# 1. linux

## sudo

Task: Avoid typing sudo each and every time

Note that this is not recommended until and unless you are an expert and aware of what you are typing:  
$ sudo -i

ftp ftp://username:password@my.domain.com

## modify mac address

/etc/init.d/networking stop

Or ifconfig eth0 down

ifconfig eth0 hw ether 00:60:16:75:12:02

/etc/init.d/networking start

Or ifconfig eth0 up

## Set ip

sudo ifconfig eth0 10.62.34.52/24

ifconfig eth0 netmask 255.255.255.224

ifconfig eth0 broadcast 172.16.25.63

ifconfig eth0 172.16.25.125 netmask 255.255.255.224 broadcast 172.16.25.63

ifconfig eth0 mtu 1000

sudo route add default gw 10.62.34.1

add dns

vim /etc/resolv.conf.

Add – nameserver 10.32.102.224

Save the file.

## mount

On PXE 36

sudo service portmap start

sudo service nfs-kernel-server start

sudo service nfs-common start

On SP

mount -o port=2049,nolock,proto=tcp -t nfs 192.168.1.1:/home/share/guoq3 /mnt/guoq3/

## format code

use “astyle” command to format each of your new files into standard format:

**astyle --style=ansi -sbCKm0pD plugin.cpp**

<http://astyle.sourceforge.net/>

## show port

netstate –a

## tar.gz

<http://www.howtogeek.com/248780/how-to-compress-and-extract-files-using-the-tar-command-on-linux/>

tar -czvf name-of-archive.tar.gz /path/to/directory-or-file

* -c: **C**reate an archive.
* -z: Compress the archive with g**z**ip.
* -v: Display progress in the terminal while creating the archive, also known as “**v**erbose” mode. The v is always optional in these commands, but it’s helpful.
* -f: Allows you to specify the **f**ilename of the archive.

tar.xz

tar xf file.tar.zx

tar xf file.tar.gz

## get diff

Once you have the source trees, e.g.

diff -ENwbur repos1/ repos2/

Even better

diff -ENwbur repos1/ repos2/ | kompare -o -

and have a crack at it in a good gui tool :)

* -Ewb ignore the bulk of whitespace changes
* -N detect new files
* -u unified
* -r recurse

## apt-get

/var/lib/apt/lists/

/etc/apt/ sources.list

/var/cache/apt/archives

## Network Cards info

1. lspci command : List all PCI devices.
2. lshw command : List all hardware.
3. dmidecode command : List all hardware data from BIOS.
4. ifconfig command : Outdated network config utility.
5. ip command : Recommended new network config utility.

<http://www.cyberciti.biz/faq/linux-list-network-cards-command/>

## adduser

adduser username

Set password prompts:

Enter new UNIX password:

Retype new UNIX password:

passwd: password updated successfully

User information prompts:

Changing the user information for username

Enter the new value, or press ENTER for the default

Full Name []:

Room Number []:

Work Phone []:

Home Phone []:

Other []:

Is the information correct? [Y/n]

Use the usermod command to add the user to the sudo group.

usermod -aG sudo username

## count all lines of folders

find . -name '\*.c' | xargs wc -l

( find ./ -name '\*.c' -print0 | xargs -0 cat ) | wc -l

works on names with spaces, only outputs one number.

## Samba server

First, make sure that you’ve [installed Samba server](http://www.howtogeek.com/howto/ubuntu/install-samba-server-on-ubuntu/).

**Install samba:**

sudo apt-get install samba

**Set a Samba password:**

sudo smbpasswd -a USERNAME.

This command will generate a prompt for a password (substitute USERNAME with your username).

**Modify smb.conf:**

To share the home directories, open up smb.conf with the following command:

sudo vim /etc/samba/smb.conf

Find this section of the file, and make it match the following:

#======================= Share Definitions =======================

# Un-comment the following (and tweak the other settings below to suit)  
# to enable the default home directory shares. This will share each  
# user’s home directory as \\server\username  
[homes]  
comment = Home Directories  
browseable = yes

# By default, \\server\username shares can be connected to by anyone  
# with access to the samba server. Un-comment the following parameter  
# to make sure that only “username” can connect to \\server\username  
valid users = %S

# By default, the home directories are exported read-only. Change next  
# parameter to ‘yes’ if you want to be able to write to them.  
writable = yes

Now you should be able to map a drive on windows using the following share format:

\\ubuntumachine\username

Add share folder:

[<floder name>]

path = <folder path>

valid users = username1, username2

read only = no

browsable = yes

public = yes

writable = yes

## adduser

adduser username

Use the usermod command to add the user to the sudo group.

* usermod -aG sudo username

## make

-g –ggdb3

## Apply patch of a diff file

1 create a diff file.

2 patch -p[num] < patchfile

patch [options] originalfile patchfile

<http://www.thegeekstuff.com/2014/12/patch-command-examples>

## readelf

## ip\_forward

Check if IP Forwarding is enabled

We have to query the **sysctl kernel**value **net.ipv4.ip\_forward** to see if forwarding is enabled or not: Using **sysctl**:

sysctl net.ipv4.ip\_forward

net.ipv4.ip\_forward = 0

cat /proc/sys/net/ipv4/ip\_forward

## linux tools

ccache - <https://ccache.samba.org/> a fast C/C++ compiler cache

**CFLAGS – remove O2, add -d dgdb**

## dmidecode

**dmi table decoder**

**dmidecode** is a tool for dumping a computer's DMI (some say SMBIOS ) table contents in a human-readable format. This table contains a description of the system's hardware components, as well as other useful pieces of information such as serial numbers and BIOSrevision.

[**https://linux.die.net/man/8/dmidecode**](https://linux.die.net/man/8/dmidecode)

## iPerf

iPerf3 is a tool for active measurements of the maximum achievable bandwidth on IP networks. It supports tuning of various parameters related to timing, buffers and protocols (TCP, UDP, SCTP with IPv4 and IPv6). For each test it reports the bandwidth, loss, and other parameters. This is a new implementation that shares no code with the original iPerf and also is not backwards compatible. iPerf was orginally developed by [NLANR/DAST](https://iperf.fr/contact.php#authors). iPerf3 is principally developed by [ESnet](https://www.es.net/" \t "_blank) / [Lawrence Berkeley National Laboratory](https://www.lbl.gov/). It is released under a three-clause [BSD license](https://en.wikipedia.org/wiki/BSD_licenses).

[**https://iperf.fr/**](https://iperf.fr/)

**packetgen**

## ps

**ps afx**

## sed

insert one line in the specific line:

sed -i '8i8 This is Line 8' FILE

find /path/to/files -type f -exec sed -i 's/oldstring/new string/g' {} \;

grep -rl matchstring somedir/ | xargs sed -i 's/string1/string2/g'

**insert operation**

#insert # at the beginning of file.txt

$ sed 's/^/#/' file.txt

#insert # at the beginning of file.txt

$ sed 's/^/ /' file.txt

sed 'line-num-1,line-num-2s/^/ /' intel\_eth\_device\_errors.h

# replace string start with “ with ;

sed '79,133s/".\*/;/g' intel\_eth\_device\_errors.h

**sed -i '23,$s/^/\t/' filename**

To append after the pattern: (-i is for inplace replace). line1 and line2 are the lines you want to append(or prepend)

sed -i '/pattern/a \

line1 \

line2' inputfile

To prepend the lines before:

sed -i '/pattern/i \

line1 \

line2' inputfile

## timezone

                $ rm /etc/localtime

                $ cd /etc

$ ln -s /usr/share/zoneinfo/Asia/Shanghai localtime

## Screen

<https://www.gnu.org/software/screen/manual/screen.html>

## CRLF&LF

Display CRLF as ^M:

:e ++ff=unix

Substitute CRLF for LF:

:setlocal ff=unix  
:w  
:e

## mkdocs

install python pip

$ wget https://bootstrap.pypa.io/get-pip.py

$ python get-pip.py

$ pip install mkdocs

$ pip install mkdocs-material

$ mkdocs –version

$ mkdocs new my-project

$ cd my-project

$ mkdocs serve

INFO - Building documentation...

INFO - Cleaning site directory

[I 160402 15:50:43 server:271] Serving on http://127.0.0.1:8000

[I 160402 15:50:43 handlers:58] Start watching changes

[I 160402 15:50:43 handlers:60] Start detecting changes

$ mkdocs build

**How to update Chameleon Docs into GitHub：**

Step1: Install tools at your environment (once and for all)

* “python” version 2.7
* “pip” ---- suggest use below commands
  1. sudo wget <https://bootstrap.pypa.io/get-pip.py> --no-check-certificat
  2. sudo python ./get-pip.py
* “mkdocs”
  1. sudo pip install --upgrade mkdocs
  2. sudo pip install mkdocs-material

Step2: Get codes

* Repo: ssh://git@eos2git.cec.lab.emc.com/Chameleon/chameleon-kit.git

Step3: Change code：

* Suggest to use markdownpad to modify your code.
* Add the new document to docs/ folder
  + 1. *cd docs/*
    2. *git add <newpage.md>*
* Edit the mkdocs.yml file

*site\_name: Chameleon Docs*

*pages:*

*- Home: index.md*

*- About: README.md*

*- <New-Page-Title>: <newpage.md>*

*theme: 'material'*

Step4: Verify at Local:

* start http server: mkdocs serve -a *10.32.136.25:8000* -f */home/wangj69*/chameleon-docs/mkdocs.yml
* access it with Chrome: [*http://10.32.136.25:8000*](http://10.32.136.25:8000)

## tree

tree --charset ASCII

## history timestamps

[How to see time stamps in bash history](http://askubuntu.com/questions/391082/how-to-see-time-stamps-in-bash-history)

export HISTTIMEFORMAT="%d/%m/%y %T "

or

echo 'export HISTTIMEFORMAT="%d/%m/%y %T "' >> ~/.bashrc

source ~/.bashrc

## ssh tunnel

ssh -f -N -D <1080 user@remote-server.com>

ssh -fN -D :1080 xxx@xxx

export all\_proxy=socks5://localhost:1080

## send message

$sudo wall -n hi

$sudo echo "Let's go have lunch... ok?" > /dev/pts/4

$write username tty

**mesg** [**y**|**n**]

**Mesg** controls the access to your terminal by others. It's typically used to allow or disallow other users to write to your terminal (see [**write**](https://linux.die.net/man/1/write)(1)).

## shell script

for i in {1..1000000}; do echo -n "$i:"; wget http://www.intel.com/cd/edesign/library/asmo-na/eng/$i.htm; done;

## lxc

lxc image list

lxc image list ubuntu: | less

lxc launch ubuntu:16.04 first

lxc list

lxc info first

lxc config show first

free -m

lxc exec first -- free -m

lxc config set first limits.memory 64MB

lxc exec first -- free -m

lxc exec first -- apt-get update

lxc exec first -- apt-get dist-upgrade -y

lxc exec first -- apt-get autoremove --purge -y

lxc snapshot first clean

lxc exec first -- rm -Rf /etc /usr

lxc exec first -- bash

lxc restore first clean

lxc exec first -- bash

lxc publish first/clean --alias clean-ubuntu

lxc stop first

lxc delete first

lxc launch clean-ubuntu second

lxc file pull second/etc/hosts .

lxc file push hosts second/etc/hosts

lxc file pull second/var/log/syslog - | less

lxc delete --force second

lxc image list images: | less

lxc launch images:centos/7 third

lxc exec third -- cat /etc/redhat-release

lxc remote list

lxc remote add tryit 2001:470:b368:1070:216:3eff:fec6:67dc --password=safe

lxc list tryit:

lxc image list tryit:

lxc launch clean-ubuntu tryit:fourth

## markdown

mkdocs serve -a *10.32.136.25:8000* -f */home/wangj69*/chameleon-docs/mkdocs.yml

## recursive execute all files

find . -type f | xargs dos2unix

## ldd note

## Write a kernel module

## Compile

Archives - An archive (or static library) is simply a collection of object files stored as a single file. (An archive is roughly the equivalent of a Windows .LIB file.) When you provide anarchive to the linker, the linker searches the archive for the object files it needs,extracts them, and links them into your program much as if you had provided thoseobject files directly.

Create an archive

% ar cr libtest.a test1.o test2.o

Link .a

// wrong method

% gcc -o app -L. -ltest app.o

// right one

% gcc -o app app.o -L. –ltest

Shared library – step 1

compile the objects that will make up the library using the -fPIC option to the compiler

% gcc -c -fPIC test1.c

Step 2 - combine the object files into a shared library, like this:

% gcc -shared -fPIC -o libtest.so test1.o test2.o

-L. –ltest will first search .so then .a in the same path.

Use -static option to demand static archives, if can’t find .so, error return

% gcc -static -o app app.o -L. –ltest

The program only search /lib /usr/lib for .so, use –Wl,-rpath to specify .so path

% gcc -o app app.o -L. -ltest -Wl,-rpath,/usr/local/lib

Another solution to this problem is to set the LD\_LIBRARY\_PATH environment variable.

**Position-Independent Code (PIC)**

PIC stands for position-independent code. The functions in a shared library may be loaded at different

addresses in different programs, so the code in the shared object must not depend on the address (or

position) at which it is loaded. This consideration has no impact on you, as the programmer, except that

you must remember to use the -fPIC flag when compiling code that will be used in a shared library.

## xxd hexdump

## find

find a pattern in files and rename them

find ./inc-test/ -name "\*.orig" -exec rename -f 's/.h.orig/.h/' {} +

find . -name '\*-GHBAG-\*' -exec bash -c 'mv $0 ${0/GHBAG/stream-agg}' {} \;

mv ./report-GHBAG-1B ./report-stream-agg-1B

mv ./reoprt-GHBAG-0.5B ./report-stream-agg-0.5B

## The command-line history

Using the command history

Use the up and down key's to scroll through previously typed commands. Press [Enter] to execute them or use the left and right arrow keys to edit the command first. Also see *history* (below).

The history command

The*history*command can be used to list Bash's log of the commands you have typed:

This log is called the “history”. To access it type:

history n

This will only list the last *n* commands. Type “history” (without options) to see the the entire history list.

You can also type *!n* to execute command number n. Use *!!* to execute the last command you typed.

*!-n*will execute the command n times before (in other words *!-1* is equivalent to *!!*).

*!string*will execute the last command starting with that “string” and *!?string?* will execute the last command containing the word “string”. For example:

!cd

Will re-run the command that you last typed starting with “cd”.

*“ commandName**!\*”* will execute the “commandName” with any arguments you used on your last command. This maybe useful if you make a spelling mistake, for example. If you typed:

emasc /home/fred/mywork.java /tmp/testme.java

In an attempt to execute emacs on the above two files this will obviously fail. So what you can do is type:

emacs !\*

This will execute emacs with the arguments that you last typed on the command-line. In other words this is equivalent to typing:

emacs /home/fred/mywork.java /tmp/testme.java

Searching through the Command History ( **CTRL**-**R** )

Use the CTRL-R key to perform a “reverse-i-search”. For example, if you wanted to use the command you used the last time you used *snort*, you would type:

**CTRL**-**R** then type “snort”.

What you will see in the console window is:

(reverse-i-search)`':

After you have typed what you are looking for, use the **CTRL**-**R** key combination to scroll backward through the history.

Use **CTRL**-**R** repeatedly to find every reference to the string you've entered. Once you've found the command you're looking for, use [Enter] to execute it.

Alternatively, using the right or left arrow keys will place the command on an actual command-line so you can edit it.

## dmidecode

A tool to analyse BIOS data

## timezone

  $ rm /etc/localtime

                $ cd /etc

$ ln -s /usr/share/zoneinfo/Asia/Shanghai localtime

# 2. git

## misc

git rebase

integrate changes from one branch into another

<https://git-scm.com/book/en/v2/Git-Branching-Rebasing>

git blame

git-blame - Show what revision and author last modified each line of a file

git show <commit-id>

For generating the patches from the topmost commits from a specific sha1 hash:

git format-patch -<n> <SHA1>

The last 10 patches from head in a single patch file:

git format-patch -10 HEAD --stdout > 0001-last-10-commits.patch

git commit --amend [-m …]

dod commit code

git push

Ubuntu

Check version

cat /etc/lsb-releases

* delete a commit

**Careful:** git reset --hard *WILL DELETE YOUR WORKING DIRECTORY CHANGES*. Be sure to**stash any local changes you want to keep** before running this command.

Assuming you are sitting on that commit, then this command will wack it...

git reset --hard HEAD~1

The HEAD~1 means the commit before head.

Or, you could look at the output of git log, find the commit id of the commit you want to back up to, and then do this:

git reset --hard <sha1-commit-id>

If you already pushed it, you will need to do a force push to get rid of it...

git push origin HEAD --force

**However**, if others may have pulled it, then you would be better off starting a new branch. Because when they pull, it will just merge it into their work, and you will get it pushed back up again.

If you already pushed, it may be better to use git revert, to create a "mirror image" commit that will undo the changes. However, both commits will be in the log.

FYI -- git reset --hard HEAD is great if you want to get rid of WORK IN PROGRESS. It will reset you back to the most recent commit, and erase all the changes in your working tree and index.

Lastly, if you need to find a commit that you "deleted", it is typically present in git reflog unless you have garbage collected your repository.

git-apply - Apply a patch to files and/or to the index

## git - 简明指南:

<http://rogerdudler.github.io/git-guide/index.zh.html>

某一个人的提交记录:

git log --author=bob

动提交到远端仓库：  
git push origin master  
可以把 *master* 换成你想要推送的任何分支。

可以使用如下命令添加：  
git remote add origin <server>  
如此你就能够将你的改动推送到所添加的服务器上去了。

除非你将分支推送到远端仓库，不然该分支就是 *不为他人所见的*：  
git push origin <branch>

改完之后，你需要执行如下命令以将它们标记为合并成功：  
git add <filename>

git log --pretty=oneline

通过 ASCII 艺术的树形结构来展示所有的分支, 每个分支都标示了他的名字和标签:

git log --graph --oneline --decorate –all

看看哪些文件改变了:   
git log --name-status

假如你想丢弃你在本地的所有改动与提交，可以到服务器上获取最新的版本历史，并将你本地主分支指向它：  
git fetch origin  
git reset --hard origin/master

### The Merge Option

git checkout feature

## git merge master

Or, you can condense this to a one-liner:

git merge master feature

git merge confict

git mergetool

It opens a GUI that steps you through each conflict, and you get to choose how to merge. Sometimes it requires a bit of hand editing afterwards, but usually it's enough by itself. It is much better than doing the whole thing by hand certainly.

Well, it doesn't necessarily open a GUI unless you install one. Running git mergetool for me resulted in vimdiff being used. You can install one of the following tools to use it instead: meld opendiff kdiff3 tkdiff xxdiff tortoisemerge gvimdiff diffuse ecmerge p4merge araxis vimdiff emerge

## Undo git add

You can use git reset. This will 'unstage' all the files you've added after your last commit.

If you want to unstage only some files, use git reset -- <file 1> <file 2> <file n>.

Also it's possible to unstage some of the changes in files by using git reset -p

## Reversing patches

Last updated 15 March 2016.

You can reverse a patch if you have finished testing it, or if you want to see whether a problem has been introduced by a particular patch. You should also reverse a patch prior to adding a newer, updated version of the same patch. To reverse the patch, use the patch command with the -R option:

patch -p1 -R < path/file.patch

(If your patch was applied with the -p0 option, use that instead.)

Or:

git apply -R path/file.patch

## Git - show history of a file

git log -p -- path/to/file

gitk path/to/file

# 3. vim

## Search and replace syntax

The syntax is as follows:

:s/Search/Replace/CommandFlag  
:s/Search-Word/Replace-Word/g  
:%s/Search-Word/Replace-Word/g  
:%s/Search-Word/Replace-Word/gc

<http://www.cyberciti.biz/faq/vim-text-editor-find-and-replace-all-text/>

below sequence will delete the space between words

w d ? \ s \ + <return>

If your cursor is in the middle of some whitespace, diw will delete whitespace left and right of the cursor. (If it is somewhere in the middle of a word, it will delete the word.)

## cscope

sudo apt-get install vim-scripts

sudo apt-get install exuberant-ctags

copy .vim/plugin

## toggle case

Toggle case "HellO" to "hELLo" with g~ then a movement.

Uppercase "HellO" to "HELLO" with gU then a movement.

Lowercase "HellO" to "hello" with gu then a movement.

select the word with visual mode (viw) and press ~, it switches case for all letters in the word.

## undo

To undo recent changes, from normal mode use the undo command:

u: undo last change (can be repeated to undo preceding commands)

Ctrl-R: Redo changes which were undone (undo the undos). Compare to . to repeat a previous change, at the current cursor position. Ctrl-R (hold down Ctrl and press r) will redo a previously undone change, wherever the change occurred.

# 4. Ethernet

## Namespace

<https://www.toptal.com/linux/separation-anxiety-isolating-your-system-with-linux-namespaces>

<http://www.opencloudblog.com/?p=42>

<https://lwn.net/Articles/580893/>

<https://lwn.net/Articles/531381/>

<http://www.opencloudblog.com/?p=66>

<http://serverfault.com/questions/127636/force-local-ip-traffic-to-an-external-interface>

<http://stackoverflow.com/questions/2734144/linux-disable-using-loopback-and-send-data-via-wire-between-2-eth-cards-of-one>

ifconfig                                             --> ip addr or just ip a

ifconfig <interface> up/down                          --> ip link set dev <interface> up/down

ifconfig <interface> <ip> netmask <netmask>        --> ip addr add <ip>/<masklen> dev <interface>

netstat -rn                                         --> ip route or just ip r

route add -net <net> netmask <netmask> gw <gateway> --> ip r add <net>/<netmasklen> via <gateway>

 ip route show table local

udhcpc -i eth1

ip netns add <new namespace name>

Then, you’d assign the interface to the namespace:

ip link set <device name> netns <namespace name>

# attach it to namespace

ip link set tap1 netns ns1

# set the ports to up

ip netns exec ns1 ip link set dev tap1 up

ip netns exec netns1 ip link list

# ip netns exec netns1 ip link set dev lo u

ip netns exec nstest ip addr add 10.0.0.2/24 dev veth-b

In addition, physical devices (those connected to real hardware) cannot be assigned to namespaces other than the root. Instead, virtual network devices (e.g. virtual ethernet or veth) can be created and assigned to a namespace.

## Ip route iptables:

Have a look in **local** routing table. With iproute2 tools installed do ip route show table local. As you can see, all packets destinated to your local IPs would never go thru NICs since they are marked as **local**.

To force packets go via ethernet card remove the appropriate route (i.e. ip route delete 192.168.122.1 dev eth0 table local). To restore this route just set the interface down and up: the kernel would do the work to insert these routes.

The trick is to use a set of dummy IP addresses to force the kernel into routing it through the wire, and NAT to change it back to the real IP address.

Let eth0 and eth1 be the two ethernet cards; IP0 and IP1 its IP address; MAC0 and MAC1 its MAC address respectively. We will be using two dummy IP addresses: IP00 and IP11.

arp -s IP00 MAC0

arp -s IP11 MAC1

ip route add IP00 dev eth1

ip route add IP11 dev eth0

iptables -t nat -A POSTROUTING -d IP11 -j SNAT --to-source IP00

iptables -t nat -A POSTROUTING -d IP00 -j SNAT --to-source IP11

iptables -t nat -A PREROUTING -d IP00 -j DNAT --to-destination IP0

iptables -t nat -A PREROUTING -d IP11 -j DNAT --to-destination IP1

Use the dummy IP addresses IP00 and IP11 instead of the real one.

## Ifconfig

eth4 Link encap:Ethernet HWaddr 00:60:48:48:8B:E0

UP BROADCAST MULTICAST MTU:1500 Metric:1

RX packets:25908936 errors:0 dropped:0 overruns:679371 frame:14

TX packets:39659170 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1000

RX bytes:38604314640 (35.9 GiB) TX bytes:59092163300 (55.0 GiB)

Interrupt:88 Memory:920024e0000-920024fffff

eth5 Link encap:Ethernet HWaddr 00:60:48:48:8B:E1

UP BROADCAST MULTICAST MTU:1500 Metric:1

RX packets:43018730 errors:0 dropped:0 overruns:1140472 frame:20

TX packets:34436309 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1000

RX bytes:64097907700 (59.6 GiB) TX bytes:51310100410 (47.7 GiB)

Interrupt:153 Memory:920024c0000-920024dffff

frame counts only misaligned frames, it means frames with a length not divisible by 8. Because of that length is not a valid frame and it is simply discarded.

Meanwhile errors counts CRC errors, too-short frames and too-long frames.

overruns counts that times when there is fifo overruns, caused by the rate at which the buffer gets full and the kernel isn't able to empty it.

At last, dropped counts things like unintended VLAN tags or receiving IPv6 frames when the interface is not configured for IPv6.

You can see how many packets have been dropped on your machine with netstat -suna. Mine has dropped 918 packets so far apparently (“918 packet receive errors”)