Linux Kernel patch submission checklist

Here are some basic things that developers should do if they want to see their kernel patch submissions accepted more quickly.

These are all above and beyond the documentation that is provided in :ref:`Documentation/process/submitting-patches.rst <submittingpatches>` and elsewhere regarding submitting Linux kernel patches.

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\process\[linux-master][Documentation][process]submit-checklist.rst`, line 9); *backlink***
>
> Unknown interpreted text role "ref".

1. If you use a facility then #include the file that defines/declares that facility. Don't depend on other header files pulling in ones that you use.
2. Builds cleanly:

   a. with applicable or modified `CONFIG` options `=y`, `=m`, and `=n`. No `gcc` warnings/errors, no linker warnings/errors.
   b. Passes `allnoconfig`, `allmodconfig`
   c. Builds successfully when using `O=builddir`
   d. Any Documentation/ changes build successfully without new warnings/errors. Use `make htmldocs` or `make pdfdocs` to check the build and fix any issues.

3. Builds on multiple CPU architectures by using local cross-compile tools or some other build farm.
4. ppc64 is a good architecture for cross-compilation checking because it tends to use `unsigned long` for 64-bit quantities.
5. Check your patch for general style as detailed in :ref:`Documentation/process/coding-style.rst <codingstyle>`. Check for trivial violations with the patch style checker prior to submission (`scripts/checkpatch.pl`). You should be able to justify all violations that remain in your patch.

   > **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\process\[linux-master][Documentation][process]submit-checklist.rst`, line 37); *backlink***
   >
   > Unknown interpreted text role "ref".

6. Any new or modified `CONFIG` options do not muck up the config menu and default to off unless they meet the exception criteria documented in `Documentation/kbuild/kconfig-language.rst` Menu attributes: default value.
7. All new `Kconfig` options have help text.
8. Has been carefully reviewed with respect to relevant `Kconfig` combinations. This is very hard to get right with testing -- brainpower pays off here.
9. Check cleanly with sparse.
10. Use `make checkstack` and fix any problems that it finds.

    > **Note**
    >
    > `checkstack` does not point out problems explicitly, but any one function that uses more than 512 bytes on the stack is a candidate for change.

11. Include :ref:`kernel-doc <kernel_doc>` to document global kernel APIs. (Not required for static functions, but OK there also.) Use `make htmldocs` or `make pdfdocs` to check the :ref:`kernel-doc <kernel_doc>` and fix any issues.

    > **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\process\[linux-master][Documentation][process]submit-checklist.rst`, line 64); *backlink***
    >
    > Unknown interpreted text role "ref".

    > **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\process\[linux-master][Documentation][process]submit-checklist.rst`, line 64); *backlink***
    >
    > Unknown interpreted text role "ref".

12. Has been tested with `CONFIG_PREEMPT`, `CONFIG_DEBUG_PREEMPT`, `CONFIG_DEBUG_SLAB`, `CONFIG_DEBUG_PAGEALLOC`, `CONFIG_DEBUG_MUTEXES`, `CONFIG_DEBUG_SPINLOCK`, `CONFIG_DEBUG_ATOMIC_SLEEP`, `CONFIG_PROVE_RCU` and `CONFIG_DEBUG_OBJECTS_RCU_HEAD` all simultaneously enabled.
13. Has been build- and runtime tested with and without `CONFIG_SMP` and `CONFIG_PREEMPT`.
14. All codepaths have been exercised with all lockdep features enabled.
15. All new `/proc` entries are documented under `Documentation/`
16. All new kernel boot parameters are documented in `Documentation/admin-guide/kernel-parameters.rst`.
17. All new module parameters are documented with `MODULE_PARM_DESC()`
18. All new userspace interfaces are documented in `Documentation/ABI/`. See `Documentation/ABI/README` for more information. Patches that change userspace interfaces should be CCed to linux-api@vger.kernel.org.
19. Has been checked with injection of at least slab and page-allocation failures. See `Documentation/fault-injection/`.

    If the new code is substantial, addition of subsystem-specific fault injection might be appropriate.
20. Newly-added code has been compiled with `gcc -W` (use `make KCFLAGS=-W`). This will generate lots of noise, but is good for finding bugs like "warning: comparison between signed and unsigned".
21. Tested after it has been merged into the -mm patchset to make sure that it still works with all of the other queued patches and various changes in the VM, VFS, and other subsystems.
22. All memory barriers {e.g., `barrier()`, `rmb()`, `wmb()`} need a comment in the source code that explains the logic of what they are doing and why.
23. If any ioctl's are added by the patch, then also update `Documentation/userspace-api/ioctl/ioctl-number.rst`.
24. If your modified source code depends on or uses any of the kernel APIs or features that are related to the following `Kconfig` symbols, then test multiple builds with the related `Kconfig` symbols disabled and/or `=m` (if that option is available) [not all of these at the same time, just various/random combinations of them]:

    `CONFIG_SMP`, `CONFIG_SYSFS`, `CONFIG_PROC_FS`, `CONFIG_INPUT`, `CONFIG_PCI`, `CONFIG_BLOCK`, `CONFIG_PM`, `CONFIG_MAGIC_SYSRQ`, `CONFIG_NET`, `CONFIG_INET=n` (but latter with `CONFIG_NET=y`).