LIBVIRT with KVM

1. INSTALL Ubuntu on PC with multi OS

First I used Wubi to install Ubuntu, but later I found that it will not work at the virtualization because of its virtual disk of windows. Besides, Ubuntu of Wubi installation has no its own bootloader, instead, it uses windows' bootloader. So when install a virtual machine in it, it will not start up. So I installed Ubuntu on my hard disk.

Please note that you should use the 64-bit Linux version.

First, you should ensure that your PC hardware can provide hardware virtualization. If so, you can open it by setting the BIOS options.

Second, you should allocate a free disk from your hard disk and ensure that this free disk is the first and primary disk of your hard disk. Then delete this free disk to left this disk unused such that it can be partitioned in the process of Ubuntu installation.

Then install Ubuntu by CD-ROM or USB. It should be ensured that it should be installed to your hard disk.

When finished the installation of Ubuntu, you should set the BIOS boot order options to change the USB hard disk's priority prior than the hard disk of desktop, So Ubuntu can directly boots from my hard disk without to use easyBCD software to configure boot options.

You can refer to http://www.ctocio.com.cn/35/12325035.shtml. But here, I did not use easyBCD software to configure the boot options. I change the BIOS boot directly from my hard disk instead of hard disk of PC itself.

2. CONFIGURE Ubuntu (working PC only)

2.1 You should configure the network using proxy

a) From Terminal

```
$export http_proxy='http://www-proxy.ericsson.se:8080'
```

After this, the Ubuntu can connect to the Internet. If you use the command of $sudo apt-get update command or install a new software, it can't connect to the software center of Ubuntu, you can fix this problem by the following command:

```
$sudo visudo
```

Then find a line that states:

```
Defaults env_reset
```

And add after it:

```
Default env_keep="http_proxy"
```

b) From GUI

Use the network setting panel to set the network proxy. You should select manual configuration in the process.

For firefox, you should configure the inner network proxy (Automatic proxy configuration URL):

http://www-proxy.ericsson.se:3132/accelerated_pac_base.pac

2.2 For email configuration of thunderbird, please refer to (internal network)

https://wiki.lmera.ericsson.se/wiki/LinuxMWP#Configuring_Thunderbirdsudo

3. INSTALLATION of KVM

(Refer to https://help.ubuntu.com/community/KVM/Installation)

3.1 Pre-install Checklist

a) Check whether your CPU supports hardware virtualization

```
$egrep -c '(vmx|svm)' /proc/cpuinfo
```

If **0** it means that your CPU doesn't support hardware virtualization.

If **1** or more it support, but you still need to make sure that virtualization is enabled in BIOS.

b) Alternatively, you may execute:

```
$kvm-ok
```

It may provide an output like this:

INFO: /dev/kvm exists

KVM acceleration can be used

This infers that your CPU supports hardware virtualization

3.2 Installation of KVM

a) Install necessary packages

```
$ sudo apt-get install qemu-kvm libvirt-bin ubuntu-vm-builder bridge-utils
```

- *libvirt-bin provides libvirtd which you need to administer qemu and kvm instances using libvirt*
- *qemu-kvm*(*kvm*in Karmic and earlier) is the backend
- *ubuntu-vm-builder* powerful command line tool for building virtual machines
- *bridge-utils* provides a bridge from your network to the virtual machines

Optional: Install virt-manager (graphical user interface)

You should first ensure that you are using the desktop vision or server vision with X.

```
$ sudo apt-get install virt-manager
```

b) Add User to Groups

You need first to add your username to group libvirtd

```
$ sudo adduser `id -un` libvirtd
```

```
$sudo adduser `id -un` kvm
```

After this, you need to logout and relogin. Then you can use the groups command to c heck whether your user has been added into group libvirtd.

c) Verify installation

You can test if your install has been successful with the following command:

```
$ virsh -c qemu:///system list
```

If you get like the following, it's OK.

```
 Id     Name                           State

 ----------------------------------

$
```

3.3 Use virt-install to create guest

a) Install virt-viewer (Desktop virsion)

We can use virt-viewer to connect to the graphical console.

```
$sudo apt-get install virt-viewer
```

b) Create an .img file for installation. This file acts as a virtual disk for your virtual machine

You have two possibilities for creating such a file.

(1) use dd command

```
$dd if=/dev/zero of=ubuntu.img bs=1G count=10
```

(2) Use qemu-img command (Recommended)

```
$qemu-img create -f raw ubuntuServer.img 12G
```

If you want to use the qcow2 format instead of raw, it left you to try. But I failed to create virtual machine using qcow2 format.

The following command can convert qcow2 format to raw format to /dev/sdb, such that it can succeed in Wubi installation of ubuntu.

```
$sudo qemu-img convert ubuntuServer.qcow2 -O raw /dev/sdb
```

c) Install VM using virt-install command

```
$sudo virt-install --connect qemu:///system -n ubuntuServer -f ubuntuServer.img -r 512 -s 12 -c ubuntu-12.04-server-amd64.iso --vnc --noautoconsole --os-type linux --accelerate --network=network:default
```

Then one by one, you will success to install the Ubuntu server.

If you want to use bridge network, you can use the following command after you have configure the bridge network first. But you will find that the installation process will be stop at the step of setting the system clock because the system uses NTP protocol to access the time. But this time your bridge network will not work, so it paused at this step.

```
$sudo virt-install --connect qemu:///system -n ubuntuServer -f ubuntuServer.img -r 512 -s 12 -c ubuntu-12.04-server-amd64.iso --vnc --noautoconsole --os-type linux --accelerate --network bridge=br0
```

Alternative, you can use following command to create a VM after you have prepared for the XML description of VM.

```
$virsh create ubuntuServerxml
```

Of course, there are many other ways to install a guest, such as the following way. If you have interesting, you can test it. (I have not tested it)

You can use kvm command to create VM

```
$sudo kvm -hda ubuntu.img -cdrom ubuntu-12.04-desktop-i386.iso -boot d
```

You can use many other parameters to configure your VM.

You also have the possibility to add more virtual disks to one VM. You can add the following parameter to the start command :

```
--hdb disk02.img
```

d) Connect to the guest

Using the following command, we can connect to a guest and can see the installing graphic and finish the installation:

```
$sudo virt-viewer   ubuntuServer
```

Alternative, you can use $sudo virt-manager, you will see ubuntuServer guest.

Step by step, you can finish the installation.

After this, you have finished the virtual machine installation. By default, the virtual machine uses NET model for internet. If you want other host to connect this machine, you should use bridge model for internet. You can refer to the following to set.

e) Configure the bridge mode

(1) To setup a bridge interface, edit /etc/network/interfaces, by this can create a virtual interface br0:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet manual

auto br0
iface br0 inet dhcp
        bridge_ports eth0
        bridge_stp off
        bridge_fd 0
        bridge_maxwait 0
```

(2) Restart the network and will create a virtual interface br0

```
$sudo /etc/init.d/networking restart
```

Then you can use command ifconfig to see the interface br0.

You also can use the following command to see the bridge interface:

```
$brctl show
```

By doing this, if your host can not connect to internet, you can change to like the following and restart the network:

auto lo

iface lo inet loopback

auto eth0

iface eth0 inet dhcp

auto br0

*iface br0 inet **dhcp***

bridge_ports eth0

*bridge_stp **on***

bridge_fd 0

bridge_maxwait 0

After your host network work, you should change to static mode.

4. CONTROL the ubuntuServer by virsh command

Virsh provides two mode, command mode and interactive mode. You can use the following command to enter interactive mode.

```
$virsh –c qemu:///system
```

Following, I just list some frequently-used commands by command mode. Some other commands, you can refer to http://www.libvirt.org/virshcmdref.html.

You can use the following command to see the ubuntuServer guest in this host:

```
$virsh -c qemu:///system list
```

You can start a virtual machine in this host created before by the following command:

```
$virsh start ubuntuServer
```

You can reboot a virtual machine in this host created before by the following command:

```
$virsh reboot ubuntuServer
```

You can shut down a virtual machine in this host created before by the following command:

```
$virsh shutdown ubuntuServer
```

You can suspend a virtual machine in this host created before by the following command:

```
$virsh suspend ubuntuServer
```

You can resume a virtual machine in this host suspended before by the following command:

```
$virsh resume ubuntuServer
```

To view the details of configurations about a particular virtual machine:

```
$virsh dumpxml ubuntuServer
```

Also, you can save the configurations to a file:

```
$virsh dumpxml ubuntuServer > ubuntuServer.xml
```

You can delete a virtual machine, first terminate it(if running), and then undefine it:

```
$virsh destroy ubuntuServer ( as plug up the power cable)
```

$virsh undefine ubuntuServer(**Be careful,** this will delete the /etc/libvirt/qemu/ubuntuServer.xml)

If you want to change some parameters this image, you can edit the /etc/libvirt/qemu/ubuntuServer.xml. After you having edited it, you can you the following to update it (sometimes, you need restart it):

```
$virsh define /etc/libvirt/qemu/ubuntuServer.xml
```

Of course, you can edit a copy of this guest like following (recommended):

```
$virsh dumpxml ubuntuServer > cpubuntu.xml
```

Then you edit this copy, stop this guest and do the following command to update it:

```
$virsh define cpubuntu.xml
```

Of course, there are many other commands. If you need, you can read the manual of virsh by the command 'man virsh'

References:

http://www.linux-kvm.org/page/HOWTO

http://www.howtoforge.com/using-kvm-on-ubuntu-gutsy-gibbon

https://help.ubuntu.com/community/KVM/Installation

http://www.howtoforge.com/installing-kvm-guests-with-virt-install-on-ubuntu-8.10-server

https://help.ubuntu.com/community/KVM/CreateGuests

5. INSTALLATION of libvirt source

5.1 Download the libvirt source and compile it

```
$sudo apt-get source libvirt
```

You can use the following command to install all the necessaries:

```
$sudo apt-get install libxml2 libxml2-dev gnutls-doc gnutls-bin libneon27-gnutls libcurl4-gnutls-dev libdevmapper-dev python-dev libnl-3-dev
```

Then get into the ./libvirt-0.9.8 directory:

```
$sudo ./configure --with-qemu --with-yajl --with-libxml=/usr --with-python --with-udev --with-hal --without-openvz --without-vmware --without-xen --without-xen-inotify --without-xenapi --without-vbox --without-lxc --without-lxc --without-esx --without-selinux
```

If the process of your configuration has the following errors, you can do the following commands of each other.

a) If it gives an error like: "configure: error: Could not find libxml2 anywhere(see config.log for details) "

Then use the following command to download and install libxml2:

```
$sudo apt-get install libxml2 libxml2-dev
```

b) If it gives an error like : "configure: error: You must install the GnuTLS library in order to compile and run libvirt"

Then use the following command to install the necessaries:

```
$sudo apt-get install gnutls-doc gnutls-bin libneon27-gnutls libcurl4-gnutls-dev
```

c) If it gives an error like :"configure:error: You must install device-mapper-devel/libdevmapper >= 1.0.0 to compile libvirt"

Then use the following command:

```
$sudo apt-get install libdevmapper-dev
```

d) If it gives an error like :"configure: error: You must install python-devel to build python bindings"

Then use the following command:

```
$sudo apt-get install python-dev
```

e) If it gives an error like :"configure: error: libnl-devel >= 1.1 is required for macvtap support"

Then use the following command:

```
$sudo apt-get install libnl-3-dev
```

f) If it gives an error like :" configure: error: You must install the YAJL development package in order to compile libvirt"

Then use the following command:

```
$sudo apt-get install libyajl-dev
```

g) If it gives an error like :"configure: error: You must install hal-devel >= 0.5.0 to compile libvirt"

Then use the following command:

```
$sudo apt-get install libhal-dev
```

h) If it gives an error like :"configure: error: You must install libpciaccess-devel >= 0.10.0 to compile libvirt"

Then use the following command:

```
$sudo apt-get install libpciaccess-dev
```

Then do the following commands to install:

```
$sudo make
```

```
$sudo make install
```

After you have make install the libvirt, when you do the following command, it may be an error :"error: Failed to connect socket to '/usr/local/var/run/libvirt/libvirt-sock': No such file or directory"

```
$sudo virsh -c qemu:///system
```

You should do the following command:

```
$sudo libvirtd -d
```

5.2 Configure a guest with bridge network

a) To setup a bridge interface, edit /etc/network/interfaces, by this can create a virtual interface br0:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet manual

auto br0
iface br0 inet dhcp
        bridge_ports eth0
        bridge_stp off
        bridge_fd 0
        bridge_maxwait 0
```

b) First dump a configuration of this guest

```
$virsh dumpxml ubuntuServer > ubuntuServerBk.xml
```

c) Modify the network interface configuration

```
<interface type='network'>
  <mac address='52:54:00:ae:fa:e3'/>
  <source network='default'/>
  <target dev='vnet0'/>
  <alias name='net0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03'
function='0x0'/>
</interface>
```

To :

```
<interface type='bridge'>
  <mac address='52:54:00:ae:fa:e3'/>
  <source bridge='br0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03'
function='0x0'/>
</interface>
```

d) Redefine the guest (shutdown first)

```
$virsh define ubuntuServerBk.xml
```

e) Start ubuntuServer

```
$virsh start ubuntuServer
```

5.3 Connect to virtual machine

There are many methods, you can choose which you like as followings.

a) Use virt-viewer or virt-manager to connect virtual machine

```
$sudo virt-viewer ubuntuServer
```

b) Use vinagre tool

$sudo vinagre localhost:0

c) Use xvnc4viewer tool

$sudo xvnc4viewer localhost 0

d) Use xvncviewer tool

$sudo xvncviewer localhost:0

$sudo xvncviewer localhost:1

Use this tool, can open multiple virtual machine.

6. API of libvirt (C++)

All API of libvirt can be viewed in :/usr/local/include/libvirt/libvirt.h, /usr/local/include/libvirt/llibvirt-qemu.h and /usr/local/include/libvirt/lvirterror.h.

6.1 Connections

a) The first thing a libvirt agent must do is to call one of the libvirt connection to obtain a virConnectPtr handle. Libvirt library provides three different functions for connecting a resource:

(a) virConnectPtr virConnectOpen(const char* name)

(b) virConnectPtr virConnectOpenReadOnly(const char *name)

(c) virConnectPtr virConnectOpenAuth(const char *name, virConnectAuthPtr auth, int flags)

In all three cases there is a name parameter which in fact refers to the URI of the hypervisor to connect to.

b) Examples of "open.c"

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <libvirt/libvirt.h>
 4
 5 int main(int argc,char *argv[])
 6 {
 7     virConnectPtr conn;
 8     conn = virConnectOpen("qemu:///system");
 9     if(conn == NULL)
10     {
11         fprintf(stderr,"Failed to open connect to qemu:///system\n");
12         return 1;
13     }
14     fprintf(stdout,"open ok\n");
15     virConnectClose(conn);
16     return 0;
17 }
```

Then make it:

$gcc -Wall -o open open.c *-lvirt*

Run it:

./open // It should print "open ok"

If have the an error like:"libvir: RPC error : Failed to connect socket to '/usr/local/var/run/libvirt/libvirt-sock-ro': No such file or directory
Failed to open connect to qemu:///system"

You should first open the libvirtd demaon by run

$sudo libvirtd -d

The usage of other two functions, you can refer to the Application development guider of the libvirt website.

c) capabilities information

The **virConnectGetCapabilities** API can be used to obtain capabilities information about the virtual host.

The following code demonstrates the use of virConnectGetCapabilities (capa.c):

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <libvirt/libvirt.h>
 4
 5 int main(int argc,char *argv[])
 6 {
 7     virConnectPtr conn;
 8     char *caps;
 9
10     conn = virConnectOpen("qemu:///system");
11     if(conn == NULL)
12     {
13         fprintf(stderr,"Failed to open connect to qemu:///system\n");
14         return 1;
15     }
16     caps = virConnectGetCapabilities(conn);
17     fprintf(stdout,"capabilities:\n%s\n",caps);
18     free(caps);
19
20     virConnectClose(conn);
21     return 0;
```

22 }

Make it by $gcc -o capa -Wall capa.c -lvirt.

The meaning of the output of $./capa can refer to the application development guider.

d) Host information

The following APIs can be used to get information about the virtualization host including the hostname, maximum support guest CPUs, etc.

int virConnectGetVersion (virConnectPtr conn, unsigned long *hvVer);

int virConnectGetLibVersion (virConnectPtr conn, unsigned long *libVer);

char * virConnectGetHostname (virConnectPtr conn);

char * virConnectGetURI (virConnectPtr conn);

int virConnectGetMaxVcpus (virConnectPtr conn, const char *type);

int virNodeGetInfo (virConnectPtr conn, virNodeInfoPtr info);

const char * virConnectGetType (virConnectPtr conn);

unsigned long long virNodeGetFreeMemory (virConnectPtr conn);

int virNodeGetCellsFreeMemory(virConnectPtr conn, unsigned long long *freeMems,

int startCell, int maxCells);
int virConnectIsEncrypted(virConnectPtr conn);

int virConnectIsSecure(virConnectPtr conn);

The usage of these APIs can refer the following example (hostInfos.c)

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <libvirt/libvirt.h>
 4
 5 int main(int argc,char *argv[])
 6 {
 7     virConnectPtr conn;
 8     unsigned long ver;
 9     virNodeInfo nodeinfo;
10     unsigned long long *freemem;
11     int numnodes;
12     int i;
13
14     conn = virConnectOpen("qemu:///system");
15     if(conn == NULL)
16     {
17         fprintf(stderr,"Failed to open connect to qemu:///system\n");
18         return 1;
19     }
20
21     fprintf(stdout,"Hostname is :%s\n",virConnectGetHostname(conn));
22
23     fprintf(stdout,"Maximun support virtual CPUs
is :%d\n",virConnectGetMaxVcpus(conn,NULL));
24
25     fprintf(stdout,"Node free memory:%llu\n",virNodeGetFreeMemory(conn));
26
27     virNodeGetInfo(conn,&nodeinfo);
28     fprintf(stdout,"\nModel :%s\n",nodeinfo.model);
29     fprintf(stdout,"Memory Size :%lukb\n",nodeinfo.memory);
30     fprintf(stdout,"Number of CPUs:%u\n",nodeinfo.cpus);
```

```
31      fprintf(stdout,"MHz of CPUs:%u\n",nodeinfo.mhz);
32      fprintf(stdout,"Number of NUMA NODES: %u\n",nodeinfo.nodes);
33      fprintf(stdout,"Number of CPU sockets :%u\n",nodeinfo.sockets);
34      fprintf(stdout,"Number of CPU cores per socket: %u\n",nodeinfo.cores);
35      fprintf(stdout,"Number of CPU threads per core: %u\n",nodeinfo.threads);
36
37      fprintf(stdout,"\n");
38      freemem = (unsigned long long*)malloc(nodeinfo.nodes * sizeof(unsigned long
long));
39      numnodes = virNodeGetCellsFreeMemory(conn,freemem,0,nodeinfo.nodes);
40      for(i=0; i < numnodes; i++)
41          fprintf(stdout,"Node %d:%llukb free memory\n",i,freemem[i]);
42      free(freemem);
43      fprintf(stdout,"\n");
44
45      fprintf(stdout,"Virtualization type:%s\n",virConnectGetType(conn));
46
47      virConnectGetVersion(conn,&ver);
48      fprintf(stdout,"Version:%lu\n",ver);
49
50      virConnectGetLibVersion(conn,&ver);
51      fprintf(stdout,"Libvirt Version:%lu\n",ver);
52
53      fprintf(stdout,"Canonical URI :%s\n",virConnectGetURI(conn));
54
55      fprintf(stdout,"Connection is encrypted:%d\n",virConnectIsEncrypted(conn));
56      fprintf(stdout,"Connection is secure:%d\n",virConnectIsSecure(conn));
57
58      virConnectClose(conn);
59      return 0;
60  }
```

Make it by $gcc -o hostInfos -Wall hostInfos.c -lvirt.

The result of this example is as following:

Hostname is :ericsson
Maximun support virtual CPUs is :16
libvir: error : this function is not supported by the connection driver: NUMA memory information not available on this platform
Node free memory:0

Model :x86_64
Memory Size :3957868kb
Number of CPUs:4
MHz of CPUs:800
Number of NUMA NODES: 1
Number of CPU sockets :1
Number of CPU cores per socket: 2
Number of CPU threads per core: 2

libvir: error : this function is not supported by the connection driver: NUMA memory information not available on this platform

Virtualization type:QEMU
Version:1000000
Libvirt Version:9008
Canonical URI :qemu:///system
Connection is encrypted:0
Connection is secure:1

6.2 Guest Domains

6.2.1 Find a domain

There are four methods at least to do lookup existing domains as followings:

virDomainPtr virDomainLookupByName (virConnectPtr conn, const char *name);

virDomainPtr virDomainLookupByID (virConnectPtr conn, int id);

virDomainPtr virDomainLookupByUUID (virConnectPtr conn, const unsigned char *uuid);

virDomainPtr virDomainLookupByUUIDString (virConnectPtr conn, const char *uuid);

The usages of them can refer to the 4.1 of Application Development Guider.

6.2.2 Listing domains

You can use the following functions to list active domains:

```
int         virConnectListDomains     (virConnectPtr conn, int *ids, int maxids);

int       virConnectNumOfDomains    (virConnectPtr conn);
```

You can use the following functions to list inactive domains:

```
int       virConnectNumOfDefinedDomains    (virConnectPtr conn);

int       virConnectListDefinedDomains (virConnectPtr conn, char **const names,
          int maxnames);
```

The Application Development Guider is wrong about this part.

Examples:

```
//domainList.c
//gcc -o domainList domainList.c -Wall -lvirt
```

```
 1 #include <stdio.h>
 2 #include <libvirt/libvirt.h>
 3 #include <stdlib.h>
 4
 5 int main(int argc,char *argv[])
 6 {
 7     int i,numDomains;
 8     int *activeDomains;
 9     virConnectPtr conn;
10     char **inactiveDomains;
11
12     conn = virConnectOpen("qemu:///system");
13     if(conn == NULL)
14     {
15         fprintf(stderr,"open failed \n");
16         return -1;
17     }
18     fprintf(stdout,"active domain\n");
19     numDomains = virConnectNumOfDomains(conn);
20     fprintf(stdout,"numDomains1 = %d\n",numDomains);
21
22     activeDomains = malloc(sizeof(int) * numDomains);
23     numDomains = virConnectListDomains(conn,activeDomains,numDomains);
24
25     for(i = 0; i < numDomains; i++)
26       printf("ID = %d,Name
= %s\n",activeDomains[i],virDomainGetName(virDomainLookupByID(conn,activeDom
ains[i])));
27     free(activeDomains);
28
29     fprintf(stdout,"\ninactive domain\n");
```

```
 30      numDomains = virConnectNumOfDefinedDomains(conn);
 31      inactiveDomains = malloc(sizeof(char*) * numDomains);
 32      printf("numDomains = %d\n",numDomains);
 33      numDomains =
virConnectListDefinedDomains(conn,inactiveDomains,numDomains);
 34      for(i = 0; i < numDomains; i++)
 35      {
 36              printf("ID = %d,Name
= %s\n",virDomainGetID(virDomainLookupByName(conn,inactiveDomains[i])),inactive
Domains[i]);
 37          free(inactiveDomains[i]);
 38      }
 39      free(inactiveDomains);
 40      return 0;
 41 }
```

This Example will list the active guest and inactive guest on you host OS.

6.2.3 Lifecycle control of guest domain

6.2.3.1 Start and create guest domain

a) Booting a transient guest domain

You can use the following functions to create a transient guest domain:

virDomainPtr virDomainCreateXML (virConnectPtr conn, const char *xmlDesc, unsigned int flags);

The Application Development Guider is wrong about this part.

Example:

// create.c ; gcc –o create –Wall create.c -lvirt

```
  1 #include <stdio.h>
  2 #include <stdlib.h>
  3 #include <sys/types.h>
```

```
 4 #include <sys/stat.h>
 5 #include <fcntl.h>
 6 #include <unistd.h>
 7 #include <libvirt/libvirt.h>
 8
 9 int main(int agrc, char *argv[])
10 {
11     char *filename = "ubuntuServerBk.xml";//dumpxml of ubuntuServer
12     struct stat filestate;
13     char *xmlconfig = NULL;
14     int fd;
15     virDomainPtr domain;
16     virConnectPtr conn;
17
18     fd = open(filename,O_RDONLY);
19     if(fd < 0)
20     {
21         printf("open file failed \n");
22         return -1;
23     }
24     if(stat(filename,&filestate) < 0)
25     {
26         perror("get the state of file failed\n");
27         return -1;
28     }
29     if(filestate.st_size > 0)
30     {
31         xmlconfig = (char *) malloc(sizeof(char) * filestate.st_size);
32         if(xmlconfig == NULL)
33         {
34             perror("failed to get memory \n");
```

```
35              return -1;
36          }
37      }
38      if(read(fd,xmlconfig,filestate.st_size) < 0)
39      {
40          perror("failed to read\n");
41          return -1;
42      }
43      conn = virConnectOpen("qemu:///system");
44      if(!conn)
45      {
46          perror("failed to open connect\n");
47          return -1;
48      }
49      domain = virDomainCreateXML(conn,xmlconfig,0);
50      if(!domain)
51      {
52          perror("Domain creation failed\n");
53          return -1;
54      }
55      printf("Guest %s has booted\n",virDomainGetName(domain));
56      virDomainFree(domain);
57      free(xmlconfig);
58      return 0;
59
60 }
```

b) Defining and booting a persistent guest domain

First, you should use the following function to define a guest:

virDomainPtr virDomainDefineXML (virConnectPtr conn, const char *xml);

Second, use the following function to create and start this guest:

int virDomainCreate (virDomainPtr domain);

The Application Development Guider is wrong about this part.

When create or define a guest domain, there are three provisioning ways, CDROM/ISO image provisioning, PXE boot provisioning and direct kernel boot provisioning. You can configure the XML file to control them. You can refer to Application Development Guider of this part.

6.2.3.2 Shutdown guest domain

You can use the following function to shut down a guest domain.

int virDomainShutdown (virDomainPtr domain);

6.2.3.3 Reboot guest domain

You can use the following function to reboot a guest domain.

int virDomainReboot (virDomainPtr domain, unsigned int flags);

6.2.3.4 Suspend guest domain

You can use the following function to suspend a guest domain.

int virDomainSuspend (virDomainPtr domain);

6.2.3.5 Resume guest domain

You can use the following function to resume a guest domain.

int virDomainResume (virDomainPtr domain);

6.2.3.6 Reset guest domain

You can use the following function to reset a guest domain.

int virDomainReset (virDomainPtr domain, unsigned int flags);

6.2.3.7 Destroy guest domain

You can use the following function to destroy a guest domain.

int virDomainDestroy (virDomainPtr domain);

int virDomainDestroyFlags (virDomainPtr domain, unsigned int flags);

6.2.3.8 Other operations of guest domain

a) You can use the following functions to set a guest domain to autostart on a particular hypervisor, either by the hypervisor itself or libvirt.

int virDomainGetAutostart (virDomainPtr domain, int *autostart)

int virDomainSetAutostart (virDomainPtr domain, int autostart)

b) You can remove a definition of a guest domain by the following functions:

int virDomainUndefine (virDomainPtr domain)

int virDomainUndefineFlags (virDomainPtr domain, unsigned int flags)

c) You save and restore guest domain's configurations

int virDomainSave (virDomainPtr domain, const char *to)

int virDomainSaveFlags (virDomainPtr domain, const char* to, const char *dxml, unsigned int flags)

int virDomainRestore (virConnectPtr conn, const char *from)

int virDomainRestorFlags (virConnectPtr conn, const char* from, const char* dxml, unsigned int flags)

6.3 Network interface

virInterface API: network interfaces on physical hosts

virNetwork API: virtual network interfaces

virInterfaceDefineXML API: configuration for an existing interface

6.3.1 Enumerating Interfaces

a)list active interfaces

```
        int   virConnectNumOfInterfaces (virConnectPtr conn);
        int   virConnectListInterfaces   (virConnectPtr conn, char
**const names, int maxnames);
```

b)list inactive interfaces

```
        int virConnectNumOfDefinedInterfaces (virConnectPtr conn);
        int virConnectListDefinedInterfaces   (virConnectPtr conn, char
**const names, int maxnames);
```

Alternatively, you can use virNodeListDevices function to get the interfaces' informations.

Examples:(listInterfaces.c)

```
 1 #include <stdlib.h>
 2 #include <stdio.h>
 3 #include <libvirt/libvirt.h>
 4 #include <libvirt/virterror.h>
 5
 6 int main(int argc,char*argv[])
 7 {
 8     virConnectPtr conn;
 9     int numInterfaces, i;
10     char **nameInterfaces;
11     virError err;
12
13     conn = virConnectOpen("qemu+unix:///system?/usr/local/var/run/libvirt/libvirt-
sock");
//or qemu:///system
14     if(conn == NULL)
15     {
16         fprintf(stderr,"Failed to open connect to qemu:///system\n");
17         return -1;
18     }
19 //   printf("magic = %d\n",conn->magic);
20     printf("active interfaces \n");
21     numInterfaces = virConnectNumOfInterfaces(conn);
22     printf("active interfaces number = %d\n",numInterfaces);
23     if(numInterfaces < 0)
24     {
25         virCopyLastError(&err);
```

```
26        fprintf(stderr,"virConnectNumOfInterfaces error: %s\n",err.message);
27        virResetError(&err);
28    }
29    else
30    {
31        nameInterfaces =(char**) malloc(sizeof(char*) * numInterfaces);
32        numInterfaces = virConnectListInterfaces(conn,nameInterfaces,numInterfaces);
33        if(numInterfaces < 0)
34        {
35            virCopyLastError(&err);
36            fprintf(stderr,"virConnectListInterfaces error: %s\n",err.message);
37            virResetError(&err);
38        }
39        else
40        {
41            for(i = 0; i < numInterfaces; i++)
42            {
43                printf("Interface name %s \n",nameInterfaces[i]);
44                free(nameInterfaces[i]);
45            }
46        }
47    }
48
49    printf("\ninactive interfaces \n");
50    numInterfaces = virConnectNumOfDefinedInterfaces(conn);
51    if(numInterfaces < 0)
52    {
53        virCopyLastError(&err);
54        fprintf(stderr,"virConnectNumOfDefinedInterfaces error: %s\n",err.message);
55        virResetError(&err);
56        printf("inactive interfaces number = %d\n",numInterfaces);
57    }
58    else
59    {
60        nameInterfaces =(char**) malloc(sizeof(char*) * numInterfaces);
61        numInterfaces =
virConnectListDefinedInterfaces(conn,nameInterfaces,numInterfaces);
62        if(numInterfaces < 0)
63        {
64            virCopyLastError(&err);
65            fprintf(stderr,"virConnectListDefinedInterfaces error: %s\n",err.message);
66            virResetError(&err);
67        }
68        else
69        {
70            for(i = 0; i < numInterfaces; i++)
```

```
71            {
72                printf("Interface name %s \n",nameInterfaces[i]);
73                 free(nameInterfaces[i]);
74            }
75         }
76     }
77
78 //  free(nameInterfaces);
79     virConnectClose(conn);
80     return 0;
81 }
```

./listInterfaces

Result:

active interfaces
libvir: error : this function is not supported by the connection driver: virConnectNumOfInterfaces
active interfaces number = -1
virConnectNumOfInterfaces error: this function is not supported by the connection driver: virConnectNumOfInterfaces

inactive interfaces
libvir: error : this function is not supported by the connection driver: virConnectNumOfDefinedInterfaces
virConnectNumOfDefinedInterfaces error: this function is not supported by the connection driver: virConnectNumOfDefinedInterfaces
inactive interfaces number = -1

But there is an error in this program. I find that by searching from the Internet that there is a bug in the function virConnectNumOfInterfaces.

You also can test by this command : **$virsh iface-list** and you will find the same error.

6.3.2 Find a virtual interface

virInterfacePtr virInterfaceLookupByName (virConnectPtr conn, const char *name);

virInterfacePtr virInterfaceLookupByMACString (virConnectPtr conn, const char *mac);

Examples:

(I eliminate something that is very common.)

```
char *ethx = "eth0";
 virInterfacePtr iface;
 conn = virConnectOpen("qemu+unix:///system?/usr/local/var/run/libvirt/libvirt-sock");
 if(conn == NULL)
 {
    fprintf(stderr,"Failed to open connect to qemu:///system\n");
    return -1;
 }
 iface = virInterfaceLookupByName(conn,ethx);
 if(iface)
 {
    printf("find eth0\n");
    virInterfaceFree(iface);
 }
 else
 {
    printf("failed to find eth0\n");
    return -1;
 }
```

Result:

libvir: error : this function is not supported by the connection driver: virInterfaceLookupByName
failed to find eth0
The same error as above example.

6.3.2 Define and undefine an interface

virInterfacePtr virInterfaceDefineXML (virConnectPtr conn, const char *xmlDesc, unsigned int flags);

int virInterfaceUndefine (virInterfacePtr iface);

int virInterfaceFree (virInterfacePtr iface);

6.3.3 Activating and deactivating an interface

int virInterfaceCreate (virInterfacePtr iface, unsigned int flags);

int virInterfaceDestroy (virInterfacePtr iface, unsigned int flags);

References:

Application Development Guider(http://libvirt.org/devguide.html)