BTOS : Building a custom ISO

Business Transparent Operation System

# Part one (Installing Centos7)

1. Requirement:

CentOS-7-x86\_64-DVD-1511.iso

CentOS-7-x86\_64-Minimal-1511.iso

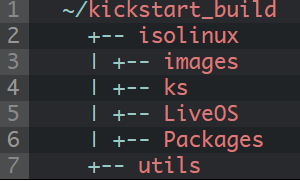
DVD-Version is your host system OS whose anaconda.cfg will be used later, and Minimal-Version is the one that has minimal packages (calculated 300MiB approximately).

1. Prepare your host system (DVD)

Installing DVD-Version and you will build host environment including root-passwd, user-name, user-passwd, patition, network and stuff. Anaconda.cfg reflects those and decides what kind of configuration will be used in BTOS (of course, you can modify it according to certain cirmustance).

1. #mkdir & #cp (to store and build iso)

Mkdir will goes like this:



1. Copy all the files from the isolinux directory on CentOS DVD into your ~/kickstart\_build/isolinux directory.
2. Copy .discinfo from the CentOS DVD into your ~/kickstart\_build/isolinux directory.
3. Recursively copy the contents of the images directory on the CentOS DVD into your ~/kickstart\_build/isolinux/images directory.
4. Copy the contents of the LiveOS directory on the CentOS DVD into your ~/kickstart\_build/isolinux/LiveOS directory.
5. Get the comps.xml file from repodata. This file is named with a unique hex string for each release. In CentOS7 example, it is called 4b9ac2454536a901fecbc1a5ad080b0efd74680c6e1f4b28fb2c7ff419872418-c7-x86\_64-comps.xml.gz.

Copy it to ~/kickstart\_build/comps.xml.gz and gunzip it so that you have ~/kickstart\_build/comps.xml. Your kickstart config file (anaconda.cfg) will go into ~/kickstart\_build/isolinux/ks. I like to have a separate directory for this, because I build a series of different config files for different machines or classes of machines.

1. Copy this anaconda.cfg (/root/anaconda.cfg: Attention! We should su root, or else we cannot find this file. BTW, #sudo will be of no use) into ~/kickstart\_build/isolinux/ks/ks.cfg.

If you want to install BTOS and clear all data stored in the disks, try “clearpart --drives=sda --all –initlabel” instead of “clearpart --none –initlabel ”.

1. Copy the packages (RPMs)

Extracting Minimal-Version, searching a particular file named “Packages”, copying all the RPMs to your host machine ~/kickstart\_build/all\_rpms.

# Part two (RPMs Dependencies)

1. Add your own rpms to ~/kickstart\_build/all\_rpms. Definitely, you are supposed to test RPMs dependencies.
2. Test dependencies by enter command as following:

cd ~/kickstart\_build/isolinux/Packages

mkdir /tmp/testdb

rpm --initdb --dbpath /tmp/testdb

rpm --test --dbpath /tmp/testdb -Uvh \*.rpm

1. Resolving dependencies by yourself if the dependencies break. The only thing to do is copy dependency packages from all\_rpms or from Packages in DVD-Versions (extracted like Minimal-Version).

# Part three (Build your own ISO)

1. Prepare for the repository:

cd ~/kickstart\_build/all\_rpms

sudo rpm -Uvh \

createrepo-\*.el7.noarch.rpm \

deltarpm-\*.el7.x86\_64.rpm \

python-deltarpm-\*.el7.x86\_64.rpm \

libxml2-python-\*.el7.x86\_64.rpm

1. Create the repository:

cd ~/kickstart\_build/isolinux

sudo createrepo -g ~/kickstart\_build/comps.xml .

1. Prepare for the ISO:

cd ~/kickstart\_build/all\_rpms

sudo rpm -Uvh \

genisoimage-\*.el7.x86\_64.rpm \

libusal-\*.el7.x86\_64.rpm

1. Add options to start your iso kickstart

label ks

menu label ^Kickstart

menu default

kernel vmlinuz

append initrd=initrd.img inst.stage2=hd:LABEL=CentOS\x207\x20x86\_64 inst.ks=cdrom:/dev/cdrom:/ks/ks.cfg

1. Build your ISO

cd ~/kickstart\_build

chmod 664 isolinux/isolinux.bin

sudo mkisofs -o custom.iso -b isolinux.bin -c boot.cat -no-emul-boot \

-V 'CentOS 7 x86\_64' \

-boot-load-size 4 -boot-info-table -R -J -v -T isolinux/

Then you can enjoy your own ISO…

# Part four (Custom Settings)

1. Figures:
2. Location: kickstart\_build/isolinux/splash.png.

Method 1: you can change that figure to the certain figure, for the background to convert into yours.

Method 2: you can change kickstart\_build/isolinux/isolinux.cfg by commenting this line “menu background splash.png”, then that fig is of no significant.

1. Location: .buildstamp

You can search through this routine: “squashfs.img” -> “squashfs-root/LiveOS” -> ”[mount point]”. The [mount point] contains that .buildstamp file. Modify it and you will find that the point where you are almost done with installation bring you a big surprise! See the .buildstamp sample.

1. Location: src/libply-splash-core/ply-text-progress-bar.c

That file stored in “plymouth-0.8.9-0.24.20140113.el7.centos.src.rpm” is used to customize the sentence near progress bar when you finish installation and reboot the computer. See the patch.

1. Location: constants.py

You can search through this routine: “squashfs.img” -> “squashfs-root/LiveOS” -> ”[mountpoint]”. Then you can follow this path : usr/lib64/python2.7/site-packages/pyanaconda. You’ll find constants.py. Modify it and you will find that the point where you are almost done with the installation bring you a big surprise! See the constants.py sample.

1. Characters
2. Location: etc/os-release [optional]

That file is not in your ISO, nor in your kickstart\_build, but in “centos-release-7-2.1511.el7.centos.2.10.src.rpm”.

Two options you need to choose when you start the compute and the title “CentOS Linux 7 (core)” when you enter your character interface, both depends your os-release file. Grub-rpm when it’s installing will read the etc/os-release file and write it into grub.cfg in /boot/grub2.

1. Location: kickstart/isolinux/isolinux.cfg.

That file will be modified according to your favourites, deciding what your installation fig looks like.

1. Location: etc/os-release

That file stored in sqaushfs.img&initrd.img is totally different from the one we talked before. When you start your installation, you will notice 2 lines printing separately on the screen (one is in red, the other blue). Modify os-release file and you will override that two lines.

1. Location: util/grub.d/10\_linux.in b/util/grub.d/10\_linux.in

That file stored in “grub2-2.02-0.29.el7.centos.src.rpm” is used to customize the grub options. See the patch.

1. Location: isolinux.cfg

That file is able to change what installation-from-beginning looks like (how many options you could choose, where the options should appear…).

# Appendix

## Squashfs.img (How to modify it)

1. #sudo unsquashfs squashfs.img

It will generate squashfs-root file at the current path because of “extracting” img.

1. #cd squashfs-root/LiveOS/

You’ll find that there’s another img called rootfs.img.

1. #sudo mkdir /mnt/rootfs.mountpoint

Create a file which is used later in mount operation called rootfs.mountpoint in the /mnt path.

1. #sudo mount xxx /mnt/rootfs.mountpoint

Execute this command and you will find what you’re looking for is in the /mnt/rootfs.moutpoint. “xxx” is the location where you put rootfs.img.

1. After finishing modifying some data, you need to convert it into what they were originally.
2. #sudo umount /mnt/rootfs.mountpoint

Unmount rootfs.img.

1. #sudo mksquashfs squashfs-root squashfs.img

Make sure that you are in the same path with squashfs-root, then you can execute that command. Everything turns back to what they were from the beginning.

Comment: It can be a little bit larger or smaller than the squashfs.img in CentOS-ISO. The reason why that a series of operations cause file-size problem is the unsquashfs&mksquashfs version. It all depends the squashfs. The closer to the latest-version, the more efficiently squashfs-compression works.

## initrd.img (How to modify it)

1. #su

#su allows users to run program with the privileges of superuser.

1. #xz -dc initrd.img | cpio –id

We can call it “extract operation”.

1. After finishing modify the file, we should turn it back.
2. #find . | cpio -c -o | xz -9 --format=lzma > initrd.img

We can call it “compress operation”.

Comment: It can be a little bit larger or smaller than the squashfs.img in CentOS-ISO. The reason why that a series of operations cause file-size problem is the unsquashfs&mksquashfs version. It all depends the squashfs. The closer to the latest-version, the more efficiently squashfs-compression works.

## \*.src.rpm (How to build your own rpm)

1. #mkdir ~/rpmbuild
2. #sudo useradd rpmbuild
3. #rpm –i xxx.src.rpm
4. #rpm –bp SPEC/xxx.spec
5. #mkdir ~/patch
6. #sudo cp –R BUILD/xxx ~/patch
7. #cd ~/patch
8. #git init
9. #git add xxx
10. #git commit xxx –m “testonly”
11. Modifying…
12. #git diff > xxx.patch
13. Copy the patch to SOURCES and Modify the spec.
14. #rpmbuild –bb/-ba xxx.spec

## Resolve Packages Dependency

There’re two ways to solve this kind of problem. One is obtain the cache from yum command, the other is using special parameters to obtain.

1. First method:
2. The first and foremost thing is to modify /etc/yum.cfg. Change from “keepcache=0” to “keepcache=1”. 1 means if we use yum to install, it will download the rpms to /var/cache/yum. 0 otherwise. So you can use #sudo yum install xxx. After you’re done with installation with rpms, you can search through /var/cache/yum. Then you can get what you want (search file whose name includes packages).
3. Second method:

Using # yum install xxx --downloadonly --downloaddir=/xxx –y command. Obviously, you know that xxx is what you want to install, that downloadonly means you do not install but download, that downloaddir is the location where you want to download.

1. Warning:

It’s likely to happen –downloadonly parameter goes wrong because you don’t install yum-utils before. Then execute by entering “# yum -y install yum-utils\* ” to install them, then you could successfully use above commands.

## Grub2-RPM (generating a custom grub.cfg)

1. Modify 10\_linux.in file

That file is located in path util/grub.d/10\_linux.in, deciding what your grub options looks like. See the patch!

Q: How does the grub generate configure?

A: grub-mkconfig searches all the file in “/etc/grub.d” path, then writes them into /boot/grub2/grub.cfg. However, not all the file in /etc/grub.d decide what kind of options you could choose when you start your newly-installed operation system. One is an exception that decides the options-----10\_linux. Change 10\_linux and you will modify your options at will. Be careful that what you’re going to change may cause fatal error. The file, when the grub-rpm-installation is running, will read a title(called menuentry) from /etc/os-release(It’s generated by another rpm) to grub.cfg. More details in patch!

## xxx.Service(how to start a service at boot time)

## Samba configurations (how to)

1. Install all the packages if you’re a newbie. See the packages included in this root directory. Or you can execute #sudo yum install samba\* to get/download all the samba-related rpms. Or you can install several rpms like samba/samba-common/samba-client. If you finish your installation, you may check whether they have been installed by entering #rpm –qa | grep samba.
2. Modify the file: /etc/samba/smb.cfg.
3. Test if your smb.cfg is correct. Using #testparm.
4. Add a new account for your smb(If you don’t use the current-account , then you need to add account to your system, then to your smb). Don’t forget set the passphrase by using #sudo smbpasswd –a [username], and make it enable by entering #sudo smbpasswd –e [username].
5. Disable the firewall [optional]

#iptables –F

# setenforce 0

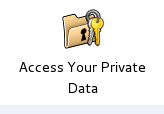
1. Modify the dns-server. Add a line “nameserver 127.0.0.1” to /etc/resolv.cfg. Then restart the smb-server by using #sudo systemctl restart smb/nmb. If you may want to check if they work as expected, you can try #systemctl status smb/nmb or #ps aux | grep smb/nmb.
2. Connect to another computer. Please ensure that two or more computers are in the same local area network.

Using #smbclient -L 127.0.0.1 -U www%[passphrase] to check some basic information.

Using #smbclient 127.0.0.1 -U www%[passphrase] to enter into smb-mode.

1. Remember that #get in smb-mode is downloading, #put uploading. If you want to get lots of files, then you need to compress them into one file and download.

## eCryptfs-utils-[xxx](installation)



1. Download eCryptfs-utils from ecryptfs Launchpad. ~~Choosing utils-100 or even higher is a good way to go because I’ve tried many times to build utils-96/82 on the centos7(1511) platform, which causes lots of errors. I recommend you to use utils-100 or higher.~~
2. Extract utils and enter into the source directory. More details in INSTALL/README.

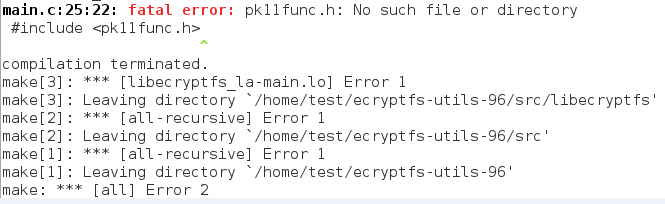
#sudo sh ./configure –prefix=/usr

#sudo make

#sudo make install

Here’re something special I want to declare:

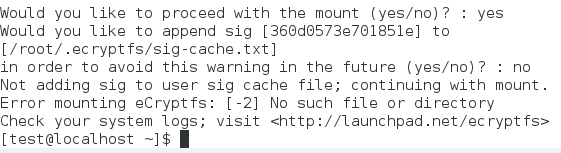
Q: Why isn’t eCryptfs able to find pk11func.h through search /usr/lib path. And the pk11func.h does exist!



A:

1. Installing \*-devel.\*.rpm like nss-devel.\*.rpm.
2. Using #sudo make clean to remove some file generated during compiling time.
3. Entering #sudo sh ./configure –prefix=/usr and #sudo make to rebuild the eCryptfs-utils source.

Q: After the eCryptfs-utils installation, if you use #sudo mount –t ecryptfs raw secure. It will cause some problems. One of them goes like this:



A: It is, indeed, an bug that haven’t been fixed by developers. So deleting those files generated during compile, build, installation time. And do as I said above, choosing utils-100 would be better.

Q: After the eCryptfs-utils installation, if you use #sudo mount –t ecryptfs raw secure. It will cause some problems. One of them goes like this:

“ [**error while loading shared libraries**](http://blog.csdn.net/dumeifang/article/details/2963223)”

“ error while loading shared libraries: xxx.so.0:cannot open shared object file: No such file or directory ”

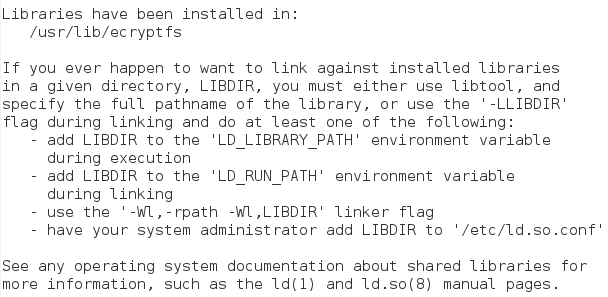
A: Modify the /etc/ld.so.conf file and using #sudo ldconfig –v to restart the so-link operations. What kind of line do you add to that file, you may ask.

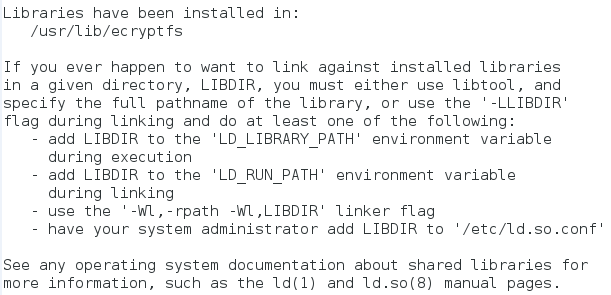
Say, adding line “include /usr/lib/libecryptfs\*” and “include /usr/lib/ecryptfs/\*” would be fine.

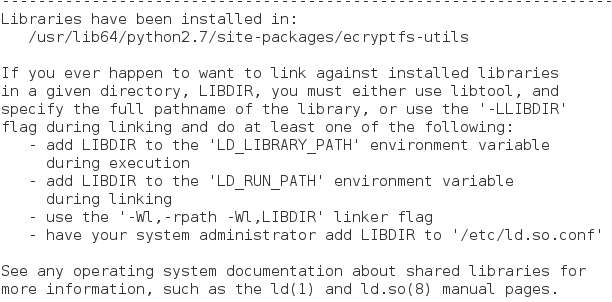
In terms of the ecryptfs-utils installation, why, you may wonder, shouldn’t the .so.0 be recorded by libtool?

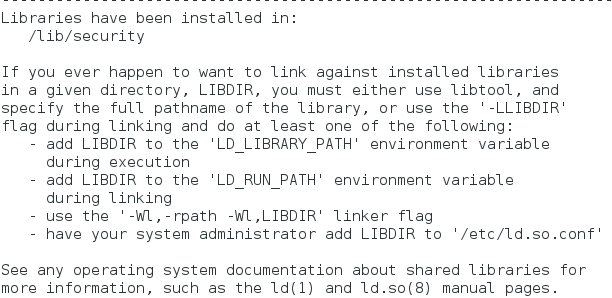
Yes, it should be recorded by libtool which are obliged to write those files(like .so file) to a certain file. Ld.so.conf is just an example. But it need to be recorded automatically! Ld.so.conf is read by ld.so.cache which is used to search those files. After your utils installation, using #sudo ldconfig –v to refresh the cache is the best way to go. Give it a shot!

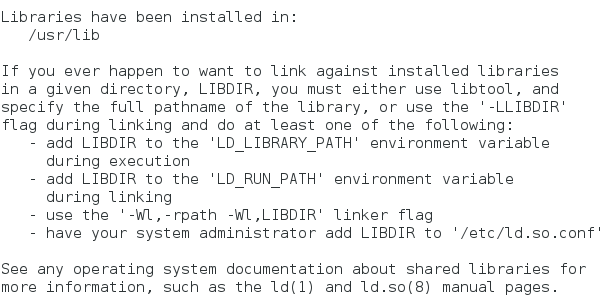
BTW, if you pay more attention to what the “make” command have done, you may be aware of such a paragraph:





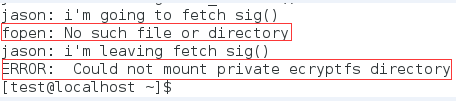






That specifications have already showed you the way to solve .so file problem.

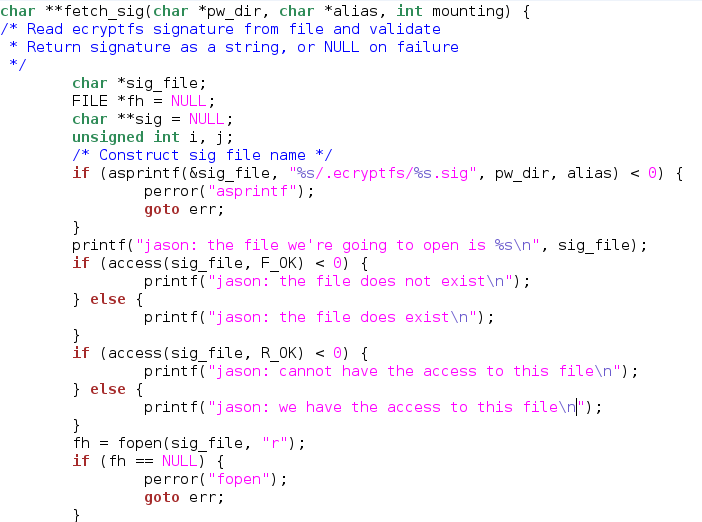
Q: Why fopen incurs error(see below) if I use #ecryptfs-setup-private?



(the sentences with Jason-prefix are debug information, in order to locate where goes wrong)

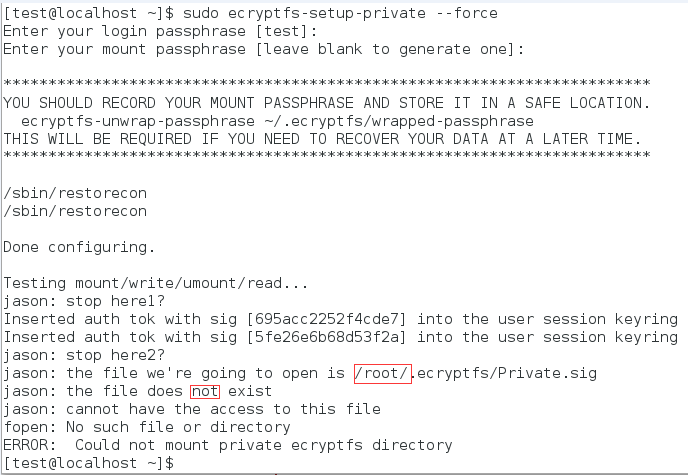
A: Apparently, the fetch\_sig() function goes wrong. The executable wants to open a file located in your $HOME path, but it failed. Be careful that you must use #sudo ecryptfs-setup-private to set up ecryptfs environment, your $HOME is your home directory; If you add sudo, that means your home directory is /root (not /home/[username])! So it fails without any doubt. Do not use sudo to setup and non-interactive-mount.

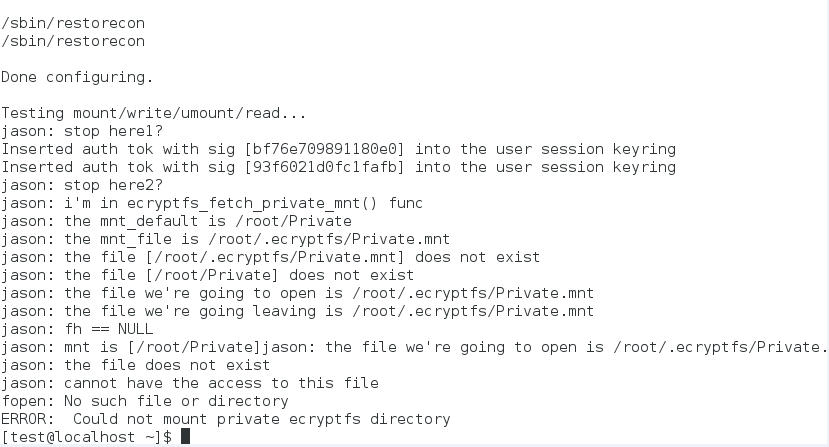
Here’s my debug code(part):



To check that whether the file exists or not, that whether the file holds the RW-permission or not.

And here’s the result for my debug information(part):





Q: If i use #ecryptfs-setup-private, there’s still one problem said “setreuid: operations not permitted”.

A: Because your certain file the system is working on don’t have the right to use setuid/setreuid… Try #ls –il /usr/sbin/mount.ecryptfs\_private

, and you will find there’s no “s” property which can make the file have the right to execute some function like setreuid().

Now, it’s effortless for you to use #sudo chmod u+s /usr/sbin/mount.ecryptfs\_private to add the “s” property! Run #ecryptfs-setup-private command one more time! I’m convinced that you can make it!

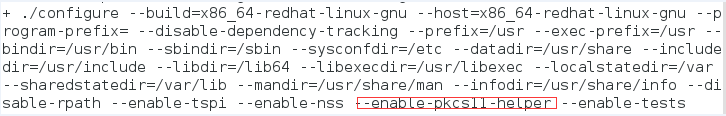
BTW, this is a striking difference between installing from source and installing from rpms for rpms is built according to SPEC which includes that “s” property by adding “$attr{4750, root, ecryptfs}”. 4750 properties include “s” property : )

ATTENTION:

1. SPEC doesn’t read or cannot recogonize well about the comment, say,

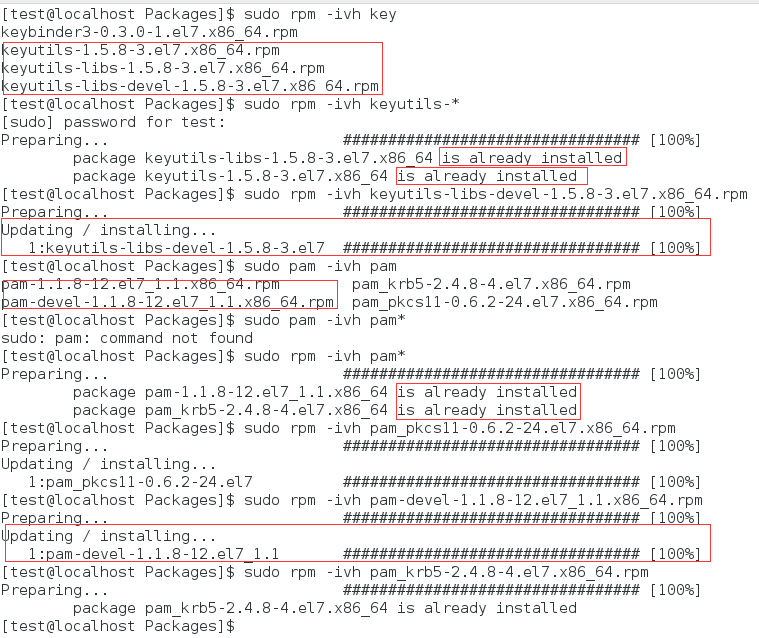


The SPEC will execute %configure twice and enable pkcs11. So the comment here doesn’t work well. Delete the sentence and you will get rid of that problem.



That figure is the information generated during #rpmbuild -bc xxx.spec time.

1. You may be stopped by rpm -ivh command, say,



Check one more time if you use \* to match all or some certain rpms to be installed later!