Advanced Module Develop Toolkit (V 0.6)

For Firmware version 5.00.0x and above

For N16000/N12000, FW 1.00.05 and above will be OK

Table of Content

Table of Modification:

Date	Version	Dept.	Author	Note
2011/01/28	0.3	SW1	Enian	1. Add FW V5 Environment
2011/03/02	0.4	SW1&SW2	Enian & Ryan	1. Add 64bit model
2011/03/21	0.5	SW1	Enian	1. Change the name of toolchain 2. Change root password 3. Includes toolchain in VMDK by default
2011/05/13	0.6	SW1	Oscar, Enian, Ryan	1. Set the environmental parameters while compiling applications for 32 bit Thecus NAS 2. Tool Chain modified 3. Changed the info of VMDK environment

A、 VMDK

The VMDK file is an image of VMware guest OS thus 3rd party developers can build up an environment to develop a user module for Thecus NAS. It will need VMware player to mount the image file.

1. System Environment

FW	Thecus NAS Model	Architecture	Development Environment
5.00.0x and above	N7700PRO/N7700PLUS/N8800PRO/N8800PLUS/N5500/1U4600/N4200/N4200PLUS/N4200PRO/N2200XXX/N5200XXX/N8200XXX/1U4200XXXR/1U4200XXXS/N3200XXX	x86-32	Ubuntu-10.4 (64 bit)
1.00.0x and above	N12000/N16000	x86-64	Fedora-12 (64 bit)

2. Operation Flow

2.1. 32bit Models

2.1.1 Environment of VMDK:

Model	Environment
N7700PRO/N7700PLUS/N8800PRO/N8800PLUS/N5500/1U4600/N4200/N4200PLUS/N4200PRO/N2200XXX/N5200XXX/N8200XXX/1U4200XXXR/1U4200XXXS/N3200XXX	1.For 64bit VMDK - VMware Player running in 64 bit OS (Version 2.5.2 build-156735 and above). Please note the 64 bit VMDK just works when the VM host is 64 bit and also the CPU supports VT. 2. For 32bit VMDK - VMware Player version 3.1.4 running in 32 or 64 bit OS

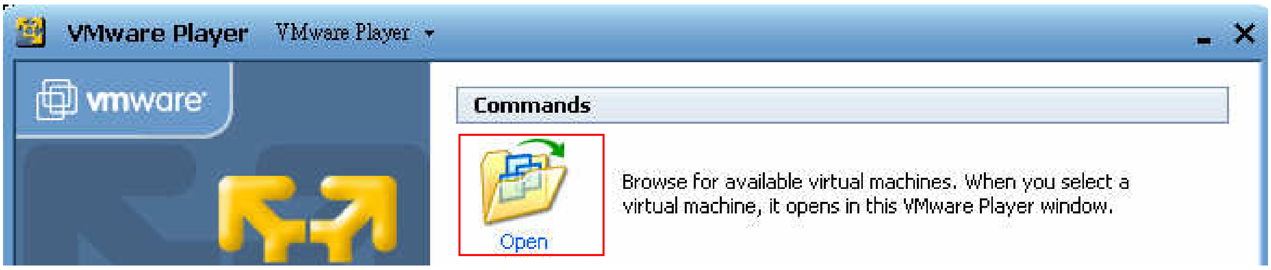
2.1.2 The login ID and password:

ID: root

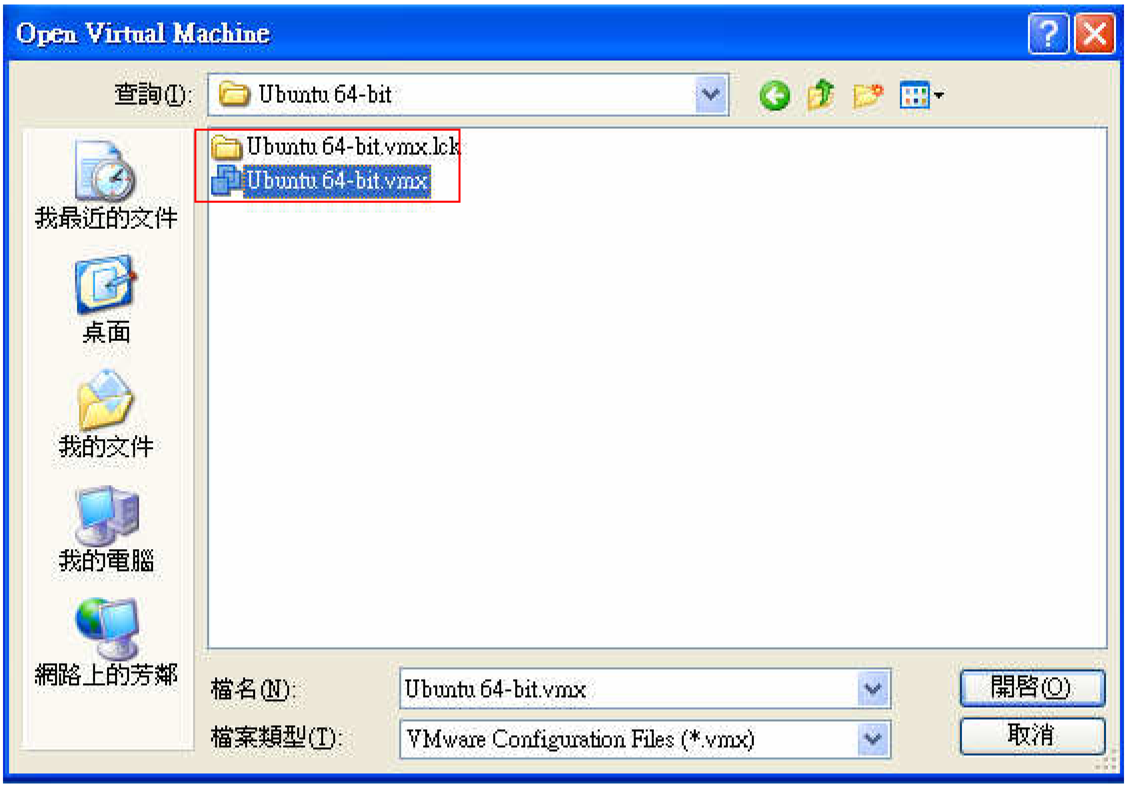
Password: 123456

2.1.3 Starting VMware Player (this document is using Vmware Player for Windows 7 for illustration):

a. Open Ubuntu 64-bit.vmdk



Browse for available virtual machines. When you select a virtual machine, it opens in this VMware Player window.

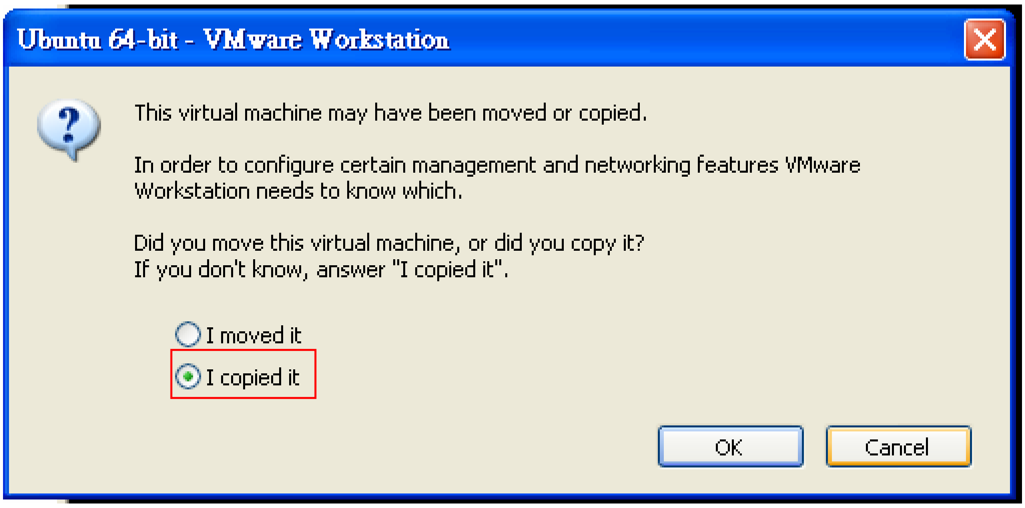


檔名(N): Ubuntu 64-bit.vmx

檔案類型(T): VMware Configuration Files (*.vmx)

b. After VMDK imported, the configuration is as below
Memory：1024MB
Hard Disk：80 GB
Network Adapter：Bridged
USB Controller：Present
Sound Card：Auto detect
Display：Auto detect
Processors：1

c. Click ”I copied it”



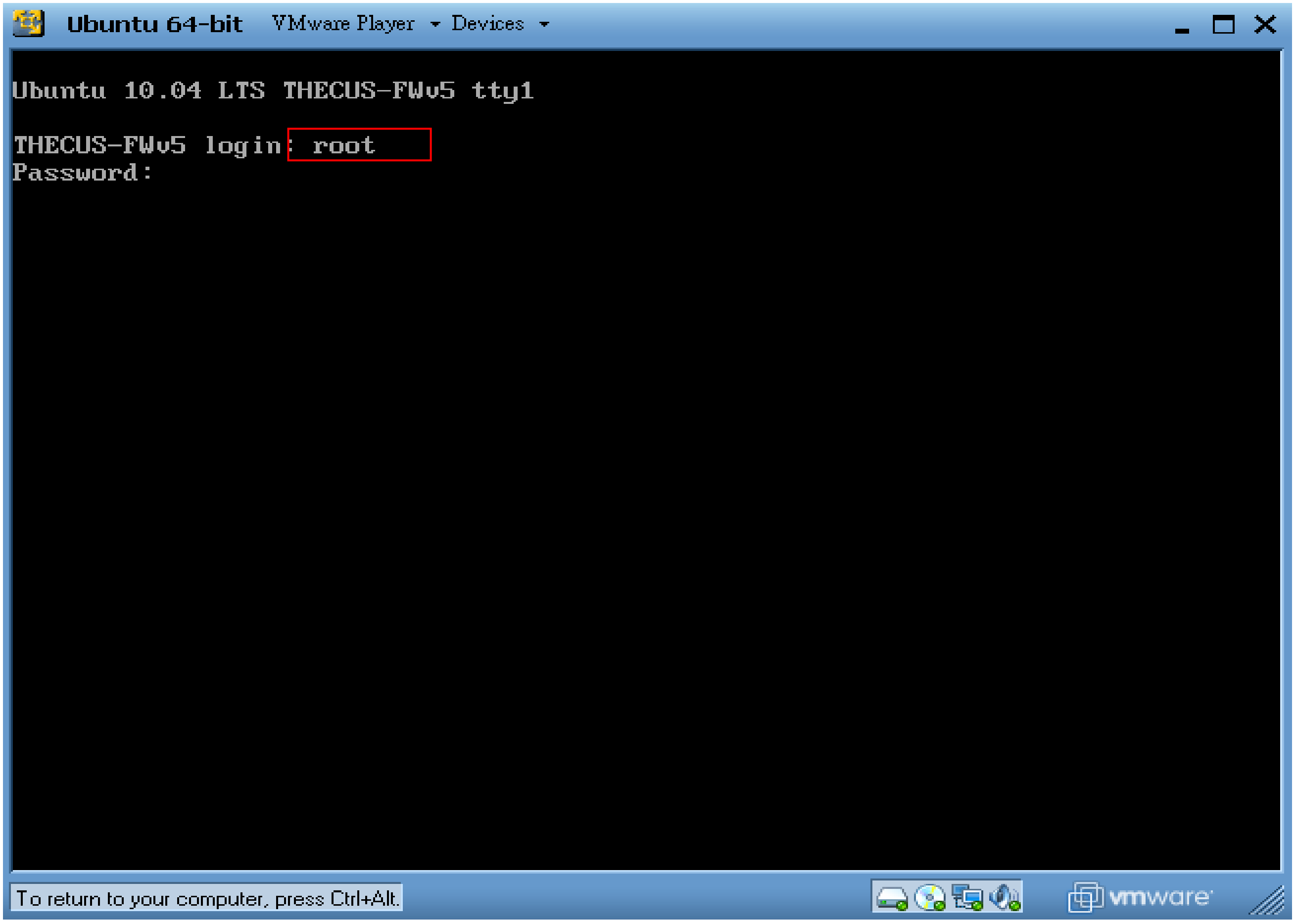
This virtual machine may have been moved or copied.

In order to configure certain management and networking features VMware Workstation needs to know which.

Did you move this virtual machine, or did you copy it?
If you don't know, answer "I copied it".

○ I moved it
● I copied it

d. Login into the Ubuntu. ID: root, Password: 123456



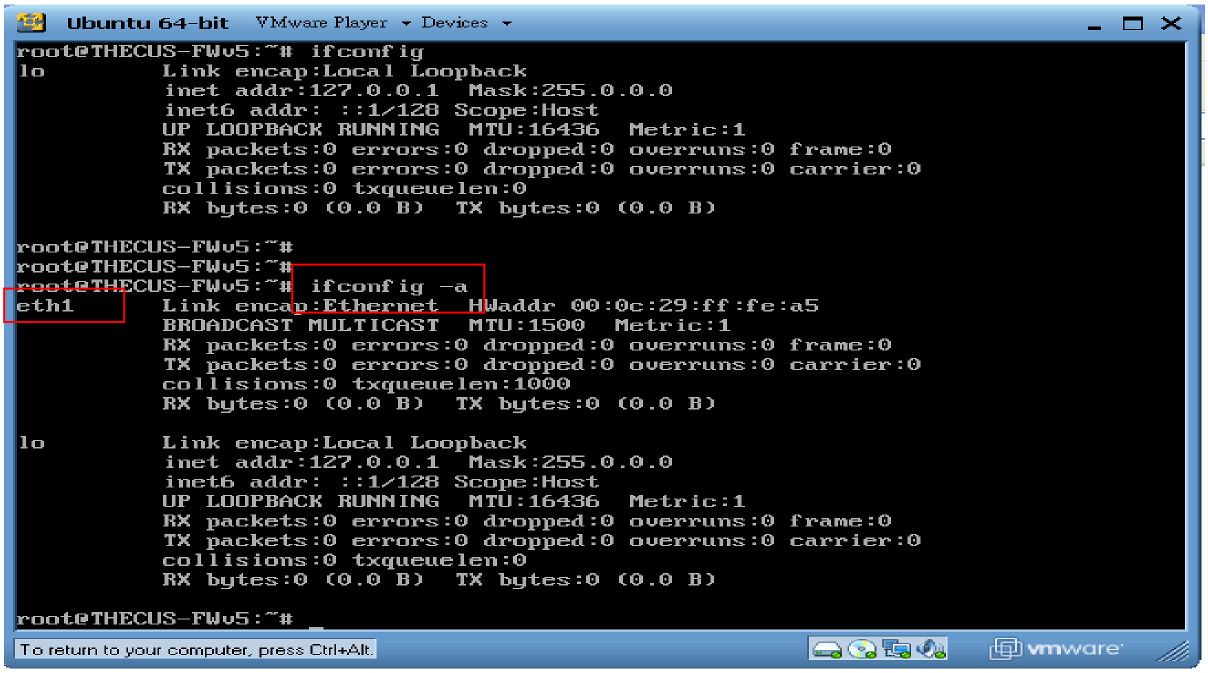
```
Ubuntu 10.04 LTS THECUS-FWv5 tty1

THECUS-FWv5 login: root
Password:
```

To return to your computer, press Ctrl+Alt.

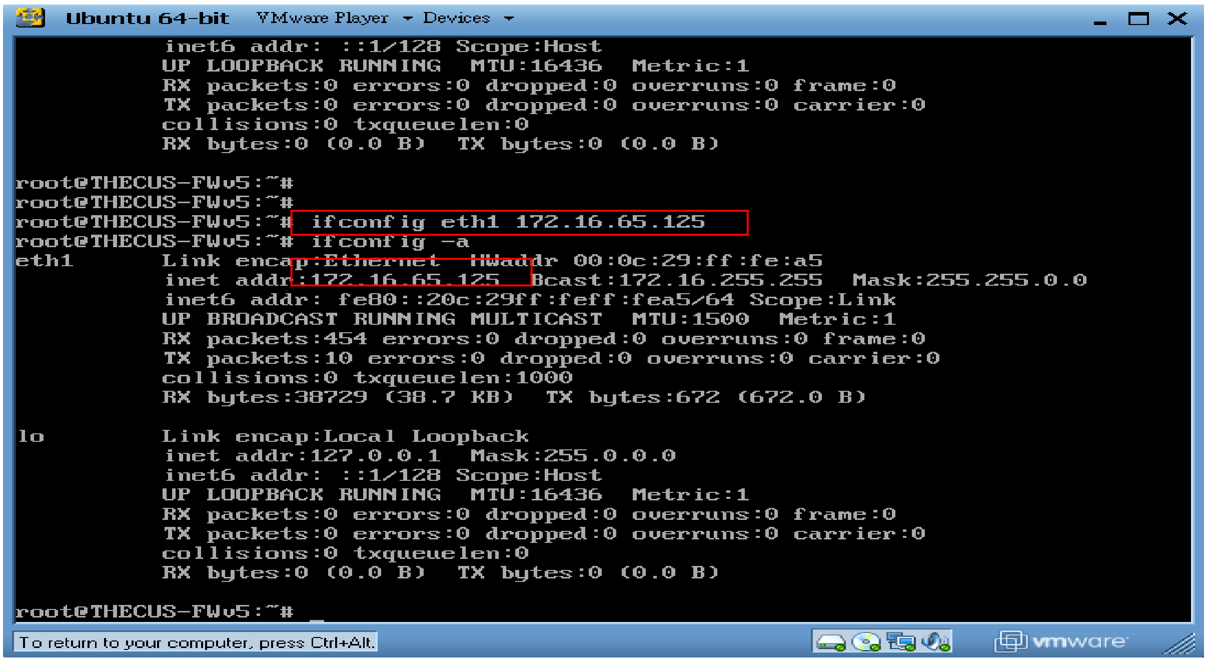
e. Config the network settings. Firstly, use (ifconfig -a) to list the current network interface. And then set the IP address (ifconfig interface xxx.xxx.xxx.xxx)

PS: It will be DHCP client by default. If you prefer a static IP address, go for this step. Also, you can modify /etc/network/interfaces thus it will be static IP address every time the guest OS boots up.

```
auto interface_name
iface interface_name inet static
    address xxx.xxx.xxx.xxx
    netmask xxx.xxx.xxx.xxx
    up flush-mail
```

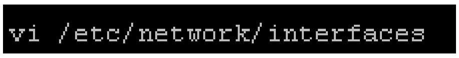


```
root@THECUS-FWv5:~# ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@THECUS-FWv5:~#
root@THECUS-FWv5:~#
root@THECUS-FWv5:~# ifconfig -a
eth1      Link encap:Ethernet  HWaddr 00:0c:29:ff:fe:a5
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@THECUS-FWv5:~# _
```



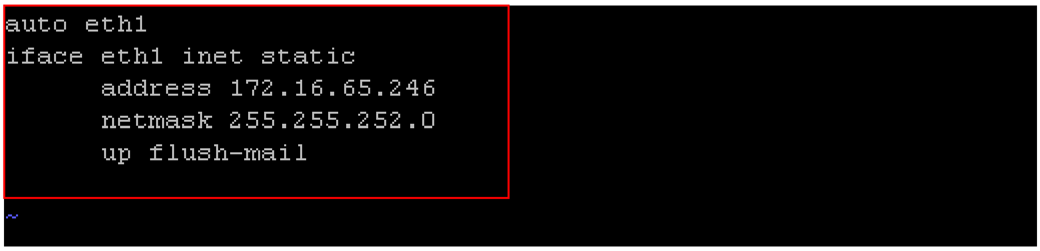
```
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@THECUS-FWv5:~#
root@THECUS-FWv5:~#
root@THECUS-FWv5:~# ifconfig eth1 172.16.65.125
root@THECUS-FWv5:~# ifconfig -a
eth1      Link encap:Ethernet  HWaddr 00:0c:29:ff:fe:a5
          inet addr:172.16.65.125  Bcast:172.16.255.255  Mask:255.255.0.0
          inet6 addr: fe80::20c:29ff:feff:fea5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:454 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:38729 (38.7 KB)  TX bytes:672 (672.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@THECUS-FWv5:~# _
```

To have static IP address every time the guest OS boots up:



```
vi /etc/network/interfaces
```

Edit the content



```
auto eth1
iface eth1 inet static
      address 172.16.65.246
      netmask 255.255.252.0
      up flush-mail

~
```

The environment is setup through steps a to e.

PS: This VMDK includes necessary Toolchain and profile file already.

2.2. 64bit Models

2.2.1Environment of VMDK:

Model	Environment
N12000/N16000	VMware Player running in 64 bit OS (Version 3.1.0 and above) Please note the 64 bit VMDK just works when the VM host is 64 bit and also the CPU supports VT.

2.2.2 Login ID and Password

ID: root

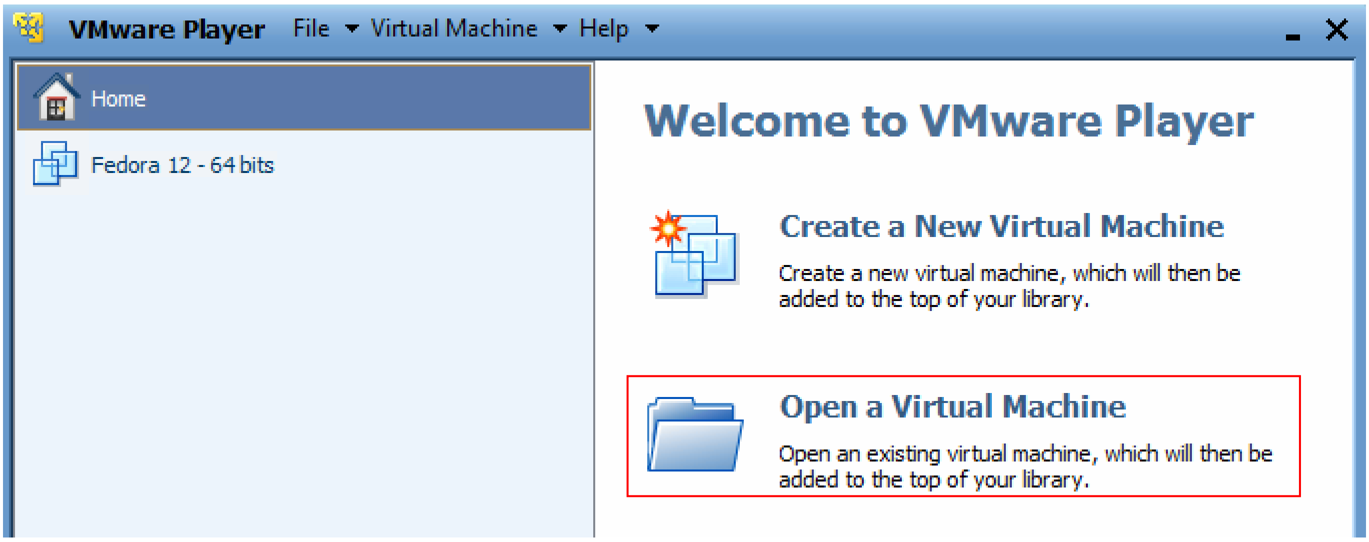
Password: 123456

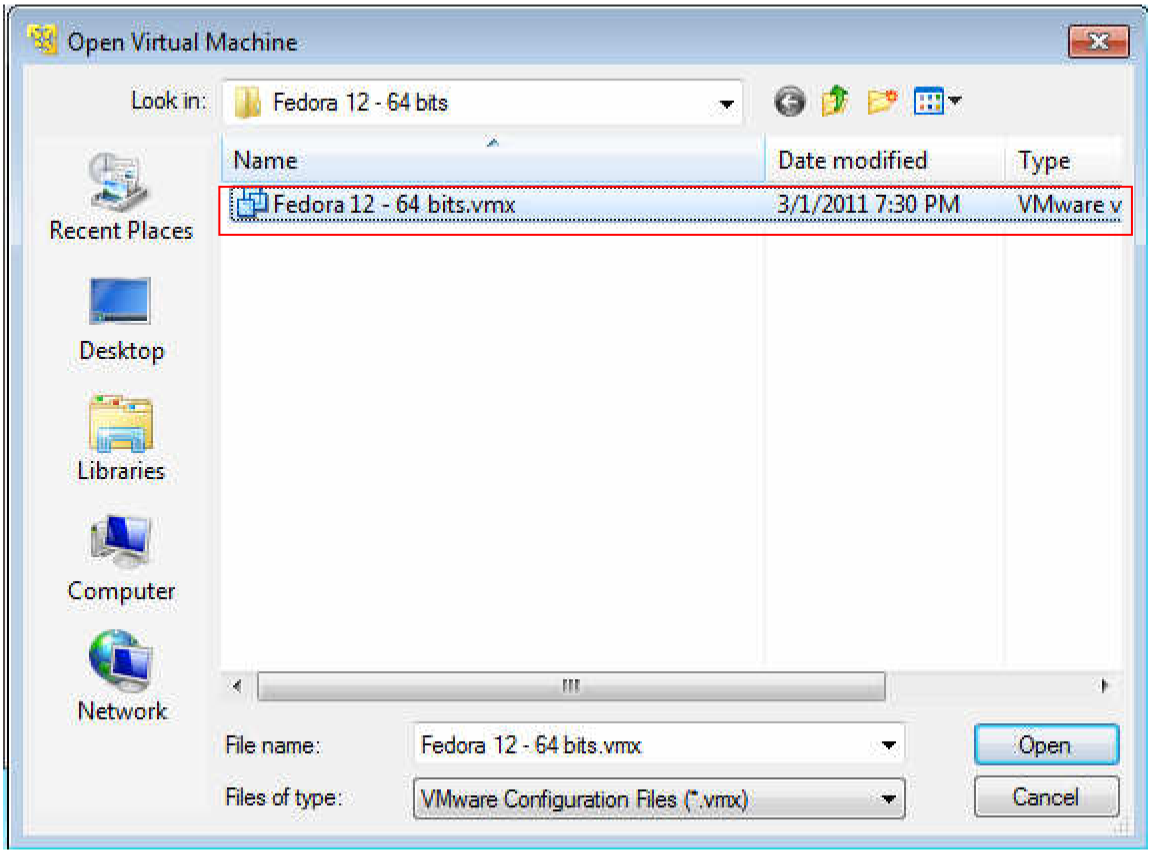
ID: admin

Password: admin

2.2.3 Starting VMware Player (this document is using VMware Player for Windows 7 for illustration):

a. Import the Fedora 12 - 64 bits.vmdk


VMware Player
File
Virtual Machine
Help
Home
Fedora 12 - 64 bits
Welcome to VMware Player
Create a New Virtual Machine
Create a new virtual machine, which will then be added to the top of your library.
Open a Virtual Machine
Open an existing virtual machine, which will then be added to the top of your library.



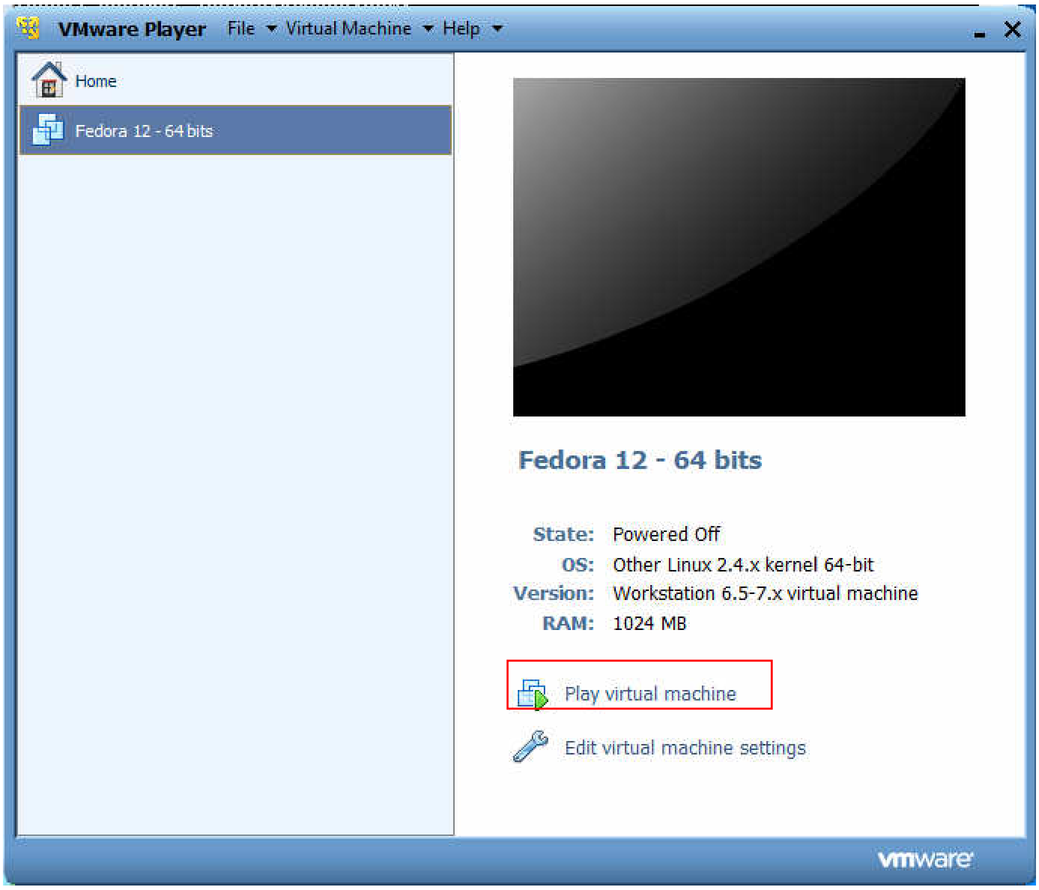
Look in: Fedora 12 - 64 bits

Name	Date modified	Type
Fedora 12 - 64 bits.vmx	3/1/2011 7:30 PM	VMware v

File name: Fedora 12 - 64 bits.vmx

Files of type: VMware Configuration Files (*.vmx)

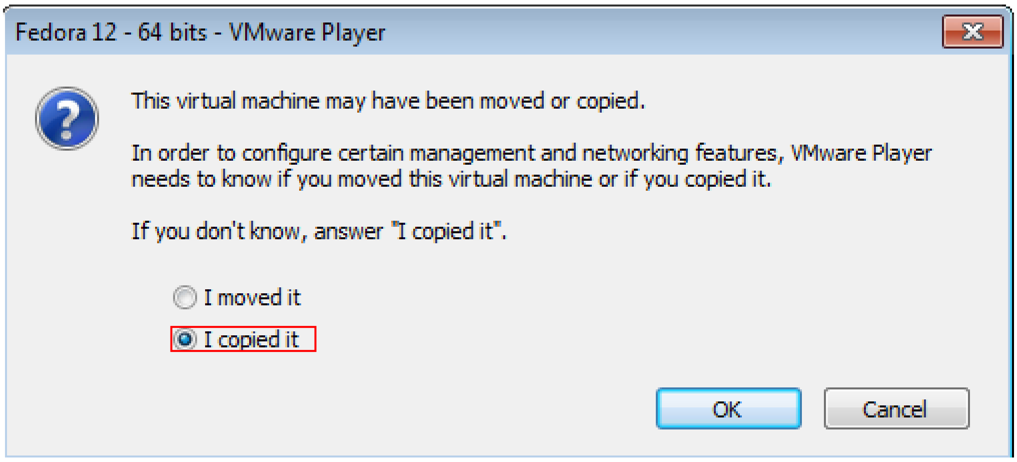
b. Click Play virtual machine



Fedora 12 - 64 bits

State: Powered Off
OS: Other Linux 2.4.x kernel 64-bit
Version: Workstation 6.5-7.x virtual machine
RAM: 1024 MB

c. Click “I copied it”



Fedora 12 - 64 bits - VMware Player

This virtual machine may have been moved or copied.

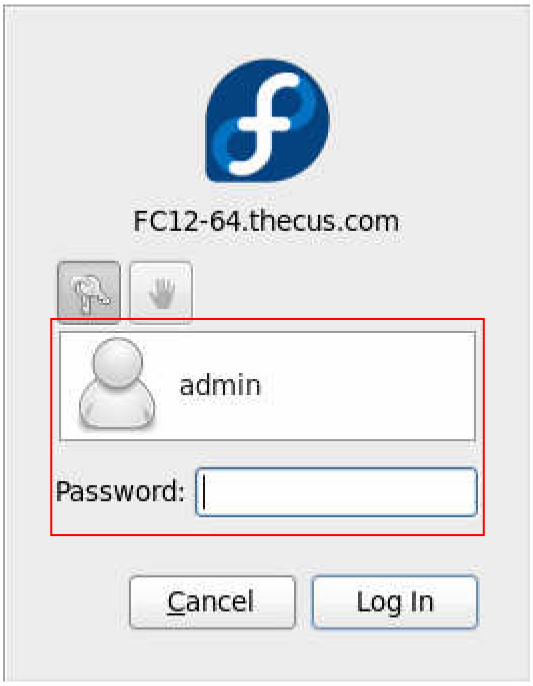
In order to configure certain management and networking features, VMware Player needs to know if you moved this virtual machine or if you copied it.

If you don't know, answer "I copied it".

○ I moved it

● I copied it

d. Login into the FC12: ID: admin, password: admin



FC12-64.thecus.com

admin

Password:

The environment is setup through steps a to d.

B、 Compile Applications (32bit Models)

1. Tool Chain

Some user modules need specific applications. In that case, you have to build them by Toolchain.

File: sw1_toolchain_1000912.tar.gz or newer available at Thecus FTP site. This document will use this file name for illustration.

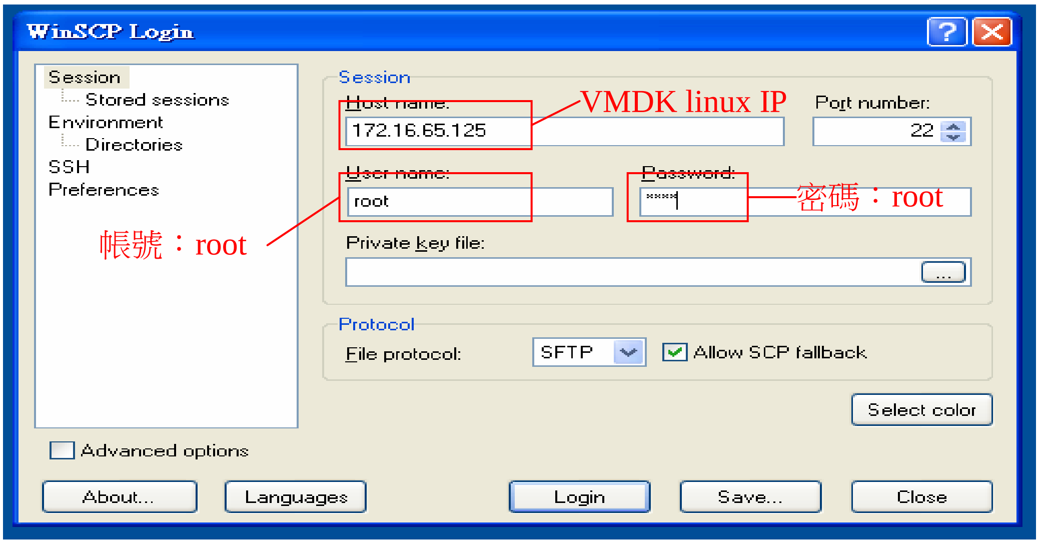
How to get Toolchain installed

1.1 sw1_toolchain_1000912.tar.gz or newer is available at Thecus FTP site:

http://ftp.thecus.com/module/category-1/toolchain/

1.2 Upload sw1_toolchain_1000912.tar.gz or newer to /usr/local/ in Linux (guest OS by VMDK). Either winscp (for Windows) or scp (for Linux) can do it.

Winscp



WinSCP Login

Session

Host name: 172.16.65.125

Port number: 22

User name: root

Password: *****

Private key file:

Protocol

File protocol: SFTP

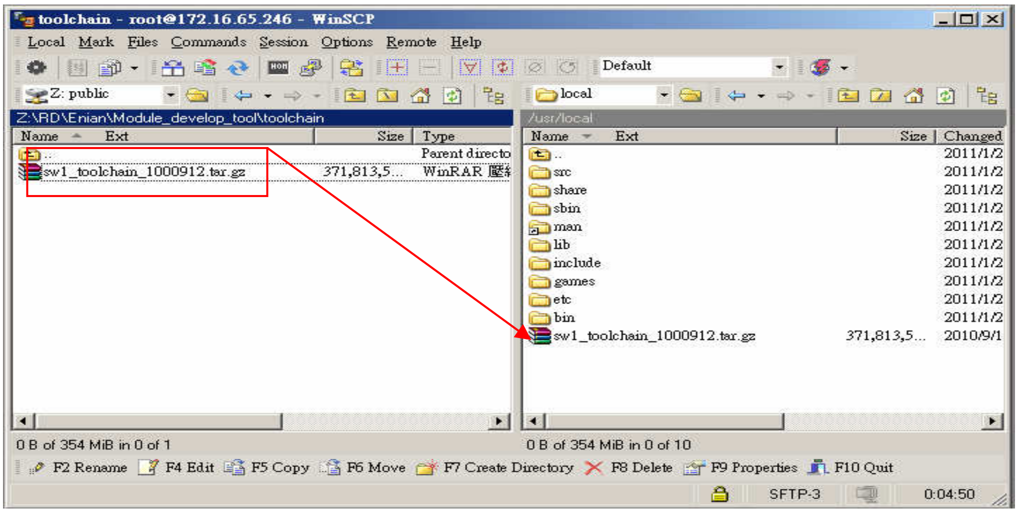
☑ Allow SCP fallback

□ Advanced options

VMDK linux IP

帳號：root

密碼：root



toolchain - root@172.16.65.246 - WinSCP

Z:\RD\Enian\Module_develop_tool\toolchain

Name	Ext	Size	Type
..			Parent directo
sw1_toolchain_1000912.tar.gz		371,813,5...	WinRAR 壓縮

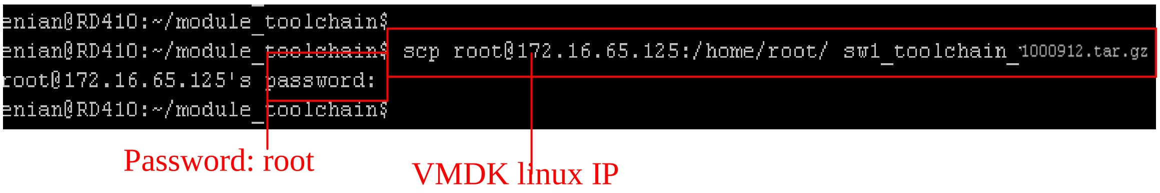
0 B of 354 MiB in 0 of 1

/usr/local

Name	Ext	Size	Changed
..			2011/1/2
src			2011/1/2
share			2011/1/2
sbin			2011/1/2
man			2011/1/2
lib			2011/1/2
include			2011/1/2
games			2011/1/2
etc			2011/1/2
bin			2011/1/2
sw1_toolchain_1000912.tar.gz		371,813,5...	2010/9/1

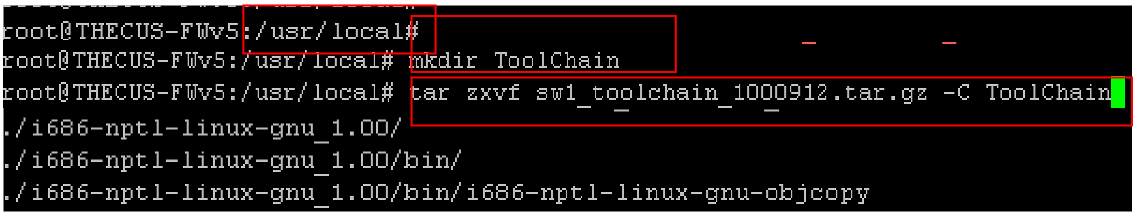
0 B of 354 MiB in 0 of 10

Linux Command

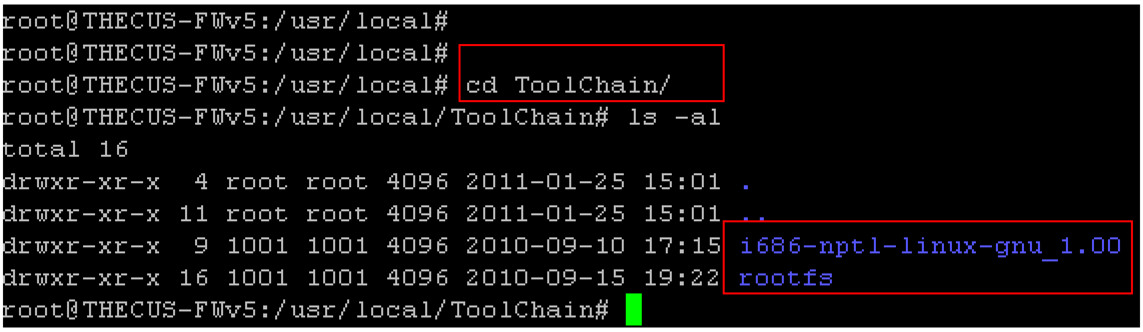


```
enian@RD410:~/module_toolchain$
enian@RD410:~/module_toolchain$ scp root@172.16.65.125:/home/root/ sw1_toolchain_1000912.tar.gz
root@172.16.65.125's password:
enian@RD410:~/module_toolchain$
```

1.3 Make a folder ToolChain in /usr/local. You can do it by putty (in Windows), ssh (in Linux), or from VMware player. Then type tar zxvf sw1_toolchain_1000912.tar.gz -C ToolChain. Two folders will be generated under ToolChain: i686-nptl-linux-gnu_1.00 and rootfs

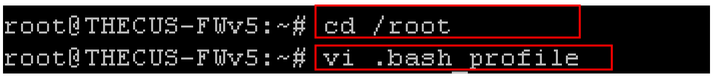


```
root@THECUS-FWv5:/usr/local#
root@THECUS-FWv5:/usr/local# mkdir ToolChain
root@THECUS-FWv5:/usr/local# tar zxvf sw1_toolchain_1000912.tar.gz -C ToolChain
./i686-nptl-linux-gnu_1.00/
./i686-nptl-linux-gnu_1.00/bin/
./i686-nptl-linux-gnu_1.00/bin/i686-nptl-linux-gnu-objcopy
```



```
root@THECUS-FWv5:/usr/local#
root@THECUS-FWv5:/usr/local#
root@THECUS-FWv5:/usr/local# cd ToolChain/
root@THECUS-FWv5:/usr/local/ToolChain# ls -al
total 16
drwxr-xr-x  4 root root 4096 2011-01-25 15:01 .
drwxr-xr-x 11 root root 4096 2011-01-25 15:01 ..
drwxr-xr-x  9 1001 1001 4096 2010-09-10 17:15 i686-nptl-linux-gnu_1.00
drwxr-xr-x 16 1001 1001 4096 2010-09-15 19:22 rootfs
root@THECUS-FWv5:/usr/local/ToolChain#
```

1.4 Change to /root, and then vi .bash_profile. It will add .bash_profile file for some environmental variables. The details will be explained in the following section. After .bash_profile created, re-login to the VMDK Linux.



```
root@THECUS-FWv5:~# cd /root
root@THECUS-FWv5:~# vi .bash_profile
```

PS: The library of NAS firmware is located at /usr/local/ToolChain/rootfs/lib or /usr/local/ToolChain/rootfs/opt/lib. If applications need them, please link to these two paths.

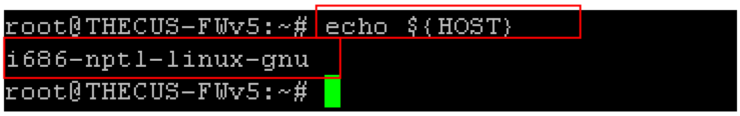
2. Set the Environmental Variables

```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup programs

HOST=i686-nptl-linux-gnu
export HOST

TOOLCHAIN_PATH=/usr/local/ToolChain/i686-nptl-linux-gnu_1.00
export TOOLCHAIN_PATH

ROOTFS_PATH=/usr/local/ToolChain/rootfs
export ROOTFS_PATH

OPT_PATH=/opt
export OPT_PATH

OPTEXTERN_PATH=${ROOTFS_PATH}/opt_extern
export OPTEXTERN_PATH

PATH=$PATH:$HOME/bin:${TOOLCHAIN_PATH}/bin:${TOOLCHAIN_PATH}/i686-nptl-linux-gnu/bin:${ROOTFS_PATH}/usr/bin
export PATH
```

Variable Name	Description
HOST	i686-nptl-linux-gnu
TOOLCHAIN_PATH	Set all compile tool path of Toolchain i686-nptl-linux-gnu_1.00 includes compile tool
ROOTFS_PATH	The rootfs path of Toolchain rootfs includes the necessary library or head files for application

.bash_profile is available at http://ftp.thecus.com/module/category-1/module/profile/ . So, you can upload it to /root by winscp (in Windows) or scp (in Linux).

PS: To confirm the environmental variables have been set, type echo $HOST. If gets the response: i686-nptl-linux-gnu, it's done.



```
root@THECUS-FWv5:~# echo ${HOST}
i686-nptl-linux-gnu
root@THECUS-FWv5:~#
```

3. Compile a program

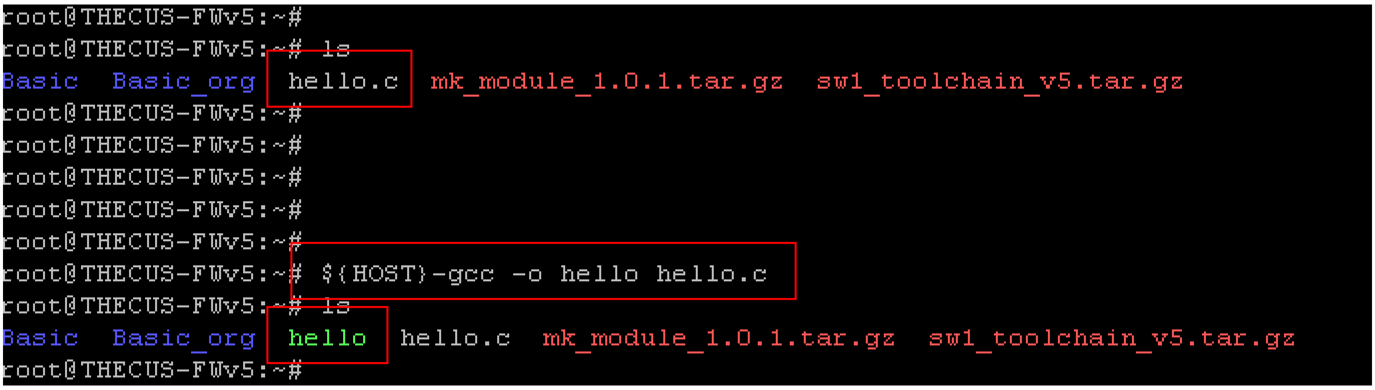
This section will talk about compiling a binary file by using ToolChain. The example is making a binary and executable “hello world”. (hello.c is available at http://ftp.thecus.com/module/category-1/module/hello_example/)

Procedure:

3.1 Make a hello.c in /root

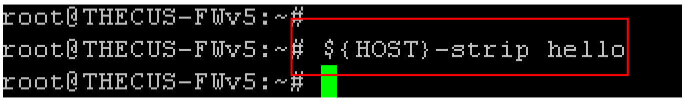
3.2 Type ${HOST}-gcc –o hello hello.c in /root

3.3 hello binary will be generated



```
root@THECUS-FWv5:~#
root@THECUS-FWv5:~# ls
Basic  Basic_org  hello.c  mk_module_1.0.1.tar.gz  sw1_toolchain_v5.tar.gz
root@THECUS-FWv5:~#
root@THECUS-FWv5:~#
root@THECUS-FWv5:~#
root@THECUS-FWv5:~#
root@THECUS-FWv5:~#
root@THECUS-FWv5:~# ${HOST}-gcc -o hello hello.c
root@THECUS-FWv5:~# ls
Basic  Basic_org  hello  hello.c  mk_module_1.0.1.tar.gz  sw1_toolchain_v5.tar.gz
root@THECUS-FWv5:~#
```

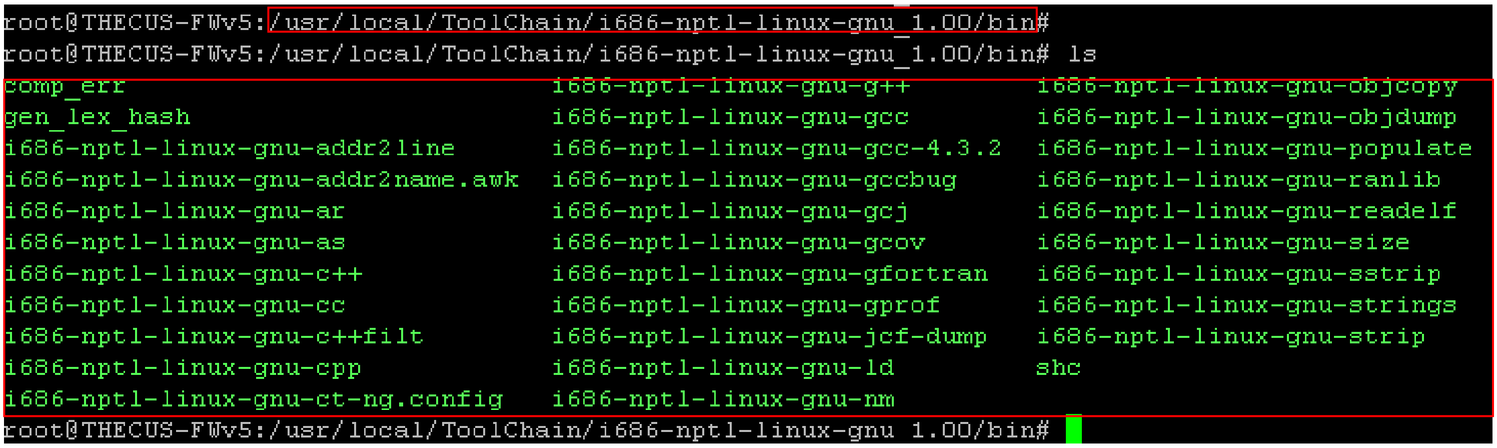
3.4 Remove unnecessary symbols from the binary by typing ${HOST}-strip hello



```
root@THECUS-FWv5:~#
root@THECUS-FWv5:~# ${HOST}-strip hello
root@THECUS-FWv5:~#
```

PS: If need the other compile tool such as g++, c++, or cpp, change ${HOST}-gcc to ${HOST}-g++

Compile tools are located at /usr/local/ToolChain/i686-nptl-linux-gnu_1.00/bin



```
root@THECUS-FWv5:/usr/local/ToolChain/i686-nptl-linux-gnu_1.00/bin#
root@THECUS-FWv5:/usr/local/ToolChain/i686-nptl-linux-gnu_1.00/bin# ls
comp_err                            i686-nptl-linux-gnu-g++          i686-nptl-linux-gnu-objcopy
gen_lex_hash                        i686-nptl-linux-gnu-gcc          i686-nptl-linux-gnu-objdump
i686-nptl-linux-gnu-addr2line       i686-nptl-linux-gnu-gcc-4.3.2    i686-nptl-linux-gnu-populate
i686-nptl-linux-gnu-addr2name.awk   i686-nptl-linux-gnu-gccbug       i686-nptl-linux-gnu-ranlib
i686-nptl-linux-gnu-ar              i686-nptl-linux-gnu-gcj          i686-nptl-linux-gnu-readelf
i686-nptl-linux-gnu-as              i686-nptl-linux-gnu-gcov         i686-nptl-linux-gnu-size
i686-nptl-linux-gnu-c++             i686-nptl-linux-gnu-gfortran     i686-nptl-linux-gnu-sstrip
i686-nptl-linux-gnu-cc              i686-nptl-linux-gnu-gprof        i686-nptl-linux-gnu-strings
i686-nptl-linux-gnu-c++filt         i686-nptl-linux-gnu-jcf-dump     i686-nptl-linux-gnu-strip
i686-nptl-linux-gnu-cpp             i686-nptl-linux-gnu-ld           shc
i686-nptl-linux-gnu-ct-ng.config    i686-nptl-linux-gnu-nm
root@THECUS-FWv5:/usr/local/ToolChain/i686-nptl-linux-gnu_1.00/bin#
```

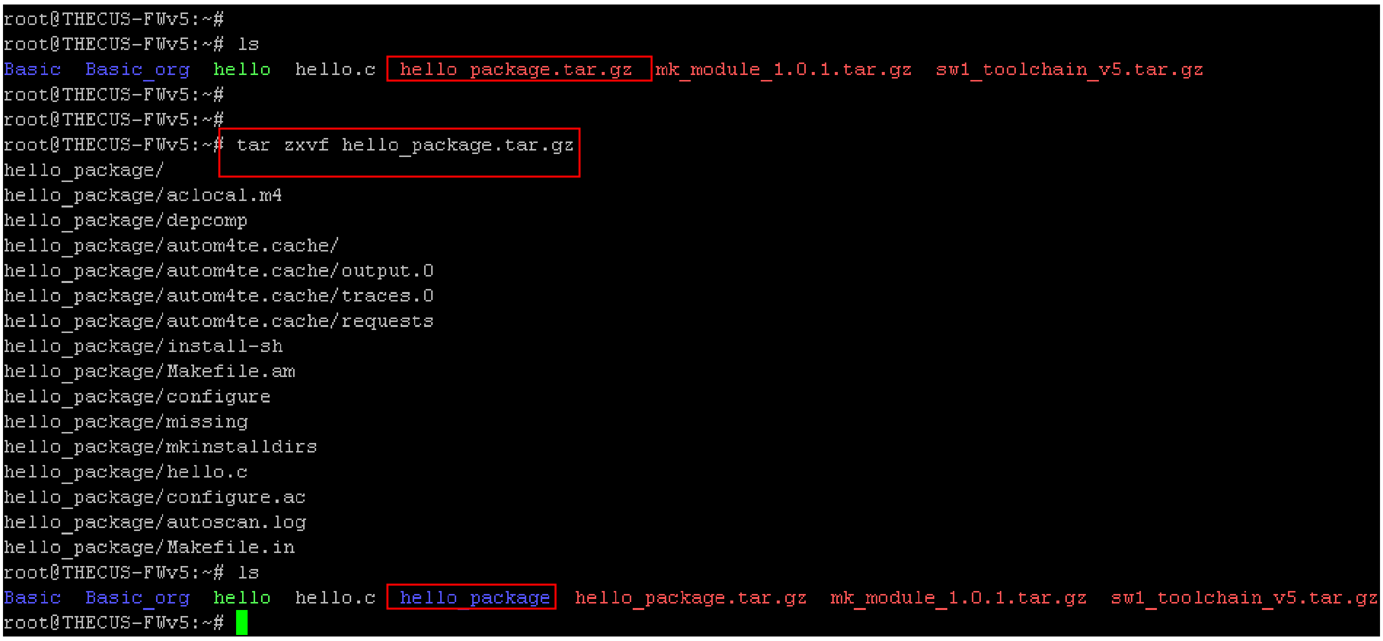
4. Compile open source application

This section will talk about compiling an open source application by using ToolChain. The example is making a package of “hello world”.

Procedure:

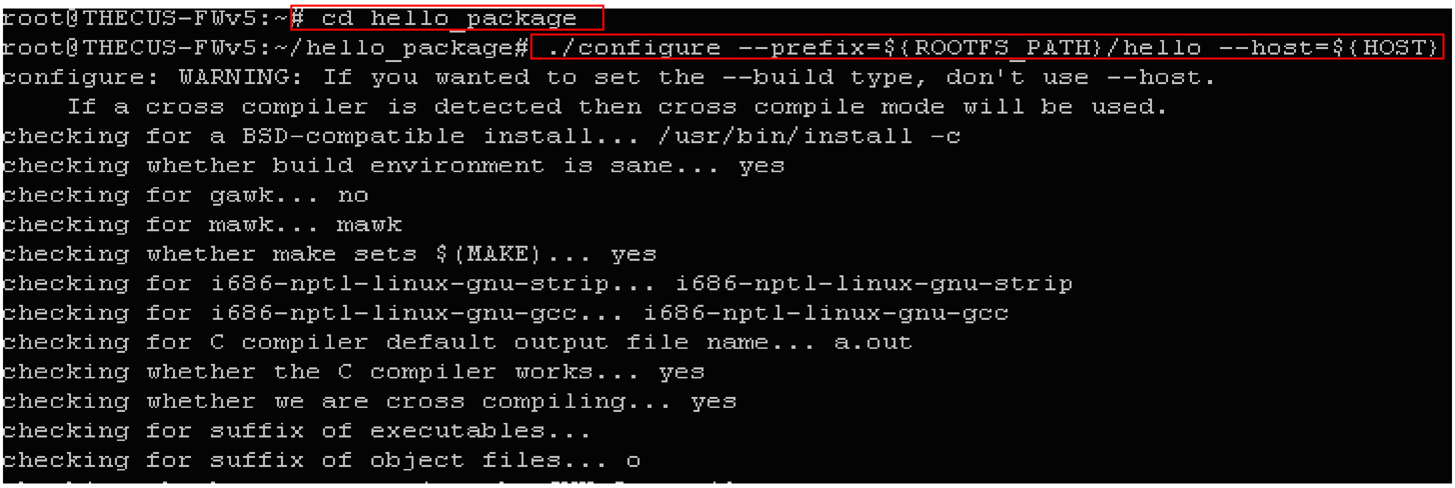
4.1 Upload hello_package.tar.gz (available at http://ftp.thecus.com/module/category-1/module/hello_example/) to /root by Winscp (in Windows) or scp (in Linux)

4.2 Type tar zxvf hello_package.tar.gz in /root to get a folder hello_package



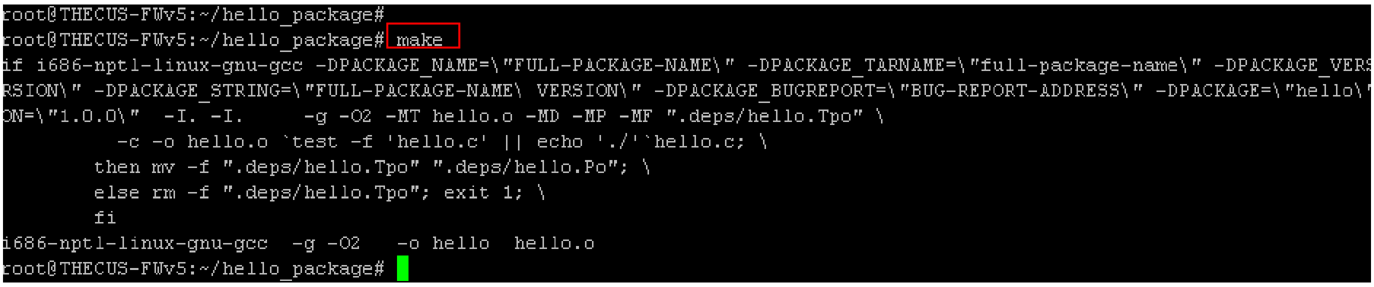
```
root@THECUS-FWv5:~#
root@THECUS-FWv5:~# ls
Basic  Basic_org  hello  hello.c  hello_package.tar.gz  mk_module_1.0.1.tar.gz  sw1_toolchain_v5.tar.gz
root@THECUS-FWv5:~#
root@THECUS-FWv5:~#
root@THECUS-FWv5:~# tar zxvf hello_package.tar.gz
hello_package/
hello_package/aclocal.m4
hello_package/depcomp
hello_package/autom4te.cache/
hello_package/autom4te.cache/output.0
hello_package/autom4te.cache/traces.0
hello_package/autom4te.cache/requests
hello_package/install-sh
hello_package/Makefile.am
hello_package/configure
hello_package/missing
hello_package/mkinstalldirs
hello_package/hello.c
hello_package/configure.ac
hello_package/autoscan.log
hello_package/Makefile.in
root@THECUS-FWv5:~# ls
Basic  Basic_org  hello  hello.c  hello_package  hello_package.tar.gz  mk_module_1.0.1.tar.gz  sw1_toolchain_v5.tar.gz
root@THECUS-FWv5:~#
```

4.3 Type ./configure --prefix=${ROOTFS_PATH}/hello --host=${HOST} in hello_package folder (prefix stands for the path where the application will be installed to. For complicated application, set prefix to /raid/data/module/module_folder_name)



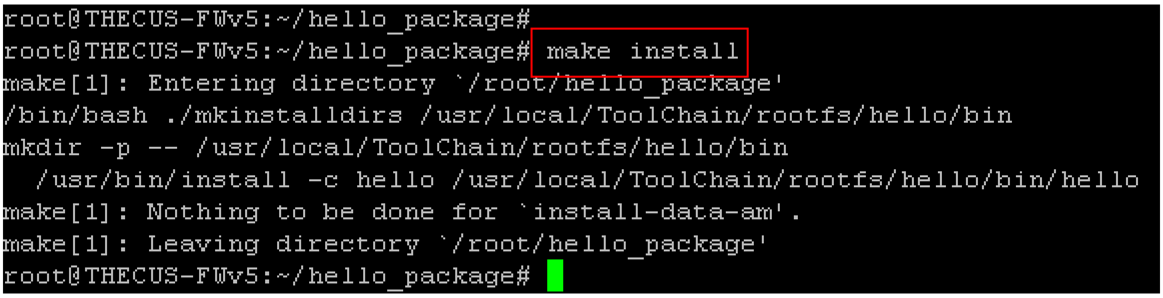
```
root@THECUS-FWv5:~# cd hello_package
root@THECUS-FWv5:~/hello_package# ./configure --prefix=${ROOTFS_PATH}/hello --host=${HOST}
configure: WARNING: If you wanted to set the --build type, don't use --host.
    If a cross compiler is detected then cross compile mode will be used.
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE)... yes
checking for i686-nptl-linux-gnu-strip... i686-nptl-linux-gnu-strip
checking for i686-nptl-linux-gnu-gcc... i686-nptl-linux-gnu-gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... yes
checking for suffix of executables...
checking for suffix of object files... o
```

4.4 make



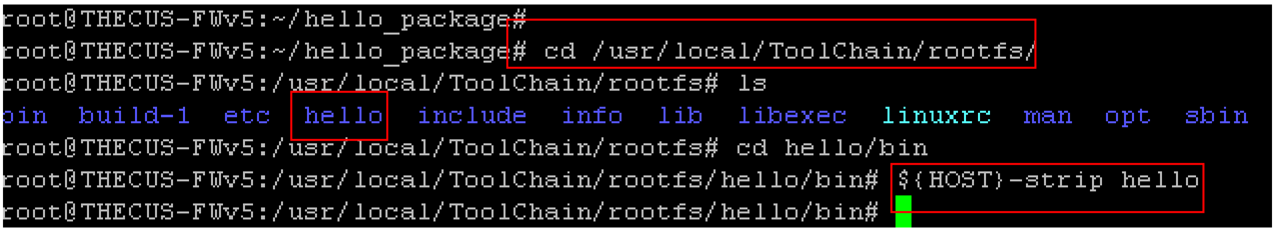
```
root@THECUS-FWv5:~/hello_package#
root@THECUS-FWv5:~/hello_package# make
if i686-nptl-linux-gnu-gcc -DPACKAGE_NAME=\"FULL-PACKAGE-NAME\" -DPACKAGE_TARNAME=\"full-package-name\" -DPACKAGE_VERS
RSION\" -DPACKAGE_STRING=\"FULL-PACKAGE-NAME\ VERSION\" -DPACKAGE_BUGREPORT=\"BUG-REPORT-ADDRESS\" -DPACKAGE=\"hello\"
ON=\"1.0.0\"  -I. -I.     -g -O2 -MT hello.o -MD -MP -MF ".deps/hello.Tpo" \
          -c -o hello.o `test -f 'hello.c' || echo './'`hello.c; \
        then mv -f ".deps/hello.Tpo" ".deps/hello.Po"; \
        else rm -f ".deps/hello.Tpo"; exit 1; \
        fi
i686-nptl-linux-gnu-gcc  -g -O2   -o hello  hello.o
root@THECUS-FWv5:~/hello_package#
```

4.5 make install



```
root@THECUS-FWv5:~/hello_package#
root@THECUS-FWv5:~/hello_package# make install
make[1]: Entering directory `/root/hello_package'
/bin/bash ./mkinstalldirs /usr/local/ToolChain/rootfs/hello/bin
mkdir -p -- /usr/local/ToolChain/rootfs/hello/bin
  /usr/bin/install -c hello /usr/local/ToolChain/rootfs/hello/bin/hello
make[1]: Nothing to be done for `install-data-am'.
make[1]: Leaving directory `/root/hello_package'
root@THECUS-FWv5:~/hello_package#
```

4.6 get into /usr/local/ToolChain/rootfs/hello/bin, and then do ${HOST}-strip hello



```
root@THECUS-FWv5:~/hello_package#
root@THECUS-FWv5:~/hello_package# cd /usr/local/ToolChain/rootfs/
root@THECUS-FWv5:/usr/local/ToolChain/rootfs# ls
bin  build-1  etc  hello  include  info  lib  libexec  linuxrc  man  opt  sbin
root@THECUS-FWv5:/usr/local/ToolChain/rootfs# cd hello/bin
root@THECUS-FWv5:/usr/local/ToolChain/rootfs/hello/bin# ${HOST}-strip hello
root@THECUS-FWv5:/usr/local/ToolChain/rootfs/hello/bin#
```

PS: 1. When application is just for library application, direct the prefix to /raid/data/module/[module folder name]/sys

For example

./configure --prefix=/raid/data/module/Basic/sys --host=${HOST}

2. When application is an executable binary, direct the prefix to /raid/data/module/[module folder name]/bin

For example

/configure --prefix=/raid/data/module/Basic/bin --host=${HOST}

3. When application needs the other library, you have to install them before pack the application; like above step 1. Also, direct the library path to /raid/data/module/[module folder name]/sys/lib

Please note the application libraries are in lib folder in most cases; but not always.

For example

LDFLAGS=”-L/raid/data/module/Basic/sys/lib -lhello” LIBS=”-L/raid/data/module/Basic/sys/lib -lhello” ./configure --prefix=/raid/data/module/Basic/bin --host=${HOST}

4. A sample

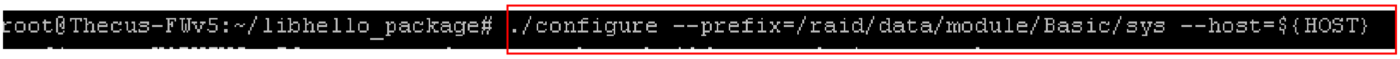
Module ‘Basic’ needs an executable binary hello_test, and hello_test needs libhello.a

Step 1) install libhello first:

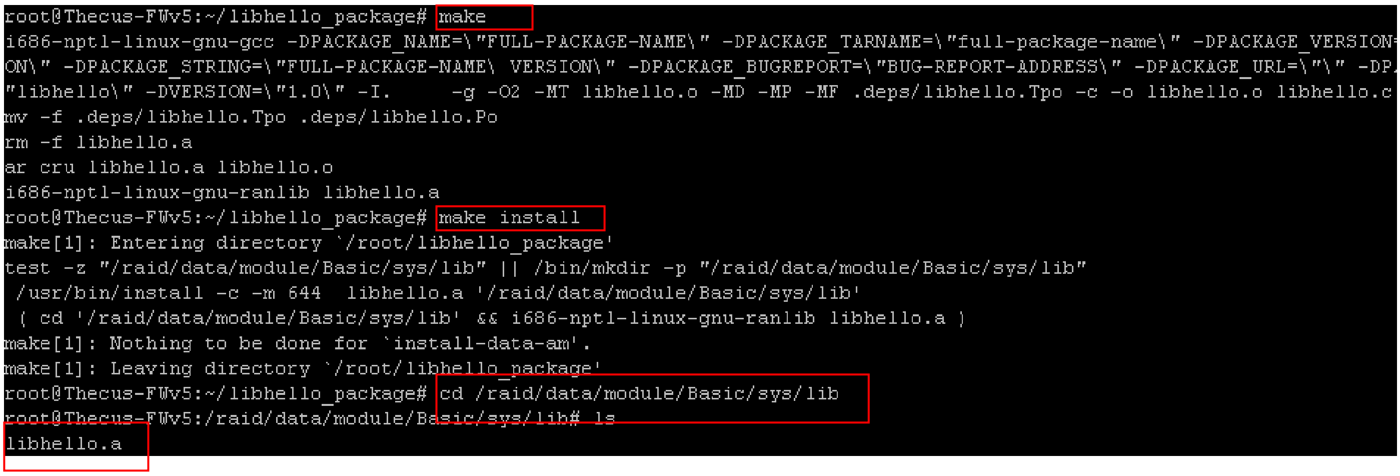
a ./configure --prefix=/raid/data/module/Basic/sys --host=${HOST}

b. make

c. make install



```
root@Thecus-FWv5:~/libhello_package# ./configure --prefix=/raid/data/module/Basic/sys --host=${HOST}
```



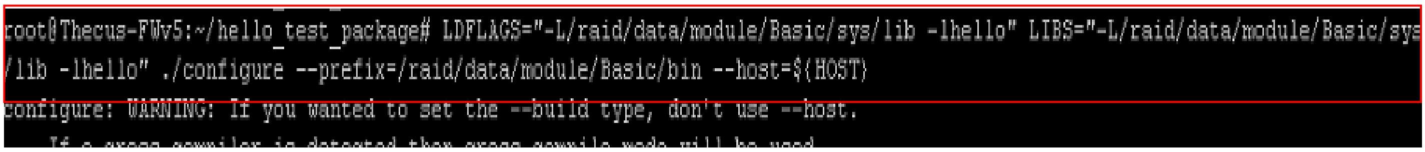
```
root@Thecus-FWv5:~/libhello_package# make
i686-nptl-linux-gnu-gcc -DPACKAGE_NAME=\"FULL-PACKAGE-NAME\" -DPACKAGE_TARNAME=\"full-package-name\" -DPACKAGE_VERSION=
ON\" -DPACKAGE_STRING=\"FULL-PACKAGE-NAME\ VERSION\" -DPACKAGE_BUGREPORT=\"BUG-REPORT-ADDRESS\" -DPACKAGE_URL=\"\" -DP
"libhello\" -DVERSION=\"1.0\" -I.     -g -O2 -MT libhello.o -MD -MP -MF .deps/libhello.Tpo -c -o libhello.o libhello.c
mv -f .deps/libhello.Tpo .deps/libhello.Po
rm -f libhello.a
ar cru libhello.a libhello.o
i686-nptl-linux-gnu-ranlib libhello.a
root@Thecus-FWv5:~/libhello_package# make install
make[1]: Entering directory `/root/libhello_package'
test -z "/raid/data/module/Basic/sys/lib" || /bin/mkdir -p "/raid/data/module/Basic/sys/lib"
 /usr/bin/install -c -m 644  libhello.a '/raid/data/module/Basic/sys/lib'
 ( cd '/raid/data/module/Basic/sys/lib' && i686-nptl-linux-gnu-ranlib libhello.a )
make[1]: Nothing to be done for `install-data-am'.
make[1]: Leaving directory `/root/libhello_package'
root@Thecus-FWv5:~/libhello_package# cd /raid/data/module/Basic/sys/lib
root@Thecus-FWv5:/raid/data/module/Basic/sys/lib# ls
libhello.a
```

Step 2) install hello:

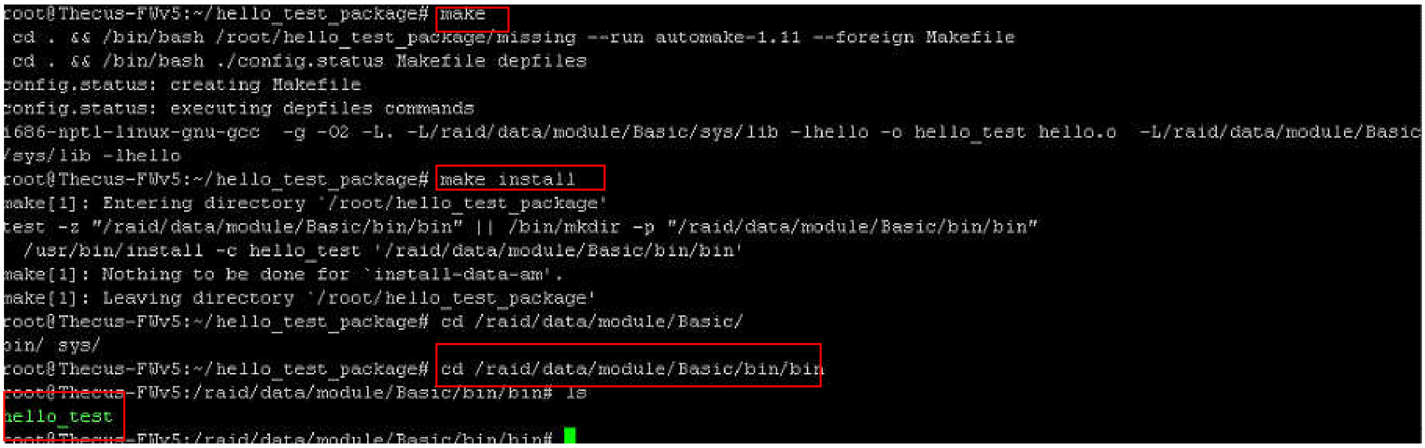
a. LDFLAGS="-L/raid/data/module/Basic/sys/lib -lhello" LIBS="-L/raid/data/module/Basic/sys/lib -lhello" ./configure --prefix=/raid/data/module/Basic/bin --host=${HOST}

b. make

c. make install



```
root@Thecus-FWv5:~/hello_test_package# LDFLAGS="-L/raid/data/module/Basic/sys/lib -lhello" LIBS="-L/raid/data/module/Basic/sys
/lib -lhello" ./configure --prefix=/raid/data/module/Basic/bin --host=${HOST}
configure: WARNING: If you wanted to set the --build type, don't use --host.
    If a cross compiler is detected then cross compile mode will be used.
```



```
root@Thecus-FWv5:~/hello_test_package# make
 cd . && /bin/bash /root/hello_test_package/missing --run automake-1.11 --foreign Makefile
 cd . && /bin/bash ./config.status Makefile depfiles
config.status: creating Makefile
config.status: executing depfiles commands
i686-nptl-linux-gnu-gcc  -g -O2 -L. -L/raid/data/module/Basic/sys/lib -lhello -o hello_test hello.o  -L/raid/data/module/Basic
/sys/lib -lhello
root@Thecus-FWv5:~/hello_test_package# make install
make[1]: Entering directory `/root/hello_test_package'
test -z "/raid/data/module/Basic/bin/bin" || /bin/mkdir -p "/raid/data/module/Basic/bin/bin"
  /usr/bin/install -c hello_test '/raid/data/module/Basic/bin/bin'
make[1]: Nothing to be done for `install-data-am'.
make[1]: Leaving directory `/root/hello_test_package'
root@Thecus-FWv5:~/hello_test_package# cd /raid/data/module/Basic/
bin/ sys/
root@Thecus-FWv5:~/hello_test_package# cd /raid/data/module/Basic/bin/bin
root@Thecus-FWv5:/raid/data/module/Basic/bin/bin# ls
hello_test
root@Thecus-FWv5:/raid/data/module/Basic/bin/bin#
```

Step 3) Make sure the executable binary is working, and then put it together with the library files to correct path; which is defined in the session ‘Module Path’ of ’Module Developers Guide’.

For example

Put all the files and sub folders in /raid/data/module/Basic/sys/lib to the System/lib

under module folder; such as Basic/System/lib
Put all the files and sub folders in /raid/data/module/Basic/bin to Binary under module folder; such as Basic/Binary

C、 Compile Applications (64bit Model)

1. Compile a program

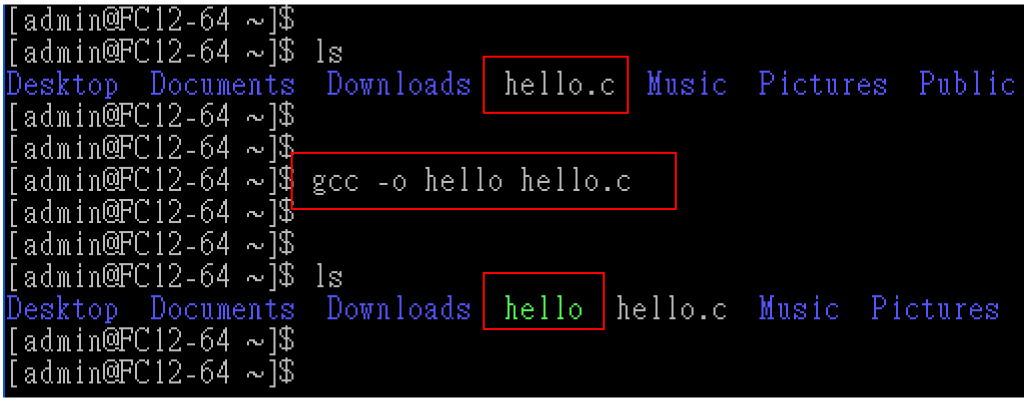
This section will talk about compiling a binary file. The example is making a binary and executable "hello world". (hello.c is available at http://ftp.thecus.com/module/category-1/module/hello_example/)

Procedure:

1.1 Make a hello.c in admin

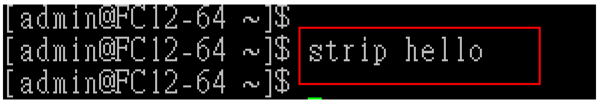
1.2 Type gcc –o hello hello.c in admin folder

1.3 hello binary will be generated



```
[admin@FC12-64 ~]$
[admin@FC12-64 ~]$ ls
Desktop  Documents  Downloads  hello.c  Music  Pictures  Public
[admin@FC12-64 ~]$
[admin@FC12-64 ~]$
[admin@FC12-64 ~]$ gcc -o hello hello.c
[admin@FC12-64 ~]$
[admin@FC12-64 ~]$
[admin@FC12-64 ~]$ ls
Desktop  Documents  Downloads  hello  hello.c  Music  Pictures
[admin@FC12-64 ~]$
[admin@FC12-64 ~]$
```

1.4 Remove unnecessary symbols from the binary by typing strip hello



```
[admin@FC12-64 ~]$
[admin@FC12-64 ~]$ strip hello
[admin@FC12-64 ~]$
```

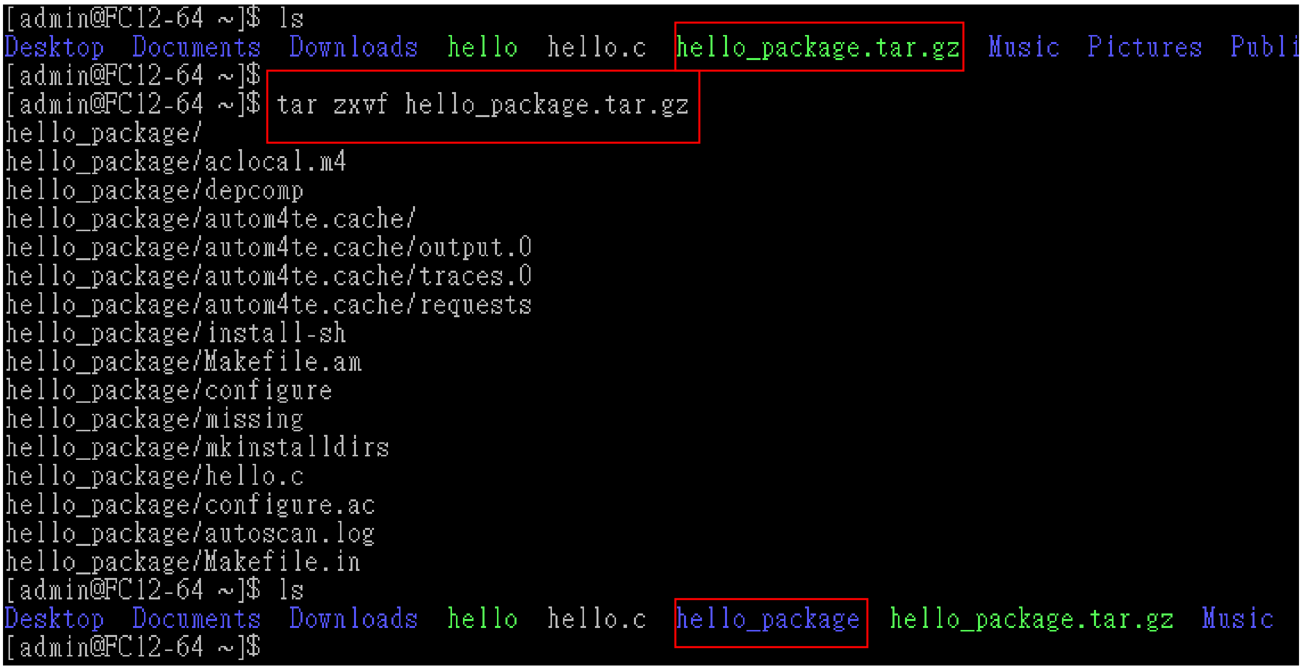
2. Compile an open source application

This section will talk about compiling an open source application. The example is making a package of “hello world”.

Procedure:

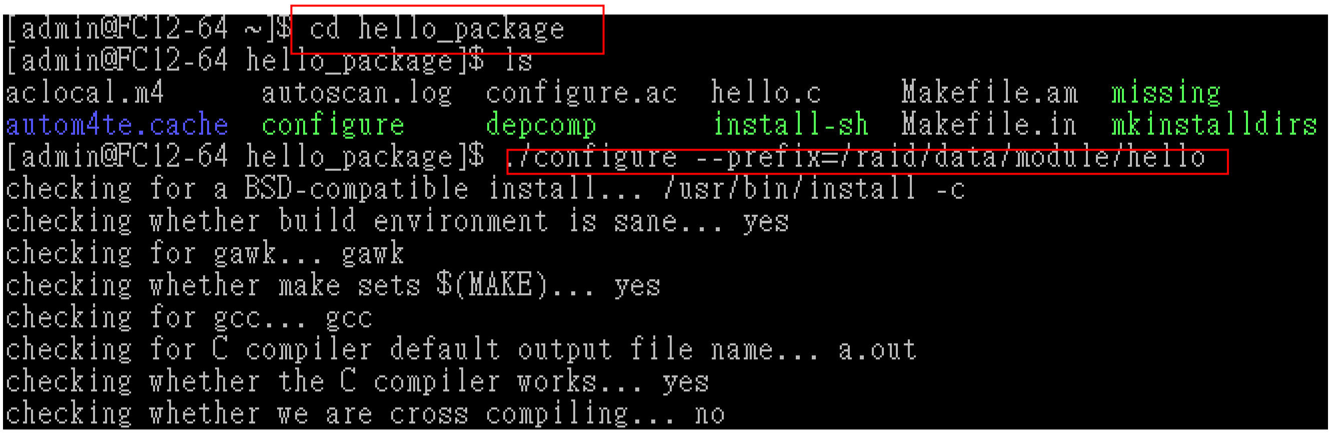
2.1 Upload hello_package.tar.gz (available at http://ftp.thecus.com/module/category-1/module/hello_example/) to /admin by Winscp (in Windows) or scp (in Linux). If there is permission concern, change to root permission by su.

2.2 Type tar zxvf hello_package.tar.gz in /admin to get a folder hello_package.



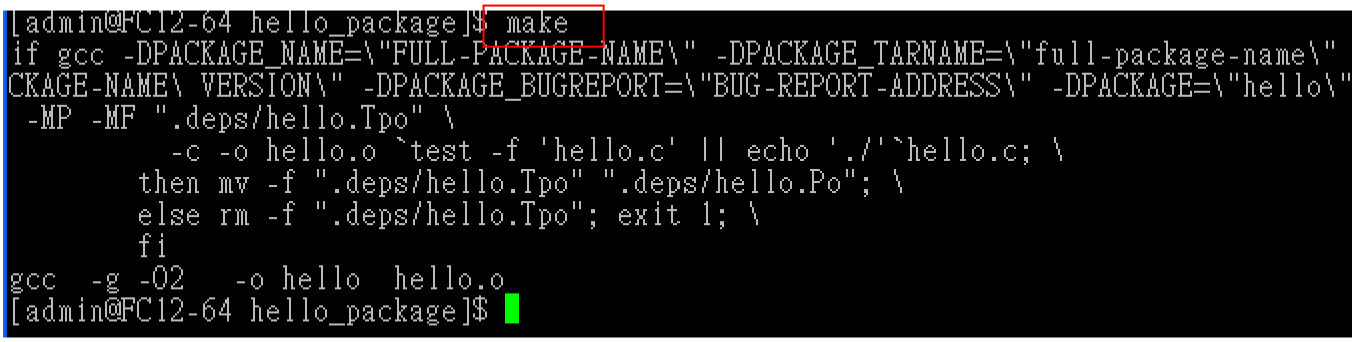
```
[admin@FC12-64 ~]$ ls
Desktop  Documents  Downloads  hello  hello.c  hello_package.tar.gz  Music  Pictures  Publi
[admin@FC12-64 ~]$
[admin@FC12-64 ~]$ tar zxvf hello_package.tar.gz
hello_package/
hello_package/aclocal.m4
hello_package/depcomp
hello_package/autom4te.cache/
hello_package/autom4te.cache/output.0
hello_package/autom4te.cache/traces.0
hello_package/autom4te.cache/requests
hello_package/install-sh
hello_package/Makefile.am
hello_package/configure
hello_package/missing
hello_package/mkinstalldirs
hello_package/hello.c
hello_package/configure.ac
hello_package/autoscan.log
hello_package/Makefile.in
[admin@FC12-64 ~]$ ls
Desktop  Documents  Downloads  hello  hello.c  hello_package  hello_package.tar.gz  Music
[admin@FC12-64 ~]$
```

2.3 Type ./configure --prefix=/raid/data/module/hello in hello_package folder (prefix stands for the path where the application will be installed to. For complicated application, set prefix to /raid/data/module/module_folder_name)



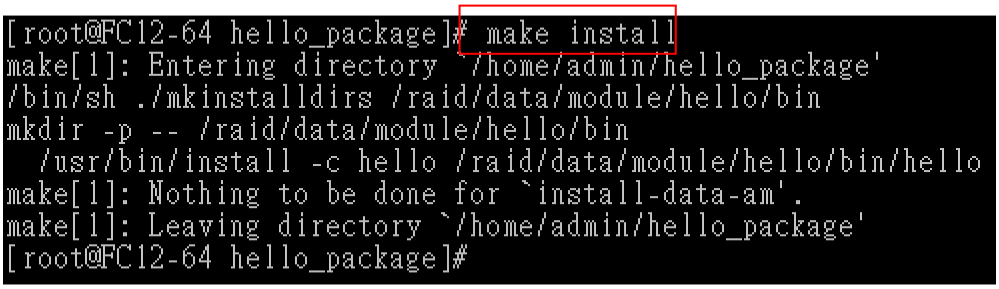
```
[admin@FC12-64 ~]$ cd hello_package
[admin@FC12-64 hello_package]$ ls
aclocal.m4      autoscan.log  configure.ac  hello.c     Makefile.am  missing
autom4te.cache  configure     depcomp       install-sh  Makefile.in  mkinstalldirs
[admin@FC12-64 hello_package]$ ./configure --prefix=/raid/data/module/hello
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
```

2.4 make



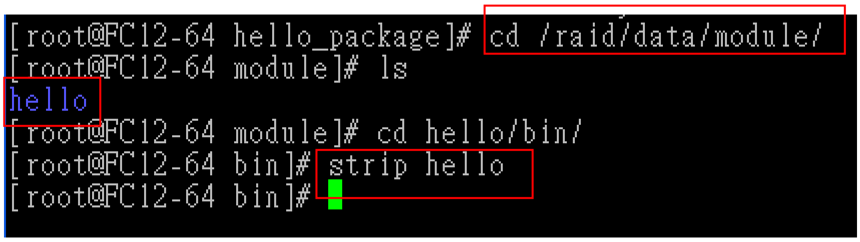
```
[admin@FC12-64 hello_package]$ make
if gcc -DPACKAGE_NAME=\"FULL-PACKAGE-NAME\" -DPACKAGE_TARNAME=\"full-package-name\"
CKAGE-NAME\ VERSION\" -DPACKAGE_BUGREPORT=\"BUG-REPORT-ADDRESS\" -DPACKAGE=\"hello\"
 -MP -MF ".deps/hello.Tpo" \
         -c -o hello.o `test -f 'hello.c' || echo './'`hello.c; \
        then mv -f ".deps/hello.Tpo" ".deps/hello.Po"; \
        else rm -f ".deps/hello.Tpo"; exit 1; \
        fi
gcc  -g -O2   -o hello  hello.o
[admin@FC12-64 hello_package]$
```

2.5 make install



```
[root@FC12-64 hello_package]# make install
make[1]: Entering directory `/home/admin/hello_package'
/bin/sh ./mkinstalldirs /raid/data/module/hello/bin
mkdir -p -- /raid/data/module/hello/bin
  /usr/bin/install -c hello /raid/data/module/hello/bin/hello
make[1]: Nothing to be done for `install-data-am'.
make[1]: Leaving directory `/home/admin/hello_package'
[root@FC12-64 hello_package]#
```

2.6 get into /raid/data/module/hello/bin, and then do strip hello



```
[root@FC12-64 hello_package]# cd /raid/data/module/
[root@FC12-64 module]# ls
hello
[root@FC12-64 module]# cd hello/bin/
[root@FC12-64 bin]# strip hello
[root@FC12-64 bin]#
```

PS: 1. When application is just for library application, direct the prefix to /raid/data/module/[module folder name]/sys

For example

./configure --prefix=/raid/data/module/Basic/sys

2. When application is an executable binary, direct the prefix to /raid/data/module/[module folder name]/bin

For example

/configure --prefix=/raid/data/module/Basic/bin

3. When application needs the other library, you have to install them before pack the application; like above step 1. Also, direct the library path to /raid/data/module/[module folder name]/sys/lib

Please note the application libraries are in lib folder in most cases; but not always.

For example

LDFLAGS=”-L/raid/data/module/Basic/sys/lib -lhello” LIBS=”-L/raid/data/module/Basic/sys/lib -lhello” ./configure

--prefix=/raid/data/module/Basic/bin

4. A sample

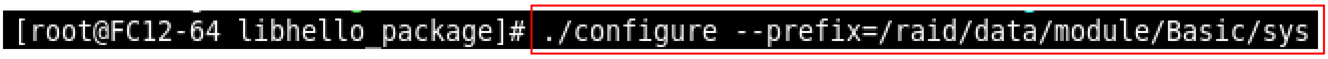
Module 'Basic' needs an executable binary hello_test, and hello_test needs libhello.a

Step 1) install libhello first:

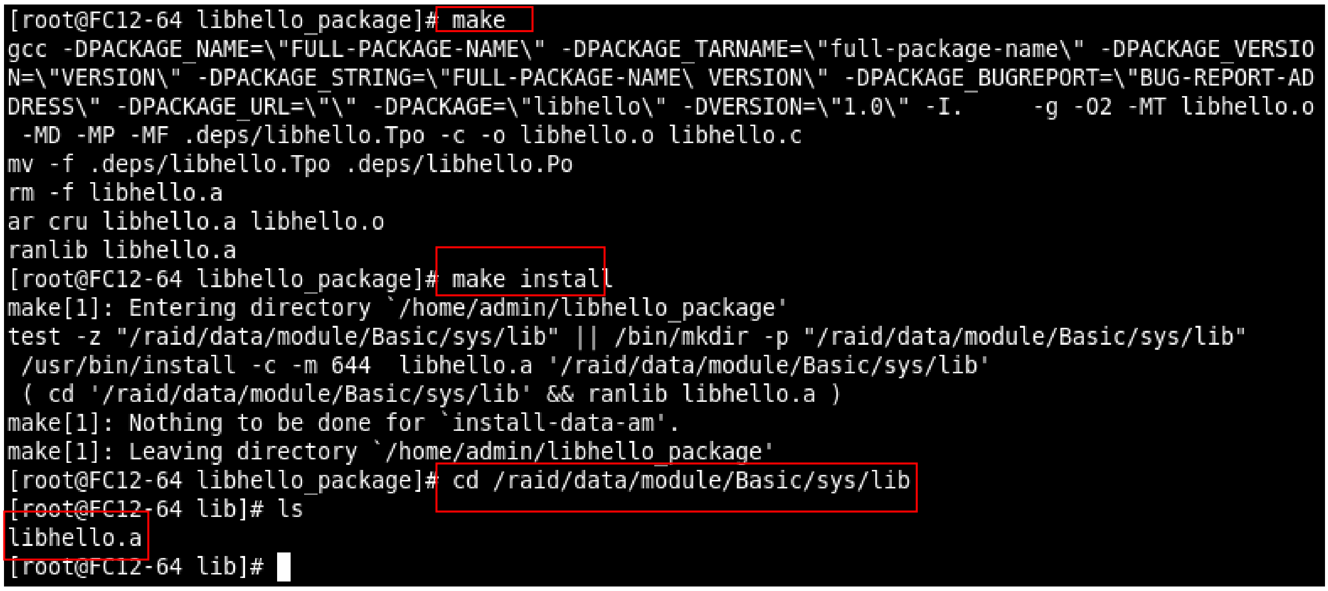
a ./configure --prefix=/raid/data/module/Basic/sys

b. make

c. make install



```
[root@FC12-64 libhello_package]# ./configure --prefix=/raid/data/module/Basic/sys
```



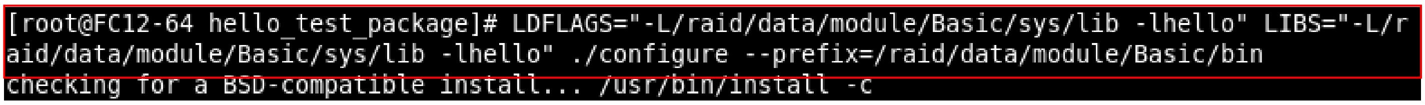
```
[root@FC12-64 libhello_package]# make
gcc -DPACKAGE_NAME=\"FULL-PACKAGE-NAME\" -DPACKAGE_TARNAME=\"full-package-name\" -DPACKAGE_VERSIO
N=\"VERSION\" -DPACKAGE_STRING=\"FULL-PACKAGE-NAME\ VERSION\" -DPACKAGE_BUGREPORT=\"BUG-REPORT-AD
DRESS\" -DPACKAGE_URL=\"\" -DPACKAGE=\"libhello\" -DVERSION=\"1.0\" -I.     -g -O2 -MT libhello.o
 -MD -MP -MF .deps/libhello.Tpo -c -o libhello.o libhello.c
mv -f .deps/libhello.Tpo .deps/libhello.Po
rm -f libhello.a
ar cru libhello.a libhello.o
ranlib libhello.a
[root@FC12-64 libhello_package]# make install
make[1]: Entering directory `/home/admin/libhello_package'
test -z "/raid/data/module/Basic/sys/lib" || /bin/mkdir -p "/raid/data/module/Basic/sys/lib"
 /usr/bin/install -c -m 644  libhello.a '/raid/data/module/Basic/sys/lib'
 ( cd '/raid/data/module/Basic/sys/lib' && ranlib libhello.a )
make[1]: Nothing to be done for `install-data-am'.
make[1]: Leaving directory `/home/admin/libhello_package'
[root@FC12-64 libhello_package]# cd /raid/data/module/Basic/sys/lib
[root@FC12-64 lib]# ls
libhello.a
[root@FC12-64 lib]# 
```

Step 2) install hello:

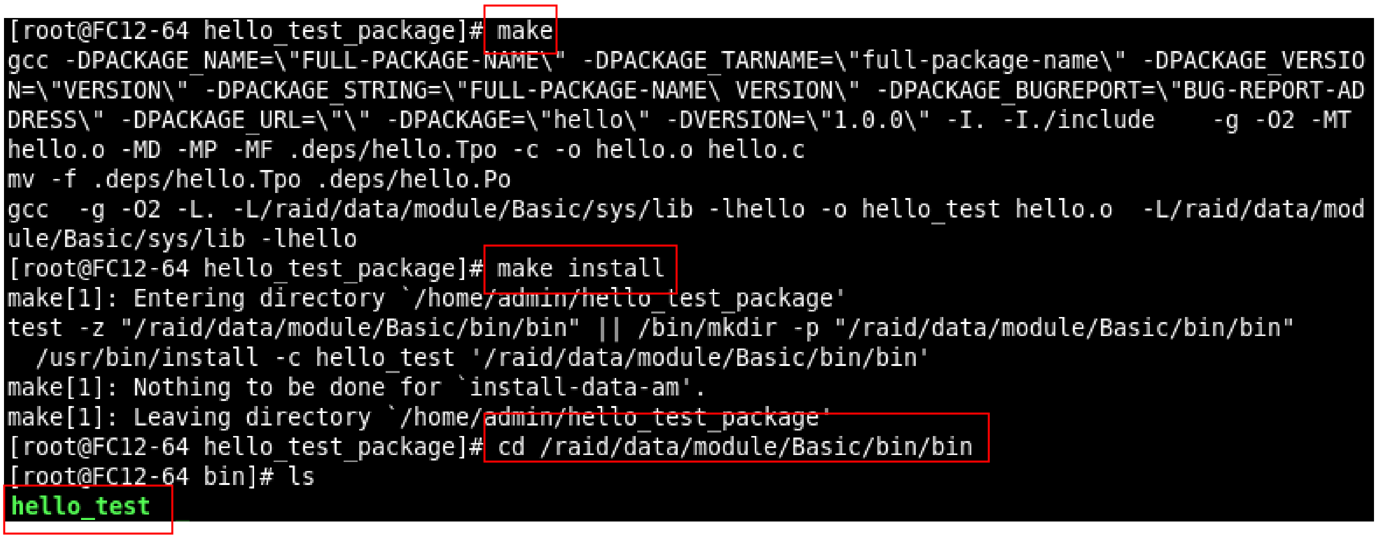
a. LDFLAGS="-L/raid/data/module/Basic/sys/lib -lhello" LIBS="-L/raid/data/module/Basic/sys/lib -lhello" ./configure --prefix=/raid/data/module/Basic/bin

b. make

c. make install



```
[root@FC12-64 hello_test_package]# LDFLAGS="-L/raid/data/module/Basic/sys/lib -lhello" LIBS="-L/r
aid/data/module/Basic/sys/lib -lhello" ./configure --prefix=/raid/data/module/Basic/bin
checking for a BSD-compatible install... /usr/bin/install -c
```



```
[root@FC12-64 hello_test_package]# make
gcc -DPACKAGE_NAME=\"FULL-PACKAGE-NAME\" -DPACKAGE_TARNAME=\"full-package-name\" -DPACKAGE_VERSIO
N=\"VERSION\" -DPACKAGE_STRING=\"FULL-PACKAGE-NAME\ VERSION\" -DPACKAGE_BUGREPORT=\"BUG-REPORT-AD
DRESS\" -DPACKAGE_URL=\"\" -DPACKAGE=\"hello\" -DVERSION=\"1.0.0\" -I. -I./include    -g -O2 -MT
hello.o -MD -MP -MF .deps/hello.Tpo -c -o hello.o hello.c
mv -f .deps/hello.Tpo .deps/hello.Po
gcc  -g -O2 -L. -L/raid/data/module/Basic/sys/lib -lhello -o hello_test hello.o  -L/raid/data/mod
ule/Basic/sys/lib -lhello
[root@FC12-64 hello_test_package]# make install
make[1]: Entering directory `/home/admin/hello_test_package'
test -z "/raid/data/module/Basic/bin/bin" || /bin/mkdir -p "/raid/data/module/Basic/bin/bin"
  /usr/bin/install -c hello_test '/raid/data/module/Basic/bin/bin'
make[1]: Nothing to be done for `install-data-am'.
make[1]: Leaving directory `/home/admin/hello_test_package'
[root@FC12-64 hello_test_package]# cd /raid/data/module/Basic/bin/bin
[root@FC12-64 bin]# ls
hello_test
```

Step 3) Make sure the executable binary is working, and then put it together with the library files to correct path; which is defined in the session ‘Module Path’ of ’Module Developers Guide’.

For example

Put all the files and sub folders in /raid/data/module/Basic/sys/lib to the System/lib under module folder; such as Basic/System/lib

Put all the files and sub folders in /raid/data/module/Basic/bin to Binary under module folder; such as Basic/Binary