Slackware Aarch64 and Virtualization

Virtualization on the Aarch64 architecture simplifies research, development, and testing. The goal of this document is to aid the user to create a favorable environment without any complicated requirements. The hope is that this will enable more Slackware ARM users to contribute to the project. This guide targets the RockPro64, the Pinebook Pro, and the Honeycomb LX2K hardware models. This same process may work on the Raspberry Pi 4, but remains untested.

Work in progress.

Software Requirements

- Full installation of Slackware Aarch64
- Qemu built with the aarch64 target
- libvirt as the back end
- virt-manager as the GUI for local and remote access
- netcat-openbsd to enable remote connections to libvirt

SlackBuilds and Packages

- The build scripts are here
- Pre-built packages here.

In the future the pre-built packages will become outdated and you will need to rebuild the packages with the included build scripts. The current package set was built on the Aarch64 port of Slackware-current. The software is constantly upgraded in Slackware-current. What works now, may not work tomorrow. It is recommended that the packages are rebuilt often as an exercise for the user.

You can use your favorite package manager for SlackBuilds.org to automate the build installation. That is beyond the scope of this document.

The whole bundle can be downloaded from the Slackware.uk public mirror like so:

```
$ mkdir -p /tmp/slackware-aarch64-virtualization
$ cd /tmp/slackware-aarch64-virtualization
$ rsync -Paav slackware.uk::slackwarearm/people/brent/slackware-aarch64-
virtualization/ .
```

Then enter the directory if you are not already in it:

```
cd /tmp/slackware-aarch64-virtualization/virtualization-scripts
```

Verify the SlackBuild scripts are all marked executable.

```
chmod -v +x arm/build  */arm/build */*SlackBuild  tools/refresh.source
```

Make sure you remove the vanilla Slackware package of netcat. It is named *nc* and can be removed easily with *removepkg*. This package is swapped out for *netcat-openbsd* to enable virt-manager and libvirt to communicate when virt-manager is connected remotely to a virtual machine host system.

Remove stock netcat:

```
removepkg nc
```

To verify the checksum of existing source archives without having to download them again:

```
CHECK=1 REFRESH=0 ./tools/refresh.source
```

Output should look similar to:

```
spice-protocol-0.14.4.tar.xz: OK
spice-0.15.1.tar.bz2: OK
spice-gtk-0.42.tar.xz: OK
gtk-vnc-1.3.1.tar.xz: OK
lloyd-yajl-2.1.0-0-ga0ecdde.tar.gz: OK
libvirt-9.1.0.tar.xz: OK
libvirt-python-9.1.0.tar.gz: OK
libvirt-glib-4.0.0.tar.xz: OK
osinfo-db-tools-1.10.0.tar.xz: OK
osinfo-db-20230308.tar.xz: OK
libosinfo-1.10.0.tar.xz: OK
virt-manager-4.1.0.tar.gz: OK
libmd-1.1.0.tar.xz: OK
libbsd-0.11.7.tar.xz: OK
netcat-openbsd-7.3_1.tar.gz: OK
qemu-7.2.1.tar.xz: OK
```

Update the SlackBuilds using newer source archives by editing arm/build and the download.info files in each sub directory. To download the source code for each build script use the utility in the tools/ directory:

```
CHECK=0 REFRESH=1 ./tools/refresh.source
```

Execute the build script in the root of the tree:

```
./arm/build
```

Then you wait. Depending on your system, it could be a significantly long time frame to wait. Each package will be built and installed automatically. The build script will exit if a build fails or cannot be installed/upgraded.

Post Installation

TODO: screen shots

Configure system to start the Libvirt daemon during system boot.

```
chmod -v +x /etc/rc.d/rc.libvirt  /etc/rc.d/rc.local
vim /etc/rc.d/rc.local
```

Add in rc.local

```
if [ -x /etc/rc.d/rc.libvirt ]; then
    /etc/rc.d/rc.libvirt start
fi
```

To start the daemon without restarting your machine:

```
/etc/rc.d/rc.libvirt start
```

Manage Graphically

TODO: screen shots

Your virtual machine host can be a x86, x86_64, or aarch64. Other architectures were not tested. It can be accessed remotely on your local network through the virt-manager graphical environment on your Aarch64 machine. It is all interchangeably the same scenario for each architecture. On 32 bit ARM, its possible to use just Virt-manager to access other machines running qemu virtual machines.