh" content="10; URL=http://www.netzmafia.de/">
  ...
</head>
```

Soll nur ein und dieselbe Seite neu geladen werden, entfällt die URL:

```
<head>
  ...
<meta http-equiv="expires" content="0">
<meta http-equiv="pragma" content="no-cache">
<meta http-equiv="refresh" content="10">
  ...
</head>
```

Die oberen beiden Zeilen ("expires" bzw. "pragma") verhindern, dass der Browser den Inhalt nich vom Originalserver, sondern aus seinem Cache nachlädt. Sie sollten auch bei den folgenden Altermativen mit Javascript und PHP vorhanden sein.

Refresh per Javascript

In Javascript wird die Timeout-Methode verwendet, um die Wartezeit zu realisieren. Zu beachten ist hier, dass die Zeitangabe im Millisekunden erfolgt. Der entsprechende Einzeile kann in einem größeren Javascript enthalten sein, z. B.:

```
setTimeout("location.reload(true);", 10000);
```

Am einfachsten ruft man das Javascript im Body-Tag auf, z. B.:

```
<body onload="JavaScript:setTimeout('location.reload(true);', 10000);">
```

Man kann das Ganze auch etwas redundanter und übersichtlicher mit einer Funktion realisieren, die dann ggf. noch weitere Anweisungen enthalten kann.

```
<head>
  ...
<meta http-equiv="expires" content="0">
<meta http-equiv="pragma" content="no-cache">
  ...
<script type="text/JavaScript">
function TimedRefresh( tim )
  {
  setTimeout("location.reload(true);", tim);
  // eventuell weitere Anweisungen
  }
</script>
</head>
```

Der Aufruf im Body-Tag lautet dann:

```
<body onload="JavaScript:TimedRefresh(10000);">
```

Refresh per PHP/Perl

Was im Folgenden für PHP beschrieben wird, funktioniert auch in jeder andern Programmiersprache. Einzige Bedingung ist, dass die Einträge im HTTP-Header erfolgen müssen. In Perl würde man die Headerzeilen einfach vor der Zeile "Content-Type" unterbringen und die Wartezeit über die Alarm-Funktion realisieren. In PHP gibt es die Funktion `header()`, die alle Arbeit erledigt. Vor dem Aufruf von `header()` dar jedoch noch nichts in Richtung Browser gesendet worden sein, kein Leerzeichen, kein Newline, nix.

```
$webpage = $_SERVER['PHP_SELF'];
$duration = "10";
header("Refresh: $duration; url=$webpage");
```

Fertiges System: FullPageOS

(Quelle: Raspberry Pi Geek, Heft 1/2016)
FullPageOS – ein Kiosksystem für den RasPi. Egal, ob Geldautomaten, Informationsterminals oder Fahrkartenautomaten: Öffentlich zugängliche Computer brauchen eine ordentliche Absicherung. Dazu müssten Sie bei einer gängigen Linux-Distribution an zahlreichen Stellschräubchen drehen. Die generische Bezeichnung solcher öffentlichen Terminals lautet "Kiosksysteme" – ein solches bietet die Distribution FullPageOS. Es macht aus einem Raspberry Pi ein abgeschottetes Internet-Terminal, mit dem sich nur vordefinierte Webseiten öffnen lassen. Das System startet nach dem Booten den Chromium-Browser im bildschirmfüllenden Modus und einer vordefinierten Webseite. Zugriff aufs System oder andere Anwendungen erlaubt die Distribution über das Display nicht. Einige Dinge sind aber noch nicht gesichert, darunter die SysRq-Taste.

Infos unter: https://guysoft.wordpress.com/2015/10/17/fullpageos-out-of-the-box-kiosk-mode-for-the-raspberrypi/
Download unter: https://github.com/guysoft/FullPageOS

Letzte Aktualisierung: 06/13/2024 11:12:33