QEMU

From Crashcourse Wiki

Contents

Overview

This page is going to take a while to finish, and you're certainly welcome to visit [QEMU's home page (http://fabrice.bellard.free.fr/qemu/)] in the meantime.

NOTE: This page is way out of date, and will be updated at some point

Installation

. While you can install the stock QEMU 0.9.0 on Fedora 8, there are [noticeable improvements in the 0.9.1 version (http://fabrice.bellard.free.fr/qemu/changelog.html)], so I'd strongly recommend manually installing the [ready-made RPM (http://rpmfind.net/linux/RPM/fedora/devel/i386/qemu-0.9.1-1.fc9.i386.html)]. Even though that's a development version, it appears to install on Fedora 8 nicely.

You can also install the graphical qemu-launcher if you want:

```
# yum install qemu-launcher
```

which will now be available under `Applications -> System Tools`.

Finally, you can install the `kqemu` QEMU Accelerator packages, which are not available in the default Fedora repos, but can be found [here (http://atrpms.net/dist/f8/kqemu/)]. For Fedora 8, you'll want both of:

- `kqemu-1.3.0-2.fc8.i386.rpm`
- `kqemu-kmdl-2.6.23.9-85.fc8-1.3.0-2.fc8.i686.rpm`

or whichever variation is appropriate for your system. Once you do that, get `udev` to take notice with:

```
# modprobe kqemu
# udevcontrol reload_rules
```

after which you can verify the proper installation of the accelerator with:

```
$ ls -l /dev/kqemu
```

So what exactly is QEMU?

QEMU is a processor emulator which has two different operating modes:

- *user mode emulation*, which allows you to run a simple cross-compiled executable, and
- *full system emulation*, which emulates a full system including the corresponding hard disk image.

Once you've installed QEMU, you can see the variety of supported architectures by running:

```
$ ls -l /usr/bin/qemu*
```

Most architectures will have two associated commands, such as:

- `qemu-arm`: the user mode emulation program for the ARM architecture, and
- `qemu-system-arm`, the full system emulator for that same architecture.

You can immediately see the difference in complexity by running these commands with no arguments, which prints their usage:

```
$ qemu-arm
...
$ qemu-system-arm
...
```

In addition, for any architecture, you can list the supported systems available for full system emulation:

```
$ qemu-system-arm -M ?
Supported machines are:
integratorcp ARM Integrator/CP (ARM926EJ-S) (default)
versatilepb ARM Versatile/PB (ARM926EJ-S)
versatileab ARM Versatile/AB (ARM926EJ-S)
realview   ARM RealView Emulation Baseboard (ARM926EJ-S)
akita      Akita PDA (PXA270)
spitz      Spitz PDA (PXA270)
borzoi     Borzoi PDA (PXA270)
terrier    Terrier PDA (PXA270)
cheetah    Palm Tungsten|E aka. Cheetah PDA (OMAP310)
lm3s811evb Stellaris LM3S811EVB
lm3s6965evb Stellaris LM3S6965EVB
connex     Gumstix Connex (PXA255)
verdex     Gumstix Verdex (PXA270)
mainstone  Mainstone II (PXA27x)
```

Testing the installation

User mode testing

If you want to test at least the user mode emulation of your installed QEMU, you can do that by downloading the user test tarball found [here (http://fabrice.bellard.free.fr/qemu/linux-user-test-0.3.tar.gz)] and expanding it into a personal directory of your choice. However, you still have a bit of work to do before the testing.

Once unloaded, that tarball contains the script `qemu-linux-user.sh`, which appears to work only if you've built QEMU from source on your system. In short, if you've installed from RPM, it's not going to do you any good. To get around this, see the updated script below, which can be invoked to test the user mode emulation in a couple different ways:

```
$ ./qemu-linux-user.sh arm      [Little-endian ARM.]
$ ./qemu-linux-user.sh armeb    [Big-endian ARM.]
$ ./qemu-linux-user.sh ppc      [PowerPC, of course.]
$ ./qemu-linux-user.sh          [Test all possible architectures.]
```

See the updated script below.

System level testing

To test QEMU at the system level, go to the [QEMU download page (http://fabrice.bellard.free.fr/qemu/download.html)] to the section "QEMU disk images" and grab one of the architecture-specific examples, download and unpack it, and follow the instructions in the `README` file.

As one example, the ARM test appears to work properly under QEMU version 0.9.1. Go into the `arm-test` directory and try:

```
$ qemu-system-arm -kernel zImage.integrator -initrd arm_root.img
```

The QEMU PC system emulator

If you have a simple PC image, you can emulate it with:

```
$ qemu [options] [disk image]
```

where "disk image" is a file representing a raw hard disk image for IDE hard disk 0 and the emulator simulates the following peripherals (among others):

- i440FX host PCI bridge and PIIX3 PCI to ISA bridge
- Cirrus CLGD 5446 PCI VGA card or dummy VGA card with Bochs VESA extensions
- PS/2 mouse and keyboard
- 2 PCI IDE interfaces with hard disk and CD-ROM support

- Floppy disk

and much more. To see the full list and an explanation of the possible options:

```
$ man qemu
```

As a working example, go to the [QEMU download page (http://fabrice.bellard.free.fr/qemu/download.html)] and download the disk image `linux-0.2.img.bz2` (which represents a "small Linux disk image containing a 2.6.20 Linux kernel, X11 and various utilities"), uncompress it, and run it with:

```
$ qemu linux-0.2.img
```

Read all about the PC system emulator [here (http://fabrice.bellard.free.fr/qemu/qemu-doc.html#SEC7)].

The updated user mode test script

This script tests the user mode emulation by trying to run the architecture-specific `ls -l` command on an empty file.

```
#!/bin/sh
# QEMU Linux user mode tests
set -e

# set qemu_dir to the directory when qemu was compiled
qemu_dir="../qemu-release"

if test "$1" = "" ; then

# Working architectures (tested on x86 host)
archs="i386 arm armeb sparc sparc32plus ppc ppc64abi32 mips mipsel sh4 sh4eb"

# Almost working
archs="$archs x86_64"

# Not working architectures. Some of the problems may come from address
# space conflicts between the host and guess.
#archs="$archs alpha ppc64 m68k sparc64"

else
archs="$1"
fi

for arch in $archs ; do
  arch1=$arch
  prog=""
  case "$arch" in
    sparc32plus)
    arch1="sparc"
  ;;
    ppc64abi32)
    arch1="ppc"
  ;;
    x86_64)
# vsyscalls not supported, so we cannot use gettimeofday, hence no -l
    prog="$arch1/ls dummyfile"
  ;;
  esac
```

```
  if type qemu-$arch > /dev/null ; then
        qemu=qemu-$arch
        echo "Found qemu-$arch"
  elif type $qemu_dir/$arch-linux-user/qemu-$arch > /dev/null ; then
        qemu="$qemu_dir/$arch-linux-user/qemu-$arch"
        echo "Going with $qemu"
  else
        echo "Can't find $qemu-arch, skipping."
        continue
  fi

  gnemul_dir="$(pwd)/gnemul/qemu-$arch1"
  if test "$prog" = "" ; then
    prog="$arch1/ls -l dummyfile"
  fi

  echo "[qemu-$arch]"
  echo $qemu -L $gnemul_dir $prog
  $qemu -L $gnemul_dir $prog
done
```

Return to Fedora_Cookbook.

Retrieved from "http://www.crashcourse.ca/wiki/index.php/QEMU"