Bulk Build on QNX (WIP)#

There's various pages out there describing bulk builds on various platforms, all slightly different. Here's one more.

Base install separation#

If you look around you'll see that most people try to do their build in a sandbox of some sort. There's various ways to to this on each platform. I actually did a bulk build in a chroot on QNX once but it wasn't stock :). Here's the layout I'm using now:

- One drive in two partitions.
  1. Approx 8Gig for the qnx install which gets mounted read only (type 179).
  2. The rest of the drive where all the work gets done (type 178).
- Here's a sample build file that does this BulkBuild/qnxbasepbulk.build. Notice this mounts the small install partition (hd0t179) to / read only and the second work partition (hd0t178) as /fs/hd0-qnx6-2 as would normally be done by 'diskboot' in the default image.

When performing setup and administration on the box I boot from the normal qnxbasesmp image. When doing an actual bulk build I boot from the qnxbasebulk image.

To make the system usable in either configuration, I copy /tmp, /home and /var to the work partition which is read / write in either case then add procmgr symlinks to point there.

```
[shell]# cp -R /tmp /fs/hd0-qnx6-2/tmp
[shell]# cp -R /var /fs/hd0-qnx6-2/var
[shell]# cp -R /home /fs/hd0-qnx6-2/home
```

Two other directories are used during bulk builds as I have them configured: /pbulk and /usr/pkg. /pbulk is where I check out the pkgsrc repository and where I install packages required by the controlling part of the build (the outer layer). /usr/pkg is where resulting packages created by this outer layer are staged before being tarred up in a binary package. In pkgsrc parlance the outer layer controls the inner layer which builds packages with the same default prefix of /usr/pkg that would normally be used if one were to build packages themselves outside of a bulk build.

Here's a sample /etc/rc.d/rc.local that sets up these links BulkBuild/rc.local.

Basic box administration#

pbulk can be configured to run in 'parallel' mode (the 'p' in 'pbulk') with one master and multiple clients. My current setup is one box as master which is also the only client with the hope that other clients may be added in the future. As can be seen in the sample rc.local, I have one 10.x.x.x network on the master for control and a 192.168.x.x network for all the clients. In this mode the master needs to be able to ssh into each client. I also run the bulk build as root (not sure if this is still required) so you need to set up /etc/ssh/sshd_config on each client with 'PermitRootLogin yes'. To avoid entering passwords all the time, start a ssh-agent session on the master prior to starting the bulk build:

```
[shell]# eval $(ssh-agent)
[shell]# ssh-add                    <- password once
[shell]# ssh 192.168.1.1 echo hello   <- ssh into client0 (self).  Shouldn't be prompted for a passwd.
```

Verify you can ssh into each client as root with no passwd.

I move /root to /home/root. Recall /root is read only during a bulk build. Alternatively you can add an extra symlink:

```
[shell]# cp -R /root /home/root
< update /etc/passwd entry for root >
[shell]# grep ^root /etc/passwd
root::0:0:Superuser:/home/root:/bin/sh
```

I change the umask to 0022 and add /pbulk/pkg_bulk dirs to PATH. Something in this process actually expected this umask (can't recall exactly what) but it's also generally good practice. More on /pbulk/pkg_bulk below:

```
[shell]# cat ~/.profile
umask 0022
export PATH=$PATH:/pbulk/pkg_bulk/bin:/pbulk/pkg_bulk/sbin
```

The default /pbulk/pkg_bulk/etc/pbulk.conf specifies 'unprivileged_user=pbulk' so such a user needs to be created.

```
[shell]# grep pbulk /etc/passwd
pbulk:x:100:100:pbulk:/home/pbulk:/bin/sh
[shell]# grep pbulk /etc/group
pbulk:x:100:
```

pbulk setup#

Notice also that the above rc.local script adds /pbulk/pkg_bulk/lib to the CS_LIBPATH. As mentioned above, the outer, controlling part of the bulk build requires a few packages for its operation (here's where we get into some pbulk specifics). I install these into /pbulk/pkg_bulk so the first step in the actual pbulk process is to bootstrap in this location:

```
[shell]# cd /pbulk
< checkout pkgsrc repo to pkgsrc>
[shell]# cd pkgsrc/bootstrap
[shell]# ./bootstrap --prefix=/pbulk/pkg_bulk --pkgdbdir=/pbulk/pkg_bulk/.pkgdb
```

Now install the packages needed for a bulk build and anything else you find generally useful. For example:

```
[shell]# cd /pbulk/pkgsrc/pkgtools/pbulk
[shell]# /pbulk/pkg_bulk/bin/bmake install
```

The last time I did a bulk build I had:

```
[shell]# /pbulk/pkg_bulk/sbin/pkg_info
install-sh-20070712 install script compatible with the BSD install program
bmake-20081111      Portable (autoconf) version of NetBSD 'make' utility
nawk-20050424nb3    Brian Kernighan's pattern-directed scanning and processing language
nbsed-20040821nb1   NetBSD-current's sed(1)
pkg_install-20090518 Package management and administration tools for pkgsrc
digest-20080510     Message digest wrapper utility
libtool-base-1.5.26nb1 Generic shared library support script (the script itself)
pax-20080110        POSIX standard archiver with many extensions
libiconv-1.12nb1    Character set conversion library
rsync-3.0.5nb1      Network file distribution/synchronisation utility
bzip2-1.0.5nb1      Block-sorting file compressor
rcs-5.7nb3          GNU Revision Control System - version control software
heirloom-mailx-12.4 BSD mail utility with MIME extensions
pbulk-0.37          Modular bulk build framework
pkg_install-info-4.5nb3 Standalone GNU info file installation utility
screen-4.0.3nb2     Multi-screen window manager
m4-1.4.13           GNU version of UNIX m4 macro language processor
tcp_wrappers-7.6.1nb4 Monitor and filter incoming requests for network services
```

sendmail-8.14.3nb4 The well known Mail Transport Agent
mailwrapper-19990412nb4 Wrapper to support arbitrary Mail Transport Agents
vim-share-7.2.184 Data files for the vim editor (vi clone)
gettext-lib-0.14.6 Internationalized Message Handling Library (libintl)
vim-7.2.184 Vim editor (vi clone) without GUI
gettext-tools-0.14.6nb1 Tools for providing messages in different languages
pkg-config-0.23nb1 System for managing library compile/link flags
apr-1.3.3 Apache Portable Runtime
expat-2.0.1 XML parser library written in C
apr-util-1.3.4nb1 Apache Portable Runtime utilities
zlib-1.2.3 General purpose data compression library
neon-0.28.3 HTTP and WebDAV client library
subversion-base-1.5.6nb3 Version control system, base programs and libraries

I recall that a bulk build required 'rsync' and 'mailx' for its reporting phase. The latter requires 'sendmail' which I had to install and configure. If you're not interested in the reporting phase none of these may be required. 'screen' is pretty much a must have if you want to check on progress remotely since a bulk build can take a long time. I think the rest are personal preference.

Configure /pbulk/pkg_bulk/etc/pbulk.conf#

Most of this should be more or less self explanatory. Set your pkg_rsync_target, report_rsync_target and report_recipients as desired. I have the following sections which follows the setup as described above:

```
pclient0="192.168.1.1"
pclient1="192.168.1.2"

master_ip="${pclient0}"
scan_clients="${pclient0}"
build_clients="${pclient0}"
...
...
...
bootstrapkit=/pbulk/bootstrap_kit.tar.gz
...
...
...
target_arch=qnx6
target_destdir=/pbulk/destdir.${target_arch}

# The directories where the various files are created.
#
bulklog=/pbulk/bulklog
packages=/pbulk/packages
prefix=/usr/pkg
pkgsrc=/pbulk/pkgsrc
```

The creation of /pbulk/bootstrap_kit.tar.gz is described below:

create /pbulk/bootstrap_kit.tar.gz#

```
[shell]# cd /pbulk/pkgsrc/bootstrap && ./bootstrap --mk-fragment /tmp/mk-fragment.conf --gzip-binary-kit /pbulk/bootstrap_kit.
```

Where the attached is a sample BulkBuild/mk-fragment.conf.