FreeBSD on the Raspberry Pi - Pt 2: Crosscompiling ARMV6 packages for FreeBSD

Hi again. My last post covered the installation of FreeBSD on the Raspberry Pi.

Here's part 2 of my series on that topic.

As I pointed out last time there's currently only few experimental packages around, and yet, those few repositories out there are far from being complete.

So it's about time to see how to build packages for the Raspberry Pi on FreeBSD.
As the Pi isn't very fast, it's a good thing to look into cross-compiling packages.

I'm using a VM which runs FreeBSD 10.2 amd64 to achieve this. The VM has 4 cores and 8 GiB RAM, which should be more than enough for compile runs at decent speed.

As we're using a dedicated build machine, our whole build dependencies will be compiled from scratch as well. This is required anyway because we need poudriere-devel, which has cross-compile support. poudriere-devel is available as pre-compiled package for x86 platforms, but does not include qemu build support.
So that's why we gonna build it ourselves, along with other tools.

Enable and Prepare ZFS

First you'll need to enable ZFS as poudriere demands it.
Add this line to **/etc/rc.conf**:

```
zfs_enable="YES"
```

Then restart ZFS daemon:

```
service zfs start
```

Afterwards a ZFS pool should be initialized. I decided to call mine **build** and mount it beneath /build because I don't like cluttering /usr/local with my poudriere stuff.
Nedless to say that you need to have plenty of free space. I gave it some 250 GiB as a start.

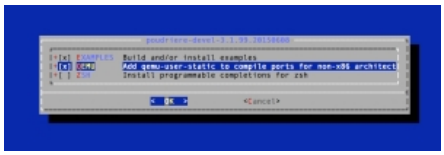
```
zpool create -m /build build /dev/da1
```

Software Dependencies

```
portsnap fetch extract
```

Then install poudriere-devel like this. Make sure to check the **qemu** option once the config dialog is shown.
Additionally you should install subversion. I'm not in the need for it as I use an existing snapshot, but it comes to good use later on if we should need to update the sources.

cd /usr/ports/ports-mgmt/poudriere-devel
make config install clean
cd /usr/ports/devel/subversion
make install clean



Configure poudriere

To use poudriere, edit the configuration file at **/usr/local/etc/poudriere.conf** and set these values:

ZPOOL=build
FREEBSD_HOST=ftp://ftp.freebsd.org # you should use the nearest local mirror here!
BASEFS=/build/poudriere/

Now let poudriere fetch the ports tree and give it a proper name.
I used the name of **p09_2015**.
You're free to call it whatever you like. Since I usually have multiple ports tree for several releases around, I make them bear a date-specific name.

poudriere ports -c -p p09_2015

Before you can actually start building the ports, we must register a handler for ARMv6 binaries. This is done using the binmiscctl utility. Be sure to check out the manpage for the proper syntax, which in our case should be:

binmiscctl add armv6
--interpreter "/usr/local/bin/qemu-arm-static"
--magic "x7fx45x4cx46x01x01x01x00x00x00x00x00x00x00x00x02x00x28x00"
--mask "xffxffxffxffxffxffx00xffxffxffxffxffxffxffxfexffxffxff"
--size 20 --set-enabled

Same once again as an image, just in case Wordpress would rip my code apart ;-)

To keep the change persistent, copy the above line to the file **/etc/rc.local**. Create it if necessary.
If you don't register the handler properly, poudriere will produce an error.

[00:00:00] ====>> Cross-building ports for armv6 on i386 requires QEMU
[00:00:00] ====>> Error: You need to setup an emulator with binmiscctl(8) for armv6

Building a Jail from a Snapshot

Right, you can use a snapshot, extract it and then feed it to poudriere in order to create the jail.

```
cd /tmp
fetch ftp://ftp.freebsd.org/pub/FreeBSD/releases/ISO-IMAGES/10.2/FreeBSD-10.2-RELEASE-arm-armv6-RPI-B.img.xz
unxz FreeBSD-10.2-RELEASE-arm-armv6-RPI-B.img.xz
```

Now attach it to a md loopback device in order to mount it.

```
mdconfig -a -t vnode -f FreeBSD-10.2-RELEASE-arm-armv6-RPI-B.img
```

mdconfig will return a device name, ie. **md0**.
Then mount the root filesystem into /mnt.
It is utterly important to copy over the qemu-arm-static binary from your host's /usr/local/bin directory into /mnt/usr/local/bin. Yes, it's true, your x86 static binary belongs in there, otherwise the jail will terribly fail at startup. The emulator must exist within the jail in order for the emulation to work!
Afterwards the contents of /mnt is to be tar'ed up.

```
mount /dev/md0s2a /mnt/
rm /mnt/firstboot
mkdir -p /mnt/usr/local/bin/
cp /usr/local/bin/qemu-arm-static /mnt/usr/local/bin/
cd /mnt
tar -cpf /build/fbsd10_2_release_armv6.tar .
umount /mnt
mdconfig -d -u 0
```

Now let poudriere create the jail from the previously created tar file:

```
# poudriere jail -c -j fbsd10armv6 -m tar=/build/fbsd10_2_release_armv6.tar -a armv6 -v 10.2-RELEASE
[00:00:00] ====>> Cross-building ports for armv6 on amd64 requires QEMU
[00:00:00] ====>> Creating fbsd10armv6 fs... done
[00:00:01] ====>> Installing 10.2-RELEASE armv6 from /build/fbsd10_2_release_armv6.tar ... done
[00:00:32] ====>> Recording filesystem state for clean... done
[00:00:32] ====>> Jail fbsd10armv6 10.2-RELEASE armv6 is ready to be used
```

Finally, have poudriere build a port, shells/bash in this example, to see if it works.

poudriere bulk -j fbsd10armv6 -p p09_2015 shells/bash

If everything goes smooth, you should see something like this:

Check out **top**, which should expose may child processes for qemu indicating that it actually works.

```
last pid: 36214;  load averages:  2.02,  0.92,  0.47                                  up 0+06:55:35  02:50:03
66 processes:  3 running, 63 sleeping
CPU: 20.5% user,  0.0% nice, 78.5% system,  0.7% interrupt,  0.3% idle
Mem: 119M Active, 1067M Inact, 318M Wired, 1496K Cache, 9792K Buf, 1494M Free
ARC: 147M Total, 42M MFU, 72M MRU, 665K Anon, 1204K Header, 31M Other
Swap: 1024M Total, 1024M Free

  PID USERNAME    THR PRI NICE   SIZE    RES STATE   C   TIME    WCPU COMMAND
36029 root          2  52    0   117M 35416K uwait   2   0:05  43.46% qemu-arm-static
36108 root          2  52    0   113M 31772K uwait   0   0:04  43.46% qemu-arm-static
36156 root          2  52    0   113M 25196K uwait   0   0:03  37.45% qemu-arm-static
36170 root          2  52    0   113M 27364K uwait   2   0:03  36.52% qemu-arm-static
36169 root          2  52    0   113M 10464K uwait   2   0:00  10.25% qemu-arm-static
36155 root          2  52    0   113M 10464K uwait   0   0:00   9.13% qemu-arm-static
36077 root          2  52    0   113M 10464K uwait   0   0:00   8.94% qemu-arm-static
36023 root          2  52    0   113M 10464K uwait   0   0:00   8.15% qemu-arm-static
36073 root          2  43    0   125M  6672K uwait   2   0:01   5.37% qemu-arm-static
36026 root          2  52    0   125M  6676K uwait   2   0:01   4.83% qemu-arm-static
36042 root          2  48    0   125M  6672K uwait   3   0:01   4.74% qemu-arm-static
35941 root          2  47    0   125M  6676K uwait   3   0:01   4.44% qemu-arm-static
```

So this is it on how to build packages for armv6 via cross-compiling.

Of course this is not yet the end of the story, since we're building a single package here.

There's two more ways on how to invoke poudriere.
You can have poudriere build from a list of ports listed in a file.
The file has the basic format of **category/portname**, i.e.

```
editors/vim-lite
shells/bash
devel/subversion
www/apache22
lang/php5
```

Run poudriere like this to process all ports in the file:

poudriere bulk -j fbsd10armv6 -p p09_2015 -f /path/to/file_with_port_names

Alternatively you could build the whole ports tree if you pass the **-a** argument.

poudriere bulk -j fbsd10armv6 -p p09_2015 -a

My next blog will cover how to expose the package repository created by poudriered via http, so stay tuned.

Know Issues

Here's some known issues I came along during the setup.
Error: Unable to execute id(1) in jail. Emulation or ABI wrong.
If poudriere bails out with this message, then emulation is indeed not working.
Reasons include:

- binmiscctl mapping for ARM is not properly configured. Be careful and follow the steps outlined above. Take extra care to use the proper arguments for the binary header. If in doubt, copy it directly from binmiscctl(8) manpage.
- qemu-arm-static is not included with the root file system. Please follow the setups outlined above and include it to the root file system before you tar it up for use with poudriere.

If you doubt that the emulation actually works, you can easily test it without poudriere.

Make sure that your ARM snapshot is mounted at /mnt as described above.

Copy the qemu-arm-static binary over to /mnt/usr/local/bin/qemu-arm-static as described.

Then run this command:

chroot /mnt /bin/sh

If you don't get an error it does work.
You can double check by issueing this command:

uname -a

If everything really works, you should get the uname output indicating that it runs on ARM. Doing the same on your host would indicate running on x86.

Further Reading

Now that you have mastered a successful package build, you'll notice that it's not really faster than compiling natively on the Raspberry Pi.

Sure, a centralized build and package repository is still better, because you don't go through the tedious package build for each

device.

If you want to get faster results, you'll need to install a cross-compile toolchain into your poudriere jail.

Be sure to read how to use host cross-compiler with poudriere if you're interested in this.