- Immutable Page
- Info
- Attachments
- More Actions:

- Ubuntu Wiki
- Login
- Help

GitKernelBuild

Many times patches for bugs are committed upstream but have yet to make their way down to the Ubuntu kernel. It is often helpful if users are able to verify if the upstream patches do indeed resolve the issue they are witnessing. Likewise, in the opposite situation, it is useful to know if a bug may still exist upstream.

Contents

The following document should help users build their own kernel from the upstream mainline kernel to help verify if a bug still exists or not. If a bug is still present in the upstream kernel, it is encouraged that the bug be reported upstream following the procedure noted in this page. Please note that the following steps are targeted towards Ubuntu users and focuses on building the mainline kernel from the git repository at http://git.kernel.org.

For building Ubuntu kernels please refer to https://help.ubuntu.com/community/Kernel/Compile.

Prerequisites

There are a few tools that are necessary in order to build your own kernel(s). The 'git' (or 'git-core' for 10.04 or before) package provides the git revision control system which will be used to clone the mainline git repository. The 'kernel-package' provides the make-kpkg utility which automatically build your kernel and generate the linux-image and linux-header .deb files which can be installed. You will need to install the following packages:

```
sudo apt-get install git build-essential kernel-package fakeroot libncurses5-dev libssl-dev ccache bison flex
```

Kernel Build and Installation

1. Change to the directory where you want to clone the git tree. In this example we will use $HOME:

   ```
   cd $HOME
   ```

2. Clone the mainline kernel git tree:

   ```
   git clone git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git
   ```

3. Change to directory linux:

   ```
   cd linux
   ```

   If you only want to test all commits up to and including a specific one, instead of every commit you just downloaded, one may execute:

   ```
   git checkout COMMIT
   ```

   where COMMIT is the specific commit number (ex. 9587190107d0c0cbaccbf7bf6b0245d29095a9ae).

4. Copy the kernel config file from your existing system to the kernel tree:

   ```
   cp /boot/config-`uname -r` .config
   ```

5. Bring the config file up to date. Answer any questions that get prompted. Unless you know you are interested in a particular feature, accepting the default option by pressing Enter should be a safe choice:

   ```
   make oldconfig
   ```

   In cases where your kernel source is significantly newer than the existing config file, you'll be presented with all of the new config options for which there is no existing config file setting. You can either sit there and keep hitting Enter to take the default (generally safe), or you can just run:

   ```
   yes '' | make oldconfig
   ```

   which emulates exactly the same thing and saves you all that time.

6. (optional) If you need to make any kernel config changes, do the following and save your changes when prompted:

   ```
   make menuconfig
   ```

7. Clean the kernel source directory:

   ```
   make clean
   ```

8. Build the linux-image and linux-header .deb files using a thread per core + 1. This process takes a lot of time:

   ```
   make -j `getconf _NPROCESSORS_ONLN` deb-pkg LOCALVERSION=-custom
   ```

   With this command the package names will be something like linux-image-2.6.24-rc5-custom and linux-headers-2.6.24-rc5-custom, and in that case the version will be 2.6.24-rc5-custom-10.00.Custom. You may change the string "custom" into something else by changing the `LOCALVERSION` option.

9. Change to one directory level up (this is where the linux-image and linux-header .deb files were put):

```
cd ..
```

10. Now install the .deb files. In this example, the files are linux-image-2.6.24-rc5-custom_2.6.24-rc5-custom-10.00.Custom_i386.deb and linux-headers-2.6.24-rc5-custom_2.6.24-rc5-custom-10.00.Custom_i386.deb. You may receive warnings about '/lib/firmware/2.6.24-rc5-custom/' - this is expected and will only be problematic if the driver you are trying to test requires firmware:

```
sudo dpkg -i linux-image-2.6.24-rc5-custom_2.6.24-rc5-custom-10.00.Custom_i386.deb
sudo dpkg -i linux-headers-2.6.24-rc5-custom_2.6.24-rc5-custom-10.00.Custom_i386.deb
```

11. You are now ready to boot into your new kernel. Just make sure you select the new kernel when you boot:

```
sudo reboot
```

Using Ubuntu Kernel Configuration

The basic instructions provided above work well if you are building your own custom kernel. However, if you want to build a kernel that matches the official Ubuntu kernel package configuration as much as possible a few extra steps are needed. Note that if you are simply trying to build the ubuntu kernel, you should be following the https://help.ubuntu.com/community/Kernel/Compile guide instead of this one.

1. Perform steps 1-7 above, use the Ubuntu kernel config in step 4

2. Override the kernel-package default packaging scripts with the Ubuntu packaging scripts:

```
cd $HOME
git clone git://kernel.ubuntu.com/ubuntu/ubuntu-lucid.git
cp -a /usr/share/kernel-package ubuntu-package
cp ubuntu-lucid/debian/control-scripts/{postinst,postrm,preinst,prerm} ubuntu-package/pkg/image/
cp ubuntu-lucid/debian/control-scripts/headers-postinst ubuntu-package/pkg/headers/
```

3. Build packages using overlay directory:

```
cd $HOME/linux
CONCURRENCY_LEVEL=`getconf _NPROCESSORS_ONLN` fakeroot make-kpkg --initrd --append-to-version=-custom --overlay-dir=$HOME/u
```

4. Perform steps 9-11 above.

Note: The "--overlay-dir" option is only available in Lucid or later. If you need to build a kernel on a previous distribution, either install a backport of kernel-package if available, or manually edit /usr/share/kernel-package as needed.

KernelTeam/GitKernelBuild (last edited 2019-04-19 16:23:17 by neyder @ localhost[127.0.0.1]:neyder)