Experiment 14: Linux Kernel

Aarya R Shankar

May 13, 2017

To view the source code:
`https://github.com/torvalds/linux/tree/master/Documentation`

1 Aim

Kernel configuration, compilation and installation : Download/access the latest kernel source code from kernel.org, compile the kernel and install it in the local system. Try to view the source code of the kernel.

2 Procedure

First download the latest kernel from kernel.org using curl or wget.

```
curl https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.11.tar.xz
-o linux-4.11.tar.gz
```

The downloaded file is in tar.gz format. To unzip it, do

```
tar -xvf linux-4.11.tar.gz
```

You will see a list of files that is being unzipped from the downloaded file. Now if you do ls, you will see a new directory named linux-4.11 Move into that directory

```
cd linux-4.11
```

Now that we have downloaded our kernel source code and expanded it, we need to make a configuration file, .config, to configure the kernel with our desired options. Note: We can create the configuration file from scratch (with

our desired options), or based on a default configuration, or based on the currently running kernel verion, or based on some kernel release distribution. Here first, we'll create a .config file from scratch

```
make config
```

After running this command, you'll be required to answer a number of questions like:

1. for which system, are you building the kernel (32 or 64 bit)

2. if you want cross compile tools (press enter if you do not compile code to be run on other processors, else give "x86_64-pc-linux-gnu-" for 64 bit PCs, or "arm-unknown-linux-gnu-" for ARM processor systems)

3. what should be appended to our custom kernel version (ex: If you give firstcustombuild, it will display XXX-4.11-firstcustombuild)

4. The kernel compression mode we prefer (Gzip is the default option)

5. Our preferred default hostname (usually developers leave this blank, so that all linux users can chose their own hostname)

6. Support for swap memory (If you have a dual boot, or are planning to get one, choose yes)

7. Preference for IPC (Inter Process Communication) (It is always best to enable this, if you want all applications to run)

8. Preference for POSIX Message Queues so that messages can be labelled with a priority (Choose yes !important)

9. Preference for Cross Memory Attach so that privileged processes can write or read from other prcoesses' memory (Allowing it is preferred)

10. Preference for uselib syscalls (CONFIG_USELIB) (If you're sure your system runs on glibc library, type no. Else run the command 'ldd –version' in another terminal, if you see your current glibc version, type n, else type yes)

11. Preference for IRQ Debug (type y if you want to know the mapping relationship between hardware irq numbers and Linux irq numbers. The mapping is exposed via debugfs in the file "irq_domain_mapping". Type n if you don't understand this)

12. Preference for Timer Tick handle (Choose NO_HZ_IDLE or NO_HZ_FULL for lesser cpu usage)

13. Preference for Old Idle dynticks config (If your system hardware is slow or old, choose no. Else type yes so that the timer interrupts can be used whenever needed, so as to enable tasks to be executed at particular interval of time)

14. Preference for High Resolution Timer (Most relatively modern systems (Pentium III and higher) have high resolution timers, allowing for more precise timing. Not really mandatory, but some applications like mplayer can benefit from using hi-res timers. You can manually check if your system supports high resolution timer by looking at the values given in /proc/timer_list)

15. Preference for Cputime accounting (TICK_CPU_ACCOUNTING is preferred as it has simple accounting)

16. Fine granularity task level IRQ time accounting (IRQ_TIME_ACCOUNTING) (Default is no)

17. BSD_PROCESS_ACCT (It basically logs all the processes that runs in the kernel. Choose no if you want a smaller and faster kernel)

18. CPU/Task time and stats accounting (Some questions are already given as yes. For TASK_XACCT, choose no if you want a small kernel as it

collects extended task accounting and sends it to userland for processing over the taskstats interface).

19. CONFIG_RCU_EXPERT (Choose yes only if you want to make expert level modifications to Read-Copy-Update configurations)

20. IKCONFIG (This option makes the whole .config file to be save in the kernel with proper documentation, so that it can be extracted form the kernel image using the command scripts/extract-ikconfig and be used to rebuild the current kernel or to build another one. We can even extract it form a running kernel by reading /proc/config.gz)

There are many more questions. There will be a '?' option along with y or n or m, we can choose that to know more about the question.

Now we want to make the configuration file based on the default configuration options (preferred, unless you really know what you're doing and have lots of time (hours) to read through all the questions and give the necessary answers. However avoiding the unnecessary options will make for a smaller and faster kernel).

```
make defconfig
```

For further manual configuration, after setting up the .config file, run

```
make menuconfig
```

or

```
make xconfig screen
```

If you want specific menu, give it as an attribute at the end of the command.

Now that we have a .config file, we can build the kernel by running

```
make
```

Now we can install it by running the installation script of the distribution, which may be used by the kernel build system to automatically install a built

kernel into the proper location and modify the bootloader so that nothing extra needs to be done by the developer.

If you have built any modules and want to use use this method to install a kernel, first enter:

```
make modules_install
```

This will install all the modules that you have built and place them in the proper location in the filesystem for the new kernel to properly find. Modules are placed in the /lib/modules/ KERNEL_VERSION directory, where KERNEL_VERSION is the kernel version of the new kernel you have just built.

After the modules have been successfully installed, we now need to install the main kernel image.

```
make install
```

This command will verify that the kernel has been successfully built properly, the build system will install the static kernel portion into the /boot directory and name this executable file based on the kernel version of the built kernel, etc

After this is finished, the kernel is successfully installed, and you can safely reboot and try out your new kernel image. Note that this installation does not overwrite any older kernel images, so if there is a problem with your new kernel image, the old kernel can be selected at boot time.

3 Result

The linux kernel is successfully downloaded, modified, compiled, and installed as given.