L34
Rebuilding your Linux/390 Kernel from source
Mark Post

zSeries Expo | Nov. 1 - 5, 2004
Miami, FL
Documentation

• The Linux Documentation Project
  http://www.tldp.org/

• Look for the “Kernel HOWTO”
  http://www.tldp.org/HOWTO/Kernel-HOWTO/
  http://www.digitalhermit.com/~kwan/kernel.html

• Practical experience here this week:
  Linux for S/390 Installation Lab, Tue. & Thu. 9227 and 9230
Basic Process

• Get the source
• Unpack/install the source
• Apply IBM patches (if not already there)
• Generate a kernel configuration
  – make menuconfig
  – make oldconfig
  – make xconfig
  – make config
• Run
  – make dep
  – make image
  – make modules
  – make modules_install
Basic Process (2)

- Put new kernel into place
- Possibly regenerate the initrd
- Possibly update /etc/zipl.conf
- Run zipl
- Take the system down
- Boot from the new kernel
- Back off to the old kernel if necessary
Where to get the source

- **“Pristine” source:**

- **Linux distribution-specific source:**
  Usually included in your distribution installation media, or...
  https://portal.suse.com/

- **IBM patches:**
Unpack/Install the Source

• If you get a kernel source RPM, then install the source:
  `rpm -ivh kernel-source.rpm`
  
  – Usually puts the source in `/usr/src/linux-$VERSION`

• If you downloaded source from ftp.kernel.org:
  `tar -zxvf linux-2.6.7.tar.gz`
  `tar -jxvf linux-2.6.7.tar.bz2`

• Don’t confuse this with a kernel SRPM
  
  – `kernel-source-2.4.20.SuSE-62.i586.rpm`
  `kernel-source-2.4.20.SuSE-62.src.rpm`
  `kernel-source-2.4.20-8.i386.rpm`
  `kernel-2.4.20-8.src.rpm`
Unpack/Install the Source

- So what is the difference?
  - SRPM = vanilla source, patches, RPM spec file
    gets installed into /usr/src/rpm/SOURCES
    gets processed with "rpmbuild -bb" command
  - RPM = updated source
    gets installed into /usr/src/linux-$VERSION (usually)
    /usr/src/linux-2.4.19
Apply IBM patches

• Patches come in .tar.gz files.
• Contain a
  – LICENSE file (GPL)
  – .readme file
  – .diff file
• Read the .readme file(s) for patching order.
• cd to top-level directory and use patch command:
  cat /path/to/diff.file | patch -p1 [ ---dry-run ]
• Repeat for each .diff file.
Generate a kernel configuration

• Many ways to specify a particular kernel configuration:
  – make config (don't do this)
  – make oldconfig (used to start from a known configuration)
  – make menuconfig
  – make xconfig (don’t do this on Linux/390)
Make menuconfig
Make menuconfig (2)
Don’t do this at home
<table>
<thead>
<tr>
<th>Code maturity level options</th>
<th>Multi-device support (RAID and LVM)</th>
<th>Kernel hacking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loadable module support</td>
<td>Character device drivers</td>
<td>Save and Exit</td>
</tr>
<tr>
<td>Processor type and features</td>
<td>Network device drivers</td>
<td>Quit Without Saving</td>
</tr>
<tr>
<td>General setup</td>
<td>Miscellaneous</td>
<td>Load Configuration from File</td>
</tr>
<tr>
<td>SCSI support</td>
<td>Networking options</td>
<td>Store Configuration to File</td>
</tr>
<tr>
<td>Block device drivers</td>
<td>File systems</td>
<td></td>
</tr>
</tbody>
</table>
### Block device drivers

<table>
<thead>
<tr>
<th>Feature</th>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loopback device support</td>
<td>y</td>
<td>RAM disk support</td>
</tr>
<tr>
<td>Network block device support</td>
<td>y</td>
<td>RAM disk support</td>
</tr>
<tr>
<td>RAM disk support</td>
<td>m</td>
<td>Default RAM disk size</td>
</tr>
<tr>
<td>Initial RAM disk (initrd) support</td>
<td>y</td>
<td>S/390 block device drivers</td>
</tr>
<tr>
<td>XPRAM disk support</td>
<td>m</td>
<td>Support for DASD devices</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>Support for ECKD Disks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Automatic activation of ECKD module</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Support for FBA Disks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Automatic activation of FBA module</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Support for DIAG access to CMS reserved Disks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Automatic activation of DIAG module</td>
</tr>
</tbody>
</table>

**Main Menu**  **Next**  **Prev**
Usual order of commands:

- Save configuration file
- make mrproper
  (this wipes out .config!)
- copy saved
  configuration file to
  .config
- make menuconfig
  (or oldconfig)
- make dep
  (no longer needed in 2.6.x kernels)
- make image
  (on Intel, will likely be
  bzImage)
- make install
  (make sure you know what this
  does)
- make modules
- make modules_install
- depmod -a version-of-
  kernel-just-built
  • depmod -a 2.4.19-xfs
Put new kernel into place

• The generated kernel is going to be:
  /path/to/linux/source/arch/s390/boot/image
    AKA
  arch/s390/boot/image
• Copy the image file to /boot/
• Copy the System.map file to /boot/
  (located in the top-level source directory)
• Copy the .config file to /boot/
  (give it a name like config-2.4.26[-something] )
Regenerate the initrd

• Newer versions of SUSE and Red Hat use an initial ramdisk to hold driver modules
• Updating the kernel and/or kernel modules requires that the initrd be re-created
• The command that does this is “mkinitrd.”
  – Read the man page for this to understand what it does.
  – Look inside the initrd to see what’s in the old one, versus the new one.
  – Look at http://linuxvm.org/Info/HOWTOs/mkinitrd-notes.html
Update /etc/zipl.conf

• Review the contents of /etc/zipl.conf
• If you need to make a change, do so
  – Correct kernel
  – Correct default kernel
  – Correct DASD volume to write the kernel
  – Correct kernel parameters specified
Re-run zipl

- If you use `/etc/zipl.conf`, just type in “zipl”
- If you don’t use `/etc/zipl.conf`, then you’ll have to specify all the parameters:
  - `zipl -t /boot -i /boot/image-2.4.26 -p /boot/parmfile -r /boot/ramdisk`
- Make sure you get messages similar to this:
  Building bootstrap './bootmap'
  Adding IPL section
    kernel image......: image at 0x10000
    kernel parmline...: 'dasd=300-305,400 root=/dev/dasda1 ro
noinitrd' at 0x1000
  Preparing boot device: dasda (0300).
  Done.
Take the system down

- shutdown -h now
- shutdown -h 23:59
- Whatever your site’s change management dictates.
Boot from the new kernel

- In an LPAR - from the HMC
- From z/VM - ipl devno clear
- How do you know what to specify for the boot device number?
  - From the /boot directory:
    df -h .
    grep dasd? /proc/dasd/devices
    First number is the device number
Back off to the old kernel

• How do you do that, when you just over-wrote your old kernel information?
  – You need multiple DASD volumes/minidisks (not LVM or RAID)
  – Create a boot directory (or some other name) in each file system
  – Copy the files from /boot, and your new kernel, etc.
  – Re-run zipl from that directory or add entries to /etc/zipl.conf and change your default

# df -h

<table>
<thead>
<tr>
<th>Filesystem</th>
<th>Size</th>
<th>Used</th>
<th>Avail</th>
<th>Use%</th>
<th>Mounted on</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/dasda1</td>
<td>2.3G</td>
<td>348M</td>
<td>1.8G</td>
<td>17%</td>
<td>/</td>
</tr>
<tr>
<td>/dev/dasdb1</td>
<td>2.3G</td>
<td>1.3G</td>
<td>848M</td>
<td>61%</td>
<td>/usr</td>
</tr>
</tbody>
</table>