Linux Usage Tips

Some useful things to know in Linux.

DNS Privacy

Dumped

dns_privacy.txt

```
public/free dns:
dns.quad9.net
9.9.9.9
149.112.112.112
2620:fe::fe
2620:fe::9

Recursive DNS Server Addresses and Features - Service based
configuration:

Recommended: Malware Blocking, DNSSEC Validation (this is the most
typical configuration)

IPv4
9.9.9.9
149.112.112.112

IPv6
2620:fe::fe
2620:fe::9

HTTPS
https://dns.quad9.net/dns-query

TLS
tls://dns.quad9.net

Secured w/ECS: Malware blocking, DNSSEC Validation, ECS enabled

IPv4
9.9.9.11
149.112.112.11

IPv6
2620:fe::11
2620:fe::fe:11
```

```
HTTPS
https://dns11.quad9.net/dns-query

TLS
tls://dns11.quad9.net

Unsecured: No Malware blocking, no DNSSEC validation (for experts
only!)

IPv4
9.9.9.10
149.112.112.10

IPv6
2620:fe::10
2620:fe::fe:10

HTTPS
https://dns10.quad9.net/dns-query

TLS
tls://dns10.quad9.net
```

Code Debugging

Note on using valgrind...

```
$ valgrind --leak-check=full --track-origins=yes --log-file=valgrind.rpt \
    ./my1asm85 ../my1asm85/asm/monitor85.asm

$ valgrind --tool=memcheck --leak-check=full --track-origins=yes -s \
    ./my1asm85 ../my1asm85/asm/monitor85.asm
```

User Admin

To disable user login

```
# passwd -l username
```

To re-enable user login

```
# passwd -u username
```

OR,

To disable user login

```
# usermod -L -e 1 username
```

To re-enable user login

```
# usermod -U -e "" username
```

I prefer/use the first one.

Avahi/mDNS Stuff

List all hosts in *.local

```
$ avahi-browse -atr | grep hostname | sort -u
```

Terminal Display

To display extended ASCII (which is a non-standard), use `Western-IBM850` encoding (instead of the currently common UTF-8). The extended ASCII is actually great in drawing boxes in text mode.

Manipulating PDF

To merge multiple PDFs (using pdftk):

```
pdftk $(ls *.pdf | sort) cat output ../output.pdf
```

Use xournal (or, maybe, xournalpp) to mark/sign - used export PDF to recreate PDF. If the created PDF is big (tends to be), import from LibreOffice Draw and re-export PDF.

Cleanup Stuff

Just found out about `/proc/sys/vm/drop_caches` kernel interface today. Need to look at this some time...

```
* clear page cache {flag:0x01}
# echo 1 > /proc/sys/vm/drop_caches
* clear dentries and inodes {flag:0x02}
# echo 2 > /proc/sys/vm/drop_caches
* can request simultaneously {flag:0x02|0x01}
# echo 3 > /proc/sys/vm/drop_caches
```

Setup User Development Space (Isolated?)

Got this info from somewhere online - I just put it into a script.

Note: *Create access to an IPad... requires ifuse and libimobiledevice-utils packages (tested on Devuan).*

setup_dev_env

```
#!/bin/bash

# setup_dev_env
# - got this info from internet, i just put it in a script
# - should source this file :p

WORK_PATH=`pwd`
[ -d "$1" ] && WORK_PATH=`cd $1;pwd`
echo "-- Setup environment in ${WORK_PATH}"

MAKE_PATH=0
add_path()
{
    [ -d "$1" ] && PATH="$1":$PATH && MAKE_PATH=1
}
add_path ${WORK_PATH}/bin
add_path ${WORK_PATH}/sbin
add_path ${WORK_PATH}/usr/bin
add_path ${WORK_PATH}/usr/sbin
[ $MAKE_PATH -eq 0 ] &&
    echo "** No binary path found!" && exit 1
export PATH

MAKE_PATH=0
add_ldlib_path()
{
    if [ -d "$1" ] ; then
        [ -z "$LD_LIBRARY_PATH" ] &&
            LD_LIBRARY_PATH="$1" ||
LD_LIBRARY_PATH="$1":$LD_LIBRARY_PATH
        MAKE_PATH=1
    fi
}
add_ldlib_path ${WORK_PATH}/lib
add_ldlib_path ${WORK_PATH}/usr/lib
[ $MAKE_PATH -ne 0 ] && export LD_LIBRARY_PATH

MAKE_PATH=0
add_c_path()
{
    if [ -d "$1" ] ; then
```

```
        [ -z "$CPATH" ] &&
            CPATH="$1" || CPATH="$1":$CPATH
        MAKE_PATH=1
    fi
}
add_c_path ${WORK_PATH}/include
add_c_path ${WORK_PATH}/usr/include
[ $MAKE_PATH -ne 0 ] && export CPATH

MAKE_PATH=0
add_man_path()
{
    if [ -d "$1" ] ; then
        [ -z "$MANPATH" ] &&
            MANPATH="$1" || MANPATH="$1":$MANPATH
        MAKE_PATH=1
    fi
}
add_man_path ${WORK_PATH}/share/man
add_man_path ${WORK_PATH}/usr/share/man
[ $MAKE_PATH -ne 0 ] && export MANPATH

MAKE_PATH=0
add_pkgcfg_path()
{
    if [ -d "$1" ] ; then
        [ -z "$PKG_CONFIG_PATH" ] &&
            PKG_CONFIG_PATH="$1" ||
PKG_CONFIG_PATH="$1":$PKG_CONFIG_PATH
        MAKE_PATH=1
    fi
}
add_pkgcfg_path ${WORK_PATH}/lib/pkgconfig
add_pkgcfg_path ${WORK_PATH}/usr/lib/pkgconfig
[ $MAKE_PATH -ne 0 ] && export PKG_CONFIG_PATH
```

Support for RTL8723DE WiFi Module

I have Slackware 14.2 on my HP laptop, which has an RTL8723DE wifi module hardware. This module is still not supported in the mainstream kernel, but the driver is available at https://github.com/lwfinger/rtlwifi_new.git.

- get source, go extended

```
$ git clone https://github.com/lwfinger/rtlwifi_new.git
$ cd rtlwifi_new
$ git checkout extended
```

- to install (as root)

```
$ make install
$ modprobe rtl8723de
```

- to uninstall module (e.g. for kernel update)

```
$ rmmod rtl8723de
$ make uninstall
```

- to get better signal strength, create file '/etc/modprobe.d/rtl8723de.conf' with:

```
options rtl8723de ant_sel=2
```

Update20200824 *That repo is no longer available. Using rtw88 instead (module is now named rtw88_8723de). Just to note that the rtw88 code is now in mainline kernel starting 5.2*

KVM Stuff

Somehow, `/dev/kvm` is missing?? So, just create this udev rule:

65-kvm.rules

```
KERNEL=="kvm", GROUP="users", MODE="0660"
KERNEL=="vhost-net", GROUP="users", MODE="0660"
```

Text-manipulation

I mostly use `sed` / `grep` / `cut` for this. But I have found the need to use `tr` at times.

For example, to filter out non-ASCII characters

```
tr -cd '\000-\177'
```

Removing ldlinux.sys

When installing using extlinux, the file ldlinux.sys is created and cannot be removed even by root! The command `chattr` can be used to remove the flag that protects that particular file

```
# chattr -i ldlinux.sys
```

To check/confirm, run

```
# lsattr ldlinux.sys
```

After that, the file can be removed by root as usual.

GTK3 Stuff

Running my GUI code compiled with GTK3 on Artix causes a runtime warning message

```
dbind-WARNING **: 11:12:11.208: Couldn't register with accessibility bus:
Did not receive a reply. Possible causes include: the remote application did
not send a reply, the message bus security policy blocked the reply, the
reply timeout expired, or the network connection was broken.
```

I found this simple fix somewhere in a forum - simply create an environment variable

```
export NO_AT_BRIDGE=1
```

To check screen dpi

```
$ xdpyinfo
$ xrdb -query
```

Optimize hard disk using hdparm

Note to self: CHECK THIS OUT! For example, on T23 laptop

```
$ hdparm -q -d1 -c3 -W1 -u1 -m 16 /dev/hda
$ hdparm -q -d1 -c3 -X66 /dev/hdc
```

Adding fonts in Linux

```
1. xorg.conf, "mkfontscale", "mkfontdir", "ttmkfdir" => old school, nobody
needs them?
2. /etc/fonts/*, ~/.fonts.conf, fc-cache, fc-list => the way to go
```

To strip binary/library files

```
$ strip --strip-debug /lib/*
$ strip --strip-unneeded /{,s}bin/*
```

Create a patch file

```
$ diff -Naur [file1] [file2] > file.patch
**Note**
  %%{{{@@ -X,Y +J,K @@}}}%% is a hunk where diff is
    X & J - starting line number
    Y & K - line counts
```

Script to create multiple users

create_users

```
#!/bin/bash

USERLIST="$1"

[ ! -f "$USERLIST" ] &&
    echo "Cannot access user list '$USERLIST'! Abort!" && exit 1

# assume format is user:pass
for userpass in `cat $USERLIST` ; do
    user=`echo $userpass | sed 's/\([^:]\):.*/\1/'`
    pass=`echo $userpass | sed 's/.*:\(.*\)/\1/'`
    what=`echo $userpass | sed 's/^\(#\)[^#]*/\1/'`
    [ "$what" == "#" ] && continue
    what=`cat /etc/passwd | grep "$user"`
    [ "$what" != "" ] && continue
    echo "User:[$user],Pass:[$pass]"
    echo "${user}:${pass}::100:User Account:/home/${user}:/bin/bash"
done
```

NTP (System Date/Time)

By default, date/time are set manually: e.g. as root,

```
ntpdate pool.ntp.org
```

If ntpdate is not available (being deprecated?), use ntp daemon:

```
ntpd -gq
```

(May have to stop running ntp daemon)

For Slackware, a startup script for NTP daemon is available `/etc/rc.d/rc.ntpd`, but do not forget

to modify `/etc/ntp.conf` and uncomment (or add custom) NTP server information. To query, use

```
ntpq -p
```

To still use `ntpdate` while NTP daemon is running, use the `-d` switch,

```
ntpdate -d pool.ntp.org
```

Useful /proc Interface

To display active pertitions,

```
cat /proc/partitions
```

To display cpu information,

```
cat /proc/cpuinfo
```

To check for Intel-VT virtualization features,

```
grep --color vmx /proc/cpuinfo
```

To check for AMD-V virtualization features,

```
grep --color svm /proc/cpuinfo
```

***Note**: Some BIOS may disable this feature by default. Also, old Linux kernel may not support it either.*

To display memory information,

```
cat /proc/meminfo
```

To check shared libraries used by a program

```
cat /proc/<proc_id>/maps
```

Playing with LDAP

I want to have LDAP-based central authentication for my two new Slackware machines used for my GMC project. Things to do:

1. get LDAP server installed and running
    - Slackware only have openldap-client by default
    - I'm referring to here to do this
2. figure out how to setup the client side
    - I may need nss_ldap only - will check into this later

SSH X Forwarding

On server, edit `/etc/ssh/sshd_config`

```
AllowTcpForwarding yes
X11Forwarding yes
X11DisplayOffset 10
X11UseLocalhost yes
```

On client, edit `/etc/ssh/ssh_config`

```
ForwardAgent yes
ForwardX11 yes
```

To connect,

```
# allow remote access for display
ssh -Y user@host
export DISPLAY=localhost:10.0
# then, run any x-program
```

Display Access Control

Only the user on the main console gets control of display protocol. If we do `su`, the root user cannot use any GUI.

To disable and enable access control

```
$ xhost +
$ xhost -
```

Find and set DISPLAY value accordingly

```
$ echo $DISPLAY
$ export DISPLAY=:0.0
```

Remote VNC (Remote Desktop)

This enables us to have GUI access on a remote machine. I use screen because I want to keep the running terminal 'alive'.

Use virtual terminal, run `screen` on the remote machine

```
$ screen -S VNC
```

Start VNC server on a display port

```
$ vncserver :23
```

Note *~/.vnc/xstartup will be executed*

To stop the VNC server

```
$ vncserver -kill :23
```

Detach the virtual terminal by hitting *<CTRL+A>-<D>*. To resume the virtual terminal from any console

```
$ screen -r VNC
```

To tunnel through SSH, start an ssh session (with port forwarding [5900+DP] and going background)

```
ssh -L 5923:localhost:5923 -N -f user@remotehost
```

Connect as if the server is on localhost

```
vncviewer localhost:23
```

Note: *Desktop response becomes too slow for me! Not using this...*

Console Multi-Tasking

On Linux, multi-tasking is also available on console. To suspend a task, hit `ctrl+z`

To list all jobs **duh!**

```
$ jobs
```

To resume

```
$ fg %<num>
```

VirtualBox Stuffs

Rebuild VirtualBox kernel module

After a kernel upgrade... do a

```
# /etc/init.d/vboxdrv setup
```

Virtual Serial Port

We can actually use minicom to connect...

```
minicom -D unix\#/tmp/xxx
```

where /tmp/xxx is the host pipe 😎

Convert Image to Disk

```
VBoxManage convertfromraw -format VDI <file.img> <file.vdi>
```

Compact Disk Image

```
VBoxManage modifymedium --compact <file.vdi>
```

Create Disk Image

```
VBoxManage createmedium disk --filename <file.vdi> --size <megabytes>
```

Video4Linux Stuffs

```
v4l2-ctl --list-devices
v4l2-ctl -d 0 --list-formats-ext
v4l2-ctl -d 0 --list-ctrls

v4l2-ctl --device=0 --set-ctrl=?

v4l2-ctl -d 0 --set-fmt-video=width=1920,height=1080,pixelformat=YUYV
```

Mounting FreeBSD Partition (Slice)

Note *At the moment, only read-only access (no write permission)!*

```
# mount -t ufs -o ufstype=ufs2 /dev/<partition> <mount-path>
```

Backing up dokuwiki pages on github

dokuwiki2github.sh

```
#!/bin/bash

DOKUWIKI=${DOKUWIKI:="$HOME/public_html/dokuwiki"}
WIKIPAGE=${WIKIPAGE:="$DOKUWIKI/data/pages"}

cd ${WIKIPAGE} && git add . && git add -u &&
    git commit -a -m "Content update `date +'%H:%M %d/%m/%Y %Z'`" &&
```

```
    git push origin master

# setup a cron job for hourly update:
#
# 0 * * * * dokuwiki2github.sh
```

Linux kernel patch for CH341 USB2Serial

Tested this on Slackware 14.2.

Using patch found here, I modified it a bit for latest Linux kernel used in Slackware 14.2 (4.4.190)

To create this patch, I copied out `ch341.c` from kernel source to my user path. Make another copy as `ch341_patched.c`. Modify as required. Do,

```
$ diff -u ch341.c ch341_patched.c > linux_4.4.190_ch341.patch
```

I do not know if this last step is needed, but I modified the patch so that the first two lines have the same file name.

linux_4.4.190_ch341.patch

```
--- ch341.c   2019-10-24 15:02:10.039646991 +0800
+++ ch341.c   2019-10-24 15:06:14.244416258 +0800
@@ -358,6 +358,7 @@
    struct ch341_private *priv = usb_get_serial_port_data(port);
    unsigned baud_rate;
    unsigned long flags;
+ unsigned int par_flags;

    baud_rate = tty_get_baud_rate(tty);

@@ -371,6 +372,30 @@
     * (cflag & PARENB) : parity {NONE, EVEN, ODD}
     * (cflag & CSTOPB) : stop bits [1, 2]
     */
+ /* CH340 doesn't appear to support variable stop bits or data bits */
+ if (C_PARENB(tty)) {
+     if (C_PARODD(tty)) {
+         if (tty->termios.c_cflag & CMSPAR) {
+             dev_dbg(&port->dev, "parity = mark\n");
+             par_flags = 0xeb;
+         } else {
+             dev_dbg(&port->dev, "parity = odd\n");
+             par_flags = 0xcb;
+         }
+     } else {
+         if (tty->termios.c_cflag & CMSPAR) {
```

```
+             dev_dbg(&port->dev, "parity = space\n");
+             par_flags = 0xfb;
+         } else {
+             dev_dbg(&port->dev, "parity = even\n");
+             par_flags = 0xdb;
+         }
+     }
+ } else {
+     dev_dbg(&port->dev, "parity = none\n");
+ par_flags = 0xc3;
+ }
+ ch341_control_out(port->serial->dev, 0x9a, 0x2518, par_flags);

    spin_lock_irqsave(&priv->lock, flags);
    if (C_BAUD(tty) == B0)
```

Steps (as root) in terminal:

1. get to location

```
$ cd /usr/src/linux/drivers/usb/serial
```

2. patch `ch341.c`

```
$ patch < /path/to/linux_4.4.190_ch341.patch
```

   - `path/to` is the path where you save the above patch

3. get to top kernel source path

```
$ cd /usr/src/linux
```

4. compile module

```
$ make M=drivers/usb/serial modules
```

5. (OPTIONAL) remove previously loaded module

```
$ rmmod ch341
```

6. 'install' the module

```
$ cp drivers/usb/serial/ch341.ko /lib/modules/`uname -
r`/kernel/drivers/usb/serial/
```