KernelNewbies

KernelHeaders

Header files in the Linux kernel are used for two purposes:

1. to define interfaces between components of the kernel, and
2. to define interfaces between the kernel and user space

Internal modules

Internal interfaces between modules are defined anywhere in below linux/include/ or linux/arch/*/include/. Interfaces between source files in a single module should be put into the same directory as the module source code, which avoids polluting the global header space.

External modules

In order to build external kernel modules, you need the exact version of the headers from the kernel that the modules are built for. http://lwn.net/Articles/21823/ explains how to write an external Makefile to use those headers. When you do this, the kernel Makefiles will automatically choose the correct include path for both source and build directory and for using the include/asm headers from the right architecture.

While traditionally the kernel source was installed in /usr/src/linux, this is no longer supported for building external modules. Instead, your Makefile should point to /lib/modules/${kver}/build, where ${kver} is the exact version string of the kernel, e.g. the output of *uname -r* for the currently running kernel.

User space programs

In general, user space programs are built against the header files provided by the distribution, typically from a package named *glibc-devel*, *glibc-kernheaders* or *linux-libc-dev*. These header files are often from an older kernel version, and they cannot safely be replaced without rebuilding glibc as well. In particular, installing /usr/include/linux as a symbolic link to /usr/src/linux/include or /lib/modules/*/build/include/linux is highly discouraged as it frequently breaks rebuilding applications. For instance, older kernels had the architecture specific header files in include/asm-${arch} instead of arch/${arch}/include/asm and had on a symlink to the architecture specific directory.

The correct way to package the header files for a distribution is to run 'make headers_install' from the kernel source directory to install the headers into /usr/include and then rebuild the C library package, with a dependency on the specific version of the just installed kernel headers.

If you are distributing a user space program that depends on a specific version of some kernel headers, e.g. because your program runs only on patched or very recent kernels, you cannot rely on the headers in /usr/include. You also cannot use the header files from /usr/src/linux/include or /lib/modules/*/build/include/ because they have not been prepared for inclusion in user space. The kernel should warn you about this if you try it and point you to this Wiki page. The correct way to address this problem is to isolate the specific interfaces that you need, e.g. a single header file that is patched in a new kernel providing the ioctl numbers for a character device used by your program. In your own program, add a copy of that source file, with a notice that it should be kept in sync with new kernel versions. If your program is not licensed under GPLv2, make sure you have permission from the author of that file to distribute it under the license of your own program. Since your program now depends on kernel interfaces that may not be present in a regular kernel, it's a good idea to add run-time checks that make sure the kernel understands the interface and give a helpful error message if there is no fallback to an older interface.