Building a bootable guest image for Palacios and Kitten

January 19, 2010

1 Getting the guest image build tools

In order to build the bootable guest ISO image, we need to build a Linux kernel from source and an initial ramdisk file system containing a set of useful tools. We will use a new directory for demonstration; the root directory for the following examples is "`test/`":

```
[jdoe@newskysaw ~]$ mkdir test/
```

There are a set of tools and sources that are useful for the guest image building procedure. You can obtain these resources from our git repositories. Change to the "`test/`" directory and clone the resources:

```
[jdoe@newskysaw test]$ git clone http://hornet.cs.northwestern.edu:9005/busybox
[jdoe@newskysaw test]$ git clone http://hornet.cs.northwestern.edu:9005/initrd
[jdoe@newskysaw test]$ git clone http://hornet.cs.northwestern.edu:9005/linux-2.6.30.y
```

2 Building the ramdisk filesystem

The guest requires an initial ramdisk filesystem. Jack has made one that you can leverage; it is temporarily located in his home directory. You will need sudo or root access to create the device files when you unpack the archive:

```
[jdoe@newskysaw test]$ cp /home/jarusl/initrd/disks/v3vee_initramfs.tar.gz .
[jdoe@newskysaw test]$ sudo tar -C initrd -xzf v3vee_initramfs.tar.gz
```

If you require a custom initial ramdisk filesystem, change to the "`initrd/initramfs/`" directory and perform the following steps:

```
[jdoe@newskysaw initramfs]$ mkdir -p proc sys var/log
```

Edit the "`init_task`" script and uncomment these lines:

```
#mknod /dev/tty0 c 4 0
#mknod /dev/tty1 c 4 1
#mknod /dev/tty2 c 4 2
```

Create the "`console`" device. If you have sudo or root access it is possible to create this device manually:

```
[jdoe@newskysaw initramfs]$ sudo mknod dev/console c 5 1
[jdoe@newskysaw initramfs]$ sudo chmod 0600 dev/console
```

If you do not have sudo or root access it is still possible to create the "`console`" device indirectly through the kernel build. Change to the "`initrd/`" directory and create a file called "`root_files`". Add the following line:

```
nod /dev/console 0600 0 0 c 5 1
```

The "`root_files`" file is used when building the Linux kernel in the section Configuring and building the Linux kernel. Finally, create any additional directories and copy any additional files that you need. Your initial ramdisk filesystem is prepped and ready for installation of the BusyBox tools as described in the section Configuring and installing BusyBox tools.

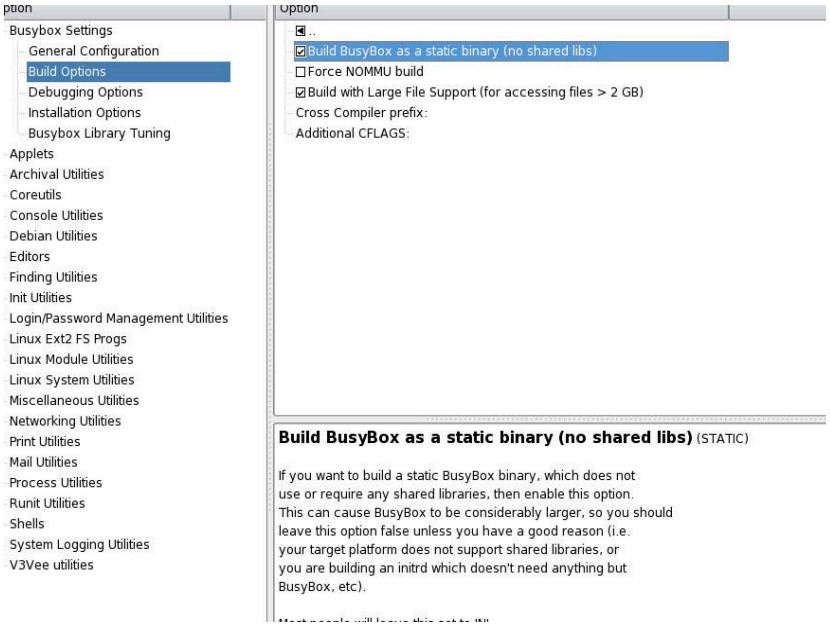

Figure 1: BusyBox configuration

3 Configuring and installing BusyBox tools

BusyBox is a software application released as Free software under the GNU GPL that provides many standard Unix tools. BusyBox combines tiny versions of many common UNIX utilities into a single, small executable. For more details on BusyBox visit `http://busybox.net`. To configure BusyBox, in the "`busybox/`" directory, type the following:

```
[jdoe@newskysaw busybox]$ make menuconfig
```

or

```
[jdoe@newskysaw busybox]$ make xconfig
```

The BusyBox tools will be installed in the guest's initial ramdisk filesystem; you can add any tools that you need. There are two required configuration options. In the "`BusyBox settings->Build Options`" menu check the "`Build BusyBox as a static binary (no shared libs)`" option, as shown in figure 1, and in the "`BusyBox settings->Installation Options`" menu set the "`Busybox installation prefix`" to the path of the "`initrd/initramfs`" directory, as shown in figure 2. After you finish configuring BusyBox, save your configuration and quit the window. Then, to make the BusyBox tools, type the following:

```
[jdoe@newskysaw busybox]$ make
```

Install the tools to the guest's initial ramdisk filesystem directory:

```
[jdoe@newskysaw busybox]$ make install
```

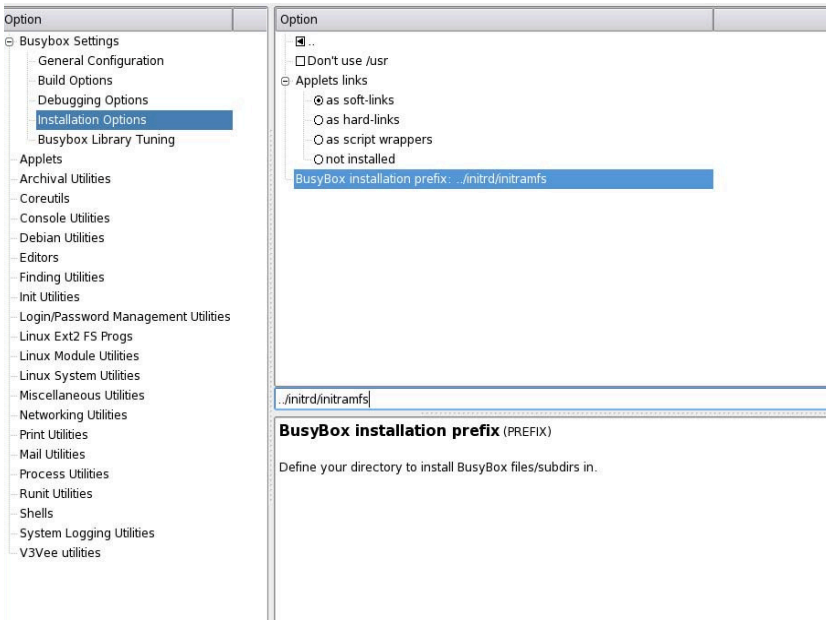

Figure 2: BusyBox configuration

4 Configuring and building the Linux kernel

The following procedure demonstrates how to configure and build a 32-bit Linux kernel. Change to the "`linux-2.6.30.y/`" directory. There is a custom configuration file "`jrl-default-config`" which is configured with minimal kernel options (all unnecessary options are removed to keep the guest booting process fast). If you are using the custom configuration file type the following:

```
[jdoe@newskysaw linux-2.6.30.y]$ cp jrl-default-config .config
```

Configure the kernel to meet your requirements. For more on configuring and building Linux kernels, check online. Type the following:

```
[jdoe@newskysaw linux-2.6.30.y]$ make ARCH=i386 menuconfig
```

or

```
[jdoe@newskysaw linux-2.6.30.y]$ make ARCH=i386 xconfig
```

The kernel must be configured with the initial ramdisk file system directory (e.g. "`initrd/initramfs/`"): in the "`General setup`" menu under option "`Initial RAM filesystem and RAM disk support`" set the "`Initramfs source file(s)`" option to the path of the "`initrd/initramfs/`" directory, as shown in figure 3. Additionally, if you are using the "`root_files`" file to create devices files, add the "`root_files`" file path, separated by a space, after the initial ramdisk filesystem directory. When you are finished configuring the kernel, save your configuration, and build a bootable ISO image:

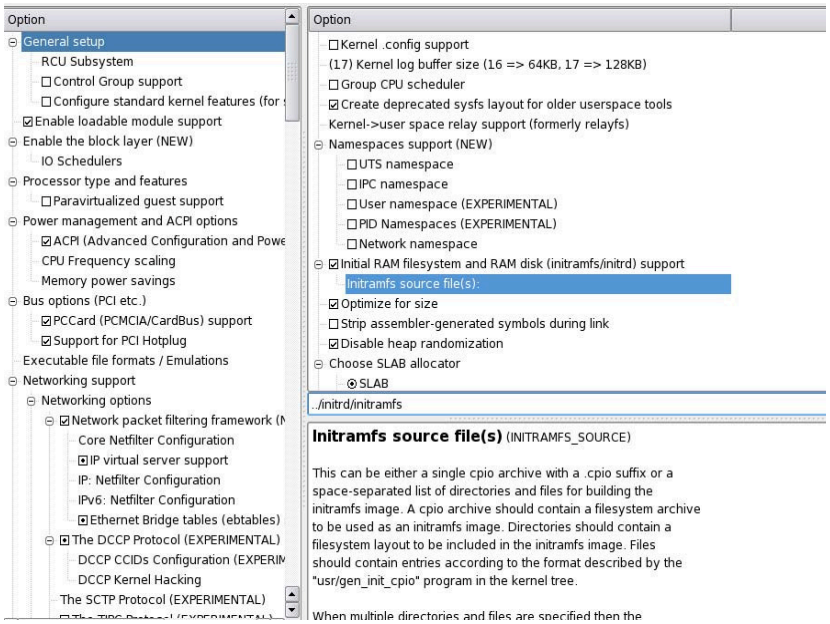

Figure 3: Linux Kernel configuration

```
[jdoe@newskysaw linux-2.6.30.y]$ make ARCH=i386 isoimage
```

The ISO image can be found here: "`arch/x86/boot/image.iso`", and will be used in the section Configuring and building the guest image.

5 Configuring and building the guest image

Checkout the updated Palacios repository to the "`palacios/`" directory. (You can find instructions for checking out the Palacios repository at `http://www.v3vee.org/palacios/`). The guest creator utility is required for building the guest image. Change to the "`palacios/utils/guest_creator`" directory and build the guest creator utility:

```
[jdoe@newskysaw guest_creator]$ make
```

You will get the "`build_vm`" utility:

```
[jdoe@newskysaw guest_creator]$ file build_vm
build_vm: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked
(uses shared libs), for GNU/Linux 2.6.9, not stripped
```

The guest configuration file is written in XML. A sample configuration file is provided: "`default.xml`". Make a copy of the default configuration file named "`myconfig.xml`" and edit the configuration elements that you are interested in (if a device is included in the guest configuration file, it must be configured in

the section Configuring and building Palacios or the guest will not boot). Of particular importance is the "`files`" element. Comment out this attribute:

```
<file id="boot-cd" filename="/home/jarusl/image.iso" />
```

Add an attribute that specifies the location of the Linux ISO image:

```
<file id="boot-cd" filename="../../../linux-2.6.30.y/arch/x86/boot/image.iso" />
```

When you are finished editing the guest configuration save the configuration file. The guest image consists of the guest configuration file and the Linux ISO image. Build the guest image with the guest creator utility:

```
[jdoe@newskysaw guest_creator]$ ./build_vm myconfig.xml -o guest.iso
```

The guest image, "`guest.iso`", is embedded in Kitten's "`init_task`" in the section Configuring and building Kitten.

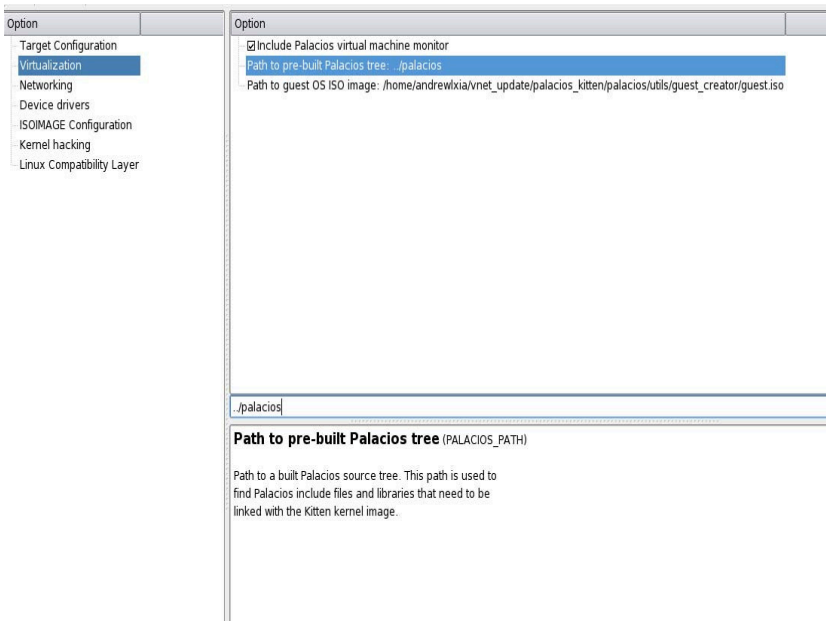

Figure 4: Kitten configuration

6 Configuring and building Palacios and Kitten

Configuring and building Palacios

You can find the detailed manual of getting and building Palacios and Kitten from scratch in the Palacios website (`http://www.v3vee.org/palacios`). Here we only give the specific requirements related to the procedure of booting the guest. To configure Palacios, change to the "`test/palacios/`" directory and type the following:

```
[jdoe@newskysaw palacios]$ make menuconfig
```

or

```
[jdoe@newskysaw palacios]$ make xconfig
```

Don't forget to include the devices that your guest image requires. When you have configured the components you want to build into Palacios, save the configuration and close the window. To build Palacios type the following:

```
[jdoe@newskysaw palacios]$ make
```

or

```
[jdoe@newskysaw palacios]$ make all
```

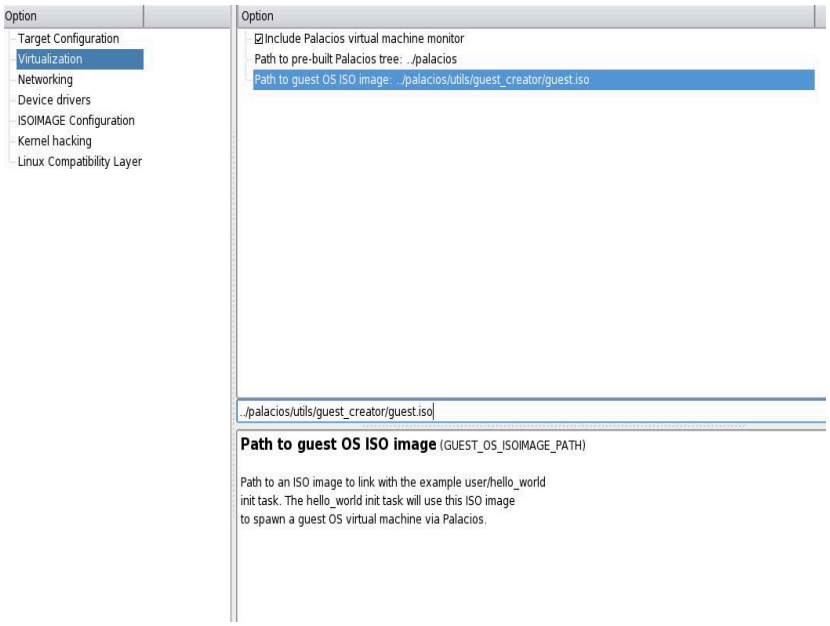

Figure 5: Kitten configuration

Once the Palacios static library has been built you can find the library file, "`libv3vee.a`", in the Palacios root directory.

Configuring and building Kitten

Configure Kitten. Change to the "`test/kitten/`" directory and type the following:

```
[jdoe@newskysaw kitten]$ make menuconfig
```

or

```
[jdoe@newskysaw kitten]$ make xconfig
```

Under the "`Virtualization`" menu select the "`Include Palacios virtual machine monitor`" option. Set the "`Path to pre-built Palacios tree`" option to the Palacios build tree path, "`..\palacios`", as shown in figure 4. Set the "`Path to guest OS ISO image`" option to the guest image path, "`../palacios/utils/guest_creator/guest.iso`", as shown in figure 5. When you have finished configuring Kitten, save the configuration and close the window. To build Kitten type the following:

```
[jdoe@newskysaw kitten]$ make isoimage
```

This builds the bootable ISO image file with guest image, Palacios, and Kitten. The ISO file is located in "`kitten/arch/x86_64/boot/image.iso`".

You have successfully created an ISO image file that can be booted on a machine. You can boot the file on Qemu using the following sample command:

```
[jdoe@newskysaw test]$ /opt/vmm-tools/qemu/bin/qemu-system-x86_64 \
        -smp 1 \
        -m 2047 \
        -serial file:./serial.out \
        -cdrom kitten/arch/x86_64/boot/image.iso \
        < /dev/null
```

We have finished the entire procedure for building a guest image and booting it on the Palacios VMM. For more updated details, check the Palacios website `http://www.v3vee.org/palacios` and Kitten website `https://software.sandia.gov/trac/kitten` regularly.