Reporting issues

The short guide (aka TL;DR)

Are you facing a regression with vanilla kernels from the same stable or longterm series? One still supported? Then search the LKML and the Linux stable mailing list archives for matching reports to join. If you don't find any, install the latest release from that series. If it still shows the issue, report it to the stable mailing list (stable@vger.kernel.org) and CC the regressions list (regressions@lists.linux.dev); ideally also CC the maintainer and the mailing list for the subsystem in question.

In all other cases try your best guess which kernel part might be causing the issue. Check the :ref:`MAINTAINERS <maintainers>` file for how its developers expect to be told about problems, which most of the time will be by email with a mailing list in CC. Check the destination's archives for matching reports; search the LKML and the web, too. If you don't find any to join, install the latest mainline kernel. If the issue is present there, send a report.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\(linux-master) (Documentation) (admin-guide) reporting-issues.rst`, **line 21**); ***backlink***

Unknown interpreted text role "ref".

The issue was fixed there, but you would like to see it resolved in a still supported stable or longterm series as well? Then install its latest release. If it shows the problem, search for the change that fixed it in mainline and check if backporting is in the works or was discarded; if it's neither, ask those who handled the change for it.

General remarks: When installing and testing a kernel as outlined above, ensure it's vanilla (IOW: not patched and not using add-on modules). Also make sure it's built and running in a healthy environment and not already tainted before the issue occurs.

If you are facing multiple issues with the Linux kernel at once, report each separately. While writing your report, include all information relevant to the issue, like the kernel and the distro used. In case of a regression, CC the regressions mailing list (regressions@lists.linux.dev) to your report. Also try to pin-point the culprit with a bisection; if you succeed, include its commit-id and CC everyone in the sign-off-by chain.

Once the report is out, answer any questions that come up and help where you can. That includes keeping the ball rolling by occasionally retesting with newer releases and sending a status update afterwards.

Step-by-step guide how to report issues to the kernel maintainers

The above TL;DR outlines roughly how to report issues to the Linux kernel developers. It might be all that's needed for people already familiar with reporting issues to Free/Libre & Open Source Software (FLOSS) projects. For everyone else there is this section. It is more detailed and uses a step-by-step approach. It still tries to be brief for readability and leaves out a lot of details; those are described below the step-by-step guide in a reference section, which explains each of the steps in more detail.

Note: this section covers a few more aspects than the TL;DR and does things in a slightly different order. That's in your interest, to make sure you notice early if an issue that looks like a Linux kernel problem is actually caused by something else. These steps thus help to ensure the time you invest in this process won't feel wasted in the end:

- Are you facing an issue with a Linux kernel a hardware or software vendor provided? Then in almost all cases you are better off to stop reading this document and reporting the issue to your vendor instead, unless you are willing to install the latest Linux version yourself. Be aware the latter will often be needed anyway to hunt down and fix issues.
- Perform a rough search for existing reports with your favorite internet search engine; additionally, check the archives of the Linux Kernel Mailing List (LKML). If you find matching reports, join the discussion instead of sending a new one.
- See if the issue you are dealing with qualifies as regression, security issue, or a really severe problem: those are 'issues of high priority' that need special handling in some steps that are about to follow.
- Make sure it's not the kernel's surroundings that are causing the issue you face.
- Create a fresh backup and put system repair and restore tools at hand.
- Ensure your system does not enhance its kernels by building additional kernel modules on-the-fly, which solutions like DKMS might be doing locally without your knowledge.
- Check if your kernel was 'tainted' when the issue occurred, as the event that made the kernel set this flag might be causing the issue you face.
- Write down coarsely how to reproduce the issue. If you deal with multiple issues at once, create separate notes for each of them and make sure they work independently on a freshly booted system. That's needed, as each issue needs to get reported to the kernel developers separately, unless they are strongly entangled.
- If you are facing a regression within a stable or longterm version line (say something broke when updating from 5.10.4 to 5.10.5), scroll down to 'Dealing with regressions within a stable and longterm kernel line'.
- Locate the driver or kernel subsystem that seems to be causing the issue. Find out how and where its developers expect reports. Note: most of the time this won't be bugzilla.kernel.org, as issues typically need to be sent by mail to a maintainer and a public mailing list.
- Search the archives of the bug tracker or mailing list in question thoroughly for reports that might match your issue. If you find anything, join the discussion instead of sending a new report.

After these preparations you'll now enter the main part:

- Unless you are already running the latest 'mainline' Linux kernel, better go and install it for the reporting process. Testing and reporting with the latest 'stable' Linux can be an acceptable alternative in some situations; during the merge window that actually might be even the best approach, but in that development phase it can be an even better idea to suspend your efforts for a few days anyway. Whatever version you choose, ideally use a 'vanilla' build. Ignoring these advices will dramatically increase the risk your report will be rejected or ignored.
- Ensure the kernel you just installed does not 'taint' itself when running.
- Reproduce the issue with the kernel you just installed. If it doesn't show up there, scroll down to the instructions for issues only happening with stable and longterm kernels.
- Optimize your notes: try to find and write the most straightforward way to reproduce your issue. Make sure the end result has all the important details, and at the same time is easy to read and understand for others that hear about it for the first time. And if you learned something in this process, consider searching again for existing reports about the issue.
- If your failure involves a 'panic', 'Oops', 'warning', or 'BUG', consider decoding the kernel log to find the line of code that triggered the error.
- If your problem is a regression, try to narrow down when the issue was introduced as much as possible.
- Start to compile the report by writing a detailed description about the issue. Always mention a few things: the latest kernel version you installed for reproducing, the Linux Distribution used, and your notes on how to reproduce the issue. Ideally, make the kernel's build configuration (.config) and the output from `dmesg` available somewhere on the net and link to it. Include or upload all other information that might be relevant, like the output/screenshot of an Oops or the output from `lspci`. Once you wrote this main part, insert a normal length paragraph on top of it outlining the issue and the impact quickly. On top of this add one sentence that briefly describes the problem and gets people to read on. Now give the thing a descriptive title or subject that yet again is shorter. Then you're ready to send or file the report like the MAINTAINERS file told you, unless you are dealing with one of those 'issues of high priority': they need special care which is explained in 'Special handling for high priority issues' below.
- Wait for reactions and keep the thing rolling until you can accept the outcome in one way or the other. Thus react publicly and in a timely manner to any inquiries. Test proposed fixes. Do proactive testing: retest with at least every first release candidate (RC) of a new mainline version and report your results. Send friendly reminders if things stall. And try to help yourself, if you don't get any help or if it's unsatisfying.

Reporting regressions within a stable and longterm kernel line

This subsection is for you, if you followed above process and got sent here at the point about regression within a stable or longterm kernel version line. You face one of those if something breaks when updating from 5.10.4 to 5.10.5 (a switch from 5.9.15 to 5.10.5 does not qualify). The developers want to fix such regressions as quickly as possible, hence there is a streamlined process to report them:

- Check if the kernel developers still maintain the Linux kernel version line you care about: go to the front page of kernel.org and make sure it mentions the latest release of the particular version line without an '[EOL]' tag.
- Check the archives of the Linux stable mailing list for existing reports.
- Install the latest release from the particular version line as a vanilla kernel. Ensure this kernel is not tainted and still shows the problem, as the issue might have already been fixed there. If you first noticed the problem with a vendor kernel, check a vanilla build of the last version known to work performs fine as well.
- Send a short problem report to the Linux stable mailing list (stable@vger.kernel.org) and CC the Linux regressions mailing list (regressions@lists.linux.dev); if you suspect the cause in a particular subsystem, CC its maintainer and its mailing list. Roughly describe the issue and ideally explain how to reproduce it. Mention the first version that shows the problem and the last version that's working fine. Then wait for further instructions.

The reference section below explains each of these steps in more detail.

Reporting issues only occurring in older kernel version lines

This subsection is for you, if you tried the latest mainline kernel as outlined above, but failed to reproduce your issue there; at the same time you want to see the issue fixed in a still supported stable or longterm series or vendor kernels regularly rebased on those. If that the case, follow these steps:

- Prepare yourself for the possibility that going through the next few steps might not get the issue solved in older releases: the fix might be too big or risky to get backported there.
- Perform the first three steps in the section "Dealing with regressions within a stable and longterm kernel line" above.
- Search the Linux kernel version control system for the change that fixed the issue in mainline, as its commit message might tell you if the fix is scheduled for backporting already. If you don't find anything that way, search the appropriate mailing lists for posts that discuss such an issue or peer-review possible fixes; then check the discussions if the fix was deemed unsuitable for backporting. If backporting was not considered at all, join the newest discussion, asking if it's in the cards.
- One of the former steps should lead to a solution. If that doesn't work out, ask the maintainers for the subsystem that seems to be causing the issue for advice; CC the mailing list for the particular subsystem as well as the stable mailing list.

The reference section below explains each of these steps in more detail.

Reference section: Reporting issues to the kernel maintainers

The detailed guides above outline all the major steps in brief fashion, which should be enough for most people. But sometimes there are situations where even experienced users might wonder how to actually do one of those steps. That's what this section is for, as it will provide a lot more details on each of the above steps. Consider this as reference documentation: it's possible to read it from top to bottom. But it's mainly meant to skim over and a place to look up details how to actually perform those steps.

A few words of general advice before digging into the details:

- The Linux kernel developers are well aware this process is complicated and demands more than other FLOSS projects. We'd love to make it simpler. But that would require work in various places as well as some infrastructure, which would need constant maintenance; nobody has stepped up to do that work, so that's just how things are for now.
- A warranty or support contract with some vendor doesn't entitle you to request fixes from developers in the upstream Linux kernel community: such contracts are completely outside the scope of the Linux kernel, its development community, and this document. That's why you can't demand anything such a contract guarantees in this context, not even if the developer handling the issue works for the vendor in question. If you want to claim your rights, use the vendor's support channel instead. When doing so, you might want to mention you'd like to see the issue fixed in the upstream Linux kernel; motivate them by saying it's the only way to ensure the fix in the end will get incorporated in all Linux distributions.
- If you never reported an issue to a FLOSS project before you should consider reading How to Report Bugs Effectively, How To Ask Questions The Smart Way, and How to ask good questions.

With that off the table, find below the details on how to properly report issues to the Linux kernel developers.

Make sure you're using the upstream Linux kernel

> *Are you facing an issue with a Linux kernel a hardware or software vendor provided? Then in almost all cases you are better off to stop reading this document and reporting the issue to your vendor instead, unless you are willing to install the latest Linux version yourself. Be aware the latter will often be needed anyway to hunt down and fix issues.*

Like most programmers, Linux kernel developers don't like to spend time dealing with reports for issues that don't even happen with their current code. It's just a waste everybody's time, especially yours. Unfortunately such situations easily happen when it comes to the kernel and often leads to frustration on both sides. That's because almost all Linux-based kernels pre-installed on devices (Computers, Laptops, Smartphones, Routers, â€¦) and most shipped by Linux distributors are quite distant from the official Linux kernel as distributed by kernel.org: these kernels from these vendors are often ancient from the point of Linux development or heavily modified, often both.

Most of these vendor kernels are quite unsuitable for reporting issues to the Linux kernel developers: an issue you face with one of them might have been fixed by the Linux kernel developers months or years ago already; additionally, the modifications and enhancements by the vendor might be causing the issue you face, even if they look small or totally unrelated. That's why you should report issues with these kernels to the vendor. Its developers should look into the report and, in case it turns out to be an upstream issue, fix it directly upstream or forward the report there. In practice that often does not work out or might not what you want. You thus might want to consider circumventing the vendor by installing the very latest Linux kernel core yourself. If that's an option for you move ahead in this process, as a later step in this guide will explain how to do that once it rules out other potential causes for your issue.

Note, the previous paragraph is starting with the word 'most', as sometimes developers in fact are willing to handle reports about issues occurring with vendor kernels. If they do in the end highly depends on the developers and the issue in question. Your chances are quite good if the distributor applied only small modifications to a kernel based on a recent Linux version; that for example often holds true for the mainline kernels shipped by Debian GNU/Linux Sid or Fedora Rawhide. Some developers will also accept reports about issues with kernels from distributions shipping the latest stable kernel, as long as its only slightly modified; that for example is often the case for Arch Linux, regular Fedora releases, and openSUSE Tumbleweed. But keep in mind, you better want to use a mainline Linux and avoid using a stable kernel for this process, as outlined in the section 'Install a fresh kernel for testing' in more detail.

Obviously you are free to ignore all this advice and report problems with an old or heavily modified vendor kernel to the upstream Linux developers. But note, those often get rejected or ignored, so consider yourself warned. But it's still better than not reporting the issue at all: sometimes such reports directly or indirectly will help to get the issue fixed over time.

Search for existing reports, first run

> *Perform a rough search for existing reports with your favorite internet search engine; additionally, check the archives of the Linux Kernel Mailing List (LKML). If you find matching reports, join the discussion instead of sending a new one.*

Reporting an issue that someone else already brought forward is often a waste of time for everyone involved, especially you as the reporter. So it's in your own interest to thoroughly check if somebody reported the issue already. At this step of the process it's okay to just perform a rough search: a later step will tell you to perform a more detailed search once you know where your issue needs to be reported to. Nevertheless, do not hurry with this step of the reporting process, it can save you time and trouble.

Simply search the internet with your favorite search engine first. Afterwards, search the Linux Kernel Mailing List (LKML) archives.

If you get flooded with results consider telling your search engine to limit search timeframe to the past month or year. And wherever you search, make sure to use good search terms; vary them a few times, too. While doing so try to look at the issue from the perspective of someone else: that will help you to come up with other words to use as search terms. Also make sure not to use too many search terms at once. Remember to search with and without information like the name of the kernel driver or the name of the affected hardware component. But its exact brand name (say 'ASUS Red Devil Radeon RX 5700 XT Gaming OC') often is not much helpful, as it is too specific. Instead try search terms like the model line (Radeon 5700 or Radeon 5000) and the code name of the main chip ('Navi' or 'Navi10') with and without its manufacturer ('AMD').

In case you find an existing report about your issue, join the discussion, as you might be able to provide valuable additional information. That can be important even when a fix is prepared or in its final stages already, as developers might look for people that can provide additional information or test a proposed fix. Jump to the section 'Duties after the report went out' for details on how to get properly involved.

Note, searching bugzilla.kernel.org might also be a good idea, as that might provide valuable insights or turn up matching reports. If you find the latter, just keep in mind: most subsystems expect reports in different places, as described below in the section "Check where you need to report your issue". The developers that should take care of the issue thus might not even be aware of the bugzilla ticket. Hence, check the ticket if the issue already got reported as outlined in this document and if not consider doing so.

Issue of high priority?

> *See if the issue you are dealing with qualifies as regression, security issue, or a really severe problem: those are 'issues of high priority' that need special handling in some steps that are about to follow.*

Linus Torvalds and the leading Linux kernel developers want to see some issues fixed as soon as possible, hence there are 'issues of high priority' that get handled slightly differently in the reporting process. Three type of cases qualify: regressions, security issues, and really severe problems.

You deal with a regression if some application or practical use case running fine with one Linux kernel works worse or not at all with a newer version compiled using a similar configuration. The document Documentation/admin-guide/reporting-regressions.rst explains this in more detail. It also provides a good deal of other information about regressions you might want to be aware of; it for example explains how to add your issue to the list of tracked regressions, to ensure it won't fall through the cracks.

What qualifies as security issue is left to your judgment. Consider reading Documentation/admin-guide/security-bugs.rst before proceeding, as it provides additional details how to best handle security issues.

An issue is a 'really severe problem' when something totally unacceptably bad happens. That's for example the case when a Linux kernel corrupts the data it's handling or damages hardware it's running on. You're also dealing with a severe issue when the kernel suddenly stops working with an error message ('kernel panic') or without any farewell note at all. Note: do not confuse a 'panic' (a fatal error where the kernel stop itself) with a 'Oops' (a recoverable error), as the kernel remains running after the latter.

Ensure a healthy environment

> *Make sure it's not the kernel's surroundings that are causing the issue you face.*

Problems that look a lot like a kernel issue are sometimes caused by build or runtime environment. It's hard to rule out that problem completely, but you should minimize it:

- Use proven tools when building your kernel, as bugs in the compiler or the binutils can cause the resulting kernel to misbehave.
- Ensure your computer components run within their design specifications; that's especially important for the main processor, the main memory, and the motherboard. Therefore, stop undervolting or overclocking when facing a potential kernel issue.
- Try to make sure it's not faulty hardware that is causing your issue. Bad main memory for example can result in a multitude of issues that will manifest itself in problems looking like kernel issues.
- If you're dealing with a filesystem issue, you might want to check the file system in question with `fsck`, as it might be damaged in a way that leads to unexpected kernel behavior.
- When dealing with a regression, make sure it's not something else that changed in parallel to updating the kernel. The problem for example might be caused by other software that was updated at the same time. It can also happen that a hardware component coincidentally just broke when you rebooted into a new kernel for the first time. Updating the systems BIOS or changing something in the BIOS Setup can also lead to problems that on look a lot like a kernel regression.

Prepare for emergencies

> *Create a fresh backup and put system repair and restore tools at hand.*

Reminder, you are dealing with computers, which sometimes do unexpected things, especially if you fiddle with crucial parts like the kernel of its operating system. That's what you are about to do in this process. Thus, make sure to create a fresh backup; also ensure you have all tools at hand to repair or reinstall the operating system as well as everything you need to restore the backup.

Make sure your kernel doesn't get enhanced

> *Ensure your system does not enhance its kernels by building additional kernel modules on-the-fly, which solutions like DKMS might be doing locally without your knowledge.*

The risk your issue report gets ignored or rejected dramatically increases if your kernel gets enhanced in any way. That's why you should remove or disable mechanisms like akmods and DKMS: those build add-on kernel modules automatically, for example when you install a new Linux kernel or boot it for the first time. Also remove any modules they might have installed. Then reboot before proceeding.

Note, you might not be aware that your system is using one of these solutions: they often get set up silently when you install Nvidia's proprietary graphics driver, VirtualBox, or other software that requires a some support from a module not part of the Linux kernel. That why your might need to uninstall the packages with such software to get rid of any 3rd party kernel module.

Check 'taint' flag

> *Check if your kernel was 'tainted' when the issue occurred, as the event that made the kernel set this flag might be causing the issue you face.*

The kernel marks itself with a 'taint' flag when something happens that might lead to follow-up errors that look totally unrelated. The issue you face might be such an error if your kernel is tainted. That's why it's in your interest to rule this out early before investing more time into this process. This is the only reason why this step is here, as this process later will tell you to install the latest mainline kernel; you will need to check the taint flag again then, as that's when it matters because it's the kernel the report will focus on.

On a running system is easy to check if the kernel tainted itself: if `cat /proc/sys/kernel/tainted` returns '0' then the kernel is not tainted and everything is fine. Checking that file is impossible in some situations; that's why the kernel also mentions the taint status when it reports an internal problem (a 'kernel bug'), a recoverable error (a 'kernel Oops') or a non-recoverable error before halting operation (a 'kernel panic'). Look near the top of the error messages printed when one of these occurs and search for a line starting with 'CPU:'. It should end with 'Not tainted' if the kernel was not tainted when it noticed the problem; it was tainted if you see 'Tainted:' followed by a few spaces and some letters.

If your kernel is tainted, study Documentation/admin-guide/tainted-kernels.rst to find out why. Try to eliminate the reason. Often it's caused by one these three things:

1. A recoverable error (a 'kernel Oops') occurred and the kernel tainted itself, as the kernel knows it might misbehave in strange ways after that point. In that case check your kernel or system log and look for a section that starts with this:

   ```
   Oops: 0000 [#1] SMP
   ```

   That's the first Oops since boot-up, as the '#1' between the brackets shows. Every Oops and any other problem that happens after that point might be a follow-up problem to that first Oops, even if both look totally unrelated. Rule this out by getting rid of the cause for the first Oops and reproducing the issue afterwards. Sometimes simply restarting will be enough, sometimes a change to the configuration followed by a reboot can eliminate the Oops. But don't invest too much time into this at this point of the process, as the cause for the Oops might already be fixed in the newer Linux kernel version you are going to install later in this process.
2. Your system uses a software that installs its own kernel modules, for example Nvidia's proprietary graphics driver or VirtualBox. The kernel taints itself when it loads such module from external sources (even if they are Open Source): they sometimes cause errors in unrelated kernel areas and thus might be causing the issue you face. You therefore have to prevent those modules from loading when you want to report an issue to the Linux kernel developers. Most of the time the easiest way to do that is: temporarily uninstall such software including any modules they might have installed. Afterwards reboot.
3. The kernel also taints itself when it's loading a module that resides in the staging tree of the Linux kernel source. That's a special area for code (mostly drivers) that does not yet fulfill the normal Linux kernel quality standards. When you report an issue with such a module it's obviously okay if the kernel is tainted; just make sure the module in question is the only reason for the taint. If the issue happens in an unrelated area reboot and temporarily block the module from being loaded by specifying `foo.blacklist=1` as kernel parameter (replace 'foo' with the name of the module in question).

Document how to reproduce issue

> *Write down coarsely how to reproduce the issue. If you deal with multiple issues at once, create separate notes for each of them and make sure they work independently on a freshly booted system. That's needed, as each issue needs to get reported to the kernel developers separately, unless they are strongly entangled.*

If you deal with multiple issues at once, you'll have to report each of them separately, as they might be handled by different developers. Describing various issues in one report also makes it quite difficult for others to tear it apart. Hence, only combine issues in one report if they are very strongly entangled.

Additionally, during the reporting process you will have to test if the issue happens with other kernel versions. Therefore, it will make your work easier if you know exactly how to reproduce an issue quickly on a freshly booted system.

Note: it's often fruitless to report issues that only happened once, as they might be caused by a bit flip due to cosmic radiation. That's why you should try to rule that out by reproducing the issue before going further. Feel free to ignore this advice if you are experienced enough to tell a one-time error due to faulty hardware apart from a kernel issue that rarely happens and thus is hard to reproduce.

Regression in stable or longterm kernel?

> *If you are facing a regression within a stable or longterm version line (say something broke when updating from 5.10.4 to 5.10.5), scroll down to 'Dealing with regressions within a stable and longterm kernel line'.*

Regression within a stable and longterm kernel version line are something the Linux developers want to fix badly, as such issues are even more unwanted than regression in the main development branch, as they can quickly affect a lot of people. The developers thus want to learn about such issues as quickly as possible, hence there is a streamlined process to report them. Note, regressions with newer kernel version line (say something broke when switching from 5.9.15 to 5.10.5) do not qualify.

Check where you need to report your issue

Locate the driver or kernel subsystem that seems to be causing the issue. Find out how and where its developers expect reports. Note: most of the time this won't be bugzilla.kernel.org, as issues typically need to be sent by mail to a maintainer and a public mailing list.

It's crucial to send your report to the right people, as the Linux kernel is a big project and most of its developers are only familiar with a small subset of it. Quite a few programmers for example only care for just one driver, for example one for a WiFi chip; its developer likely will only have small or no knowledge about the internals of remote or unrelated "subsystems", like the TCP stack, the PCIe/PCI subsystem, memory management or file systems.

Problem is: the Linux kernel lacks a central bug tracker where you can simply file your issue and make it reach the developers that need to know about it. That's why you have to find the right place and way to report issues yourself. You can do that with the help of a script (see below), but it mainly targets kernel developers and experts. For everybody else the MAINTAINERS file is the better place.

How to read the MAINTAINERS file

To illustrate how to use the :ref:`MAINTAINERS <maintainers>` file, lets assume the WiFi in your Laptop suddenly misbehaves after updating the kernel. In that case it's likely an issue in the WiFi driver. Obviously it could also be some code it builds upon, but unless you suspect something like that stick to the driver. If it's really something else, the driver's developers will get the right people involved.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\(linux-master) (Documentation) (admin-guide)reporting-issues.rst`, line 614); *backlink*

Unknown interpreted text role "ref".

Sadly, there is no way to check which code is driving a particular hardware component that is both universal and easy.

In case of a problem with the WiFi driver you for example might want to look at the output of `lspci -k`, as it lists devices on the PCI/PCIe bus and the kernel module driving it:

```
[user@something ~]$ lspci -k
[...]
3a:00.0 Network controller: Qualcomm Atheros QCA6174 802.11ac Wireless Network Adapter (rev 32)
  Subsystem: Bigfoot Networks, Inc. Device 1535
  Kernel driver in use: ath10k_pci
  Kernel modules: ath10k_pci
[...]
```

But this approach won't work if your WiFi chip is connected over USB or some other internal bus. In those cases you might want to check your WiFi manager or the output of `ip link`. Look for the name of the problematic network interface, which might be something like 'wlp58s0'. This name can be used like this to find the module driving it:

```
[user@something ~]$ realpath --relative-to=/sys/module/ /sys/class/net/wlp58s0/device/driver/module
ath10k_pci
```

In case tricks like these don't bring you any further, try to search the internet on how to narrow down the driver or subsystem in question. And if you are unsure which it is: just try your best guess, somebody will help you if you guessed poorly.

Once you know the driver or subsystem, you want to search for it in the MAINTAINERS file. In the case of 'ath10k_pci' you won't find anything, as the name is too specific. Sometimes you will need to search on the net for help; but before doing so, try a somewhat shorted or modified name when searching the MAINTAINERS file, as then you might find something like this:

```
QUALCOMM ATHEROS ATH10K WIRELESS DRIVER
Mail:          A. Some Human <shuman@example.com>
Mailing list:  ath10k@lists.infradead.org
Status:        Supported
Web-page:      https://wireless.wiki.kernel.org/en/users/Drivers/ath10k
SCM:           git git://git.kernel.org/pub/scm/linux/kernel/git/kvalo/ath.git
Files:         drivers/net/wireless/ath/ath10k/
```

Note: the line description will be abbreviations, if you read the plain MAINTAINERS file found in the root of the Linux source tree. 'Mail:' for example will be 'M:', 'Mailing list:' will be 'L', and 'Status:' will be 'S:'. A section near the top of the file explains these and other abbreviations.

First look at the line 'Status'. Ideally it should be 'Supported' or 'Maintained'. If it states 'Obsolete' then you are using some outdated approach that was replaced by a newer solution you need to switch to. Sometimes the code only has someone who provides 'Odd Fixes' when feeling motivated. And with 'Orphan' you are totally out of luck, as nobody takes care of the code anymore. That only leaves these options: arrange yourself to live with the issue, fix it yourself, or find a programmer somewhere willing to fix it.

After checking the status, look for a line starting with 'bugs:': it will tell you where to find a subsystem specific bug tracker to file your issue. The example above does not have such a line. That is the case for most sections, as Linux kernel development is completely driven by mail. Very few subsystems use a bug tracker, and only some of those rely on bugzilla.kernel.org.

In this and many other cases you thus have to look for lines starting with 'Mail:' instead. Those mention the name and the email addresses for the maintainers of the particular code. Also look for a line starting with 'Mailing list:', which tells you the public mailing list where the code is developed. Your report later needs to go by mail to those addresses. Additionally, for all issue reports sent by email, make sure to add the Linux Kernel Mailing List (LKML) <linux-kernel@vger.kernel.org> to CC. Don't omit either of the mailing lists when sending your issue report by mail later! Maintainers are busy people and might leave some work for other developers on the subsystem specific list; and LKML is important to have one place where all issue reports can be found.

Finding the maintainers with the help of a script

For people that have the Linux sources at hand there is a second option to find the proper place to report: the script 'scripts/get_maintainer.pl' which tries to find all people to contact. It queries the MAINTAINERS file and needs to be called with a path to the source code in question. For drivers compiled as module if often can be found with a command like this:

```
$ modinfo ath10k_pci | grep filename | sed 's!/lib/modules/.*/kernel/!!; s!filename:!!; s!\.ko\(\|\.xz\)!!'
drivers/net/wireless/ath/ath10k/ath10k_pci.ko
```

Pass parts of this to the script:

```
$ ./scripts/get_maintainer.pl -f drivers/net/wireless/ath/ath10k*
Some Human <shuman@example.com> (supporter:QUALCOMM ATHEROS ATH10K WIRELESS DRIVER)
Another S. Human <asomehuman@example.com> (maintainer:NETWORKING DRIVERS)
ath10k@lists.infradead.org (open list:QUALCOMM ATHEROS ATH10K WIRELESS DRIVER)
linux-wireless@vger.kernel.org (open list:NETWORKING DRIVERS (WIRELESS))
netdev@vger.kernel.org (open list:NETWORKING DRIVERS)
linux-kernel@vger.kernel.org (open list)
```

Don't sent your report to all of them. Send it to the maintainers, which the script calls "supporter:"; additionally CC the most specific mailing list for the code as well as the Linux Kernel Mailing List (LKML). In this case you thus would need to send the report to 'Some Human <shuman@example.com>' with 'ath10k@lists.infradead.org' and 'linux-kernel@vger.kernel.org' in CC.

Note: in case you cloned the Linux sources with git you might want to call `get_maintainer.pl` a second time with `--git`. The script then will look at the commit history to find which people recently worked on the code in question, as they might be able to help. But use these results with care, as it can easily send you in a wrong direction. That for example happens quickly in areas rarely changed (like old or unmaintained drivers): sometimes such code is modified during tree-wide cleanups by developers that do not care about the particular driver at all.

Search for existing reports, second run

> *Search the archives of the bug tracker or mailing list in question thoroughly for reports that might match your issue. If you find anything, join the discussion instead of sending a new report.*

As mentioned earlier already: reporting an issue that someone else already brought forward is often a waste of time for everyone involved, especially you as the reporter. That's why you should search for existing report again, now that you know where they need to be reported to. If it's mailing list, you will often find its archives on lore.kernel.org.

But some list are hosted in different places. That for example is the case for the ath10k WiFi driver used as example in the previous step. But you'll often find the archives for these lists easily on the net. Searching for 'archive ath10k@lists.infradead.org' for example will lead you to the Info page for the ath10k mailing list, which at the top links to its list archives. Sadly this and quite a few other lists miss a way to search the archives. In those cases use a regular internet search engine and add something like 'site:lists.infradead.org/pipermail/ath10k/' to your search terms, which limits the results to the archives at that URL.

It's also wise to check the internet, LKML and maybe bugzilla.kernel.org again at this point. If your report needs to be filed in a bug tracker, you may want to check the mailing list archives for the subsystem as well, as someone might have reported it only there.

For details how to search and what to do if you find matching reports see "Search for existing reports, first run" above.

Do not hurry with this step of the reporting process: spending 30 to 60 minutes or even more time can save you and others quite a lot of time and trouble.

Install a fresh kernel for testing

> *Unless you are already running the latest 'mainline' Linux kernel, better go and install it for the reporting process. Testing and reporting with the latest 'stable' Linux can be an acceptable alternative in some situations; during the merge window that actually might be even the best approach, but in that development phase it can be an even better idea to suspend your efforts for a few days anyway. Whatever version you choose, ideally use a 'vanilla' built. Ignoring these advices will dramatically increase the risk your report will be rejected or ignored.*

As mentioned in the detailed explanation for the first step already: Like most programmers, Linux kernel developers don't like to spend time dealing with reports for issues that don't even happen with the current code. It's just a waste everybody's time, especially yours. That's why it's in everybody's interest that you confirm the issue still exists with the latest upstream code before reporting it. You are free to ignore this advice, but as outlined earlier: doing so dramatically increases the risk that your issue report might get rejected or simply ignored.

In the scope of the kernel "latest upstream" normally means:

- Install a mainline kernel; the latest stable kernel can be an option, but most of the time is better avoided. Longterm kernels (sometimes called 'LTS kernels') are unsuitable at this point of the process. The next subsection explains all of this in more detail.
- The over next subsection describes way to obtain and install such a kernel. It also outlines that using a pre-compiled

kernel are fine, but better are vanilla, which means: it was built using Linux sources taken straight from kernel.org and not modified or enhanced in any way.

Choosing the right version for testing

Head over to kernel.org to find out which version you want to use for testing. Ignore the big yellow button that says 'Latest release' and look a little lower at the table. At its top you'll see a line starting with mainline, which most of the time will point to a pre-release with a version number like '5.8-rc2'. If that's the case, you'll want to use this mainline kernel for testing, as that where all fixes have to be applied first. Do not let that 'rc' scare you, these 'development kernels' are pretty reliable â€" and you made a backup, as you were instructed above, didn't you?

In about two out of every nine to ten weeks, mainline might point you to a proper release with a version number like '5.7'. If that happens, consider suspending the reporting process until the first pre-release of the next version (5.8-rc1) shows up on kernel.org. That's because the Linux development cycle then is in its two-week long 'merge window'. The bulk of the changes and all intrusive ones get merged for the next release during this time. It's a bit more risky to use mainline during this period. Kernel developers are also often quite busy then and might have no spare time to deal with issue reports. It's also quite possible that one of the many changes applied during the merge window fixes the issue you face; that's why you soon would have to retest with a newer kernel version anyway, as outlined below in the section 'Duties after the report went out'.

That's why it might make sense to wait till the merge window is over. But don't to that if you're dealing with something that shouldn't wait. In that case consider obtaining the latest mainline kernel via git (see below) or use the latest stable version offered on kernel.org. Using that is also acceptable in case mainline for some reason does currently not work for you. An in general: using it for reproducing the issue is also better than not reporting it issue at all.

Better avoid using the latest stable kernel outside merge windows, as all fixes must be applied to mainline first. That's why checking the latest mainline kernel is so important: any issue you want to see fixed in older version lines needs to be fixed in mainline first before it can get backported, which can take a few days or weeks. Another reason: the fix you hope for might be too hard or risky for backporting; reporting the issue again hence is unlikely to change anything.

These aspects are also why longterm kernels (sometimes called "LTS kernels") are unsuitable for this part of the reporting process: they are to distant from the current code. Hence go and test mainline first and follow the process further: if the issue doesn't occur with mainline it will guide you how to get it fixed in older version lines, if that's in the cards for the fix in question.

How to obtain a fresh Linux kernel

Using a pre-compiled kernel: This is often the quickest, easiest, and safest way for testing â€" especially is you are unfamiliar with the Linux kernel. The problem: most of those shipped by distributors or add-on repositories are build from modified Linux sources. They are thus not vanilla and therefore often unsuitable for testing and issue reporting: the changes might cause the issue you face or influence it somehow.

But you are in luck if you are using a popular Linux distribution: for quite a few of them you'll find repositories on the net that contain packages with the latest mainline or stable Linux built as vanilla kernel. It's totally okay to use these, just make sure from the repository's description they are vanilla or at least close to it. Additionally ensure the packages contain the latest versions as offered on kernel.org. The packages are likely unsuitable if they are older than a week, as new mainline and stable kernels typically get released at least once a week.

Please note that you might need to build your own kernel manually later: that's sometimes needed for debugging or testing fixes, as described later in this document. Also be aware that pre-compiled kernels might lack debug symbols that are needed to decode messages the kernel prints when a panic, Oops, warning, or BUG occurs; if you plan to decode those, you might be better off compiling a kernel yourself (see the end of this subsection and the section titled 'Decode failure messages' for details).

Using git: Developers and experienced Linux users familiar with git are often best served by obtaining the latest Linux kernel sources straight from the official development repository on kernel.org. Those are likely a bit ahead of the latest mainline pre-release. Don't worry about it: they are as reliable as a proper pre-release, unless the kernel's development cycle is currently in the middle of a merge window. But even then they are quite reliable.

Conventional: People unfamiliar with git are often best served by downloading the sources as tarball from kernel.org.

How to actually build a kernel is not described here, as many websites explain the necessary steps already. If you are new to it, consider following one of those how-to's that suggest to use `make localmodconfig`, as that tries to pick up the configuration of your current kernel and then tries to adjust it somewhat for your system. That does not make the resulting kernel any better, but quicker to compile.

Note: If you are dealing with a panic, Oops, warning, or BUG from the kernel, please try to enable CONFIG_KALLSYMS when configuring your kernel. Additionally, enable CONFIG_DEBUG_KERNEL and CONFIG_DEBUG_INFO, too; the latter is the relevant one of those two, but can only be reached if you enable the former. Be aware CONFIG_DEBUG_INFO increases the storage space required to build a kernel by quite a bit. But that's worth it, as these options will allow you later to pinpoint the exact line of code that triggers your issue. The section 'Decode failure messages' below explains this in more detail.

But keep in mind: Always keep a record of the issue encountered in case it is hard to reproduce. Sending an undecoded report is better than not reporting the issue at all.

Check 'taint' flag

Ensure the kernel you just installed does not 'taint' itself when running.

As outlined above in more detail already: the kernel sets a 'taint' flag when something happens that can lead to follow-up errors that look totally unrelated. That's why you need to check if the kernel you just installed does not set this flag. And if it does, you in almost all the cases needs to eliminate the reason for it before you reporting issues that occur with it. See the section above for details how to do that.

Reproduce issue with the fresh kernel

> *Reproduce the issue with the kernel you just installed. If it doesn't show up there, scroll down to the instructions for issues only happening with stable and longterm kernels.*

Check if the issue occurs with the fresh Linux kernel version you just installed. If it was fixed there already, consider sticking with this version line and abandoning your plan to report the issue. But keep in mind that other users might still be plagued by it, as long as it's not fixed in either stable and longterm version from kernel.org (and thus vendor kernels derived from those). If you prefer to use one of those or just want to help their users, head over to the section "Details about reporting issues only occurring in older kernel version lines" below.

Optimize description to reproduce issue

> *Optimize your notes: try to find and write the most straightforward way to reproduce your issue. Make sure the end result has all the important details, and at the same time is easy to read and understand for others that hear about it for the first time. And if you learned something in this process, consider searching again for existing reports about the issue.*

An unnecessarily complex report will make it hard for others to understand your report. Thus try to find a reproducer that's straight forward to describe and thus easy to understand in written form. Include all important details, but at the same time try to keep it as short as possible.

In this in the previous steps you likely have learned a thing or two about the issue you face. Use this knowledge and search again for existing reports instead you can join.

Decode failure messages

> *If your failure involves a 'panic', 'Oops', 'warning', or 'BUG', consider decoding the kernel log to find the line of code that triggered the error.*

When the kernel detects an internal problem, it will log some information about the executed code. This makes it possible to pinpoint the exact line in the source code that triggered the issue and shows how it was called. But that only works if you enabled CONFIG_DEBUG_INFO and CONFIG_KALLSYMS when configuring your kernel. If you did so, consider to decode the information from the kernel's log. That will make it a lot easier to understand what lead to the 'panic', 'Oops', 'warning', or 'BUG', which increases the chances that someone can provide a fix.

Decoding can be done with a script you find in the Linux source tree. If you are running a kernel you compiled yourself earlier, call it like this:

```
[user@something ~]$ sudo dmesg | ./linux-5.10.5/scripts/decode_stacktrace.sh ./linux-5.10.5/vmlinux
```

If you are running a packaged vanilla kernel, you will likely have to install the corresponding packages with debug symbols. Then call the script (which you might need to get from the Linux sources if your distro does not package it) like this:

```
[user@something ~]$ sudo dmesg | ./linux-5.10.5/scripts/decode_stacktrace.sh \
 /usr/lib/debug/lib/modules/5.10.10-4.1.x86_64/vmlinux /usr/src/kernels/5.10.10-4.1.x86_64/
```

The script will work on log lines like the following, which show the address of the code the kernel was executing when the error occurred:

```
[   68.387301] RIP: 0010:test_module_init+0x5/0xffa [test_module]
```

Once decoded, these lines will look like this:

```
[   68.387301] RIP: 0010:test_module_init (/home/username/linux-5.10.5/test-module/test-module.c:16) test_mo
```

In this case the executed code was built from the file '~/linux-5.10.5/test-module/test-module.c' and the error occurred by the instructions found in line '16'.

The script will similarly decode the addresses mentioned in the section starting with 'Call trace', which show the path to the function where the problem occurred. Additionally, the script will show the assembler output for the code section the kernel was executing.

Note, if you can't get this to work, simply skip this step and mention the reason for it in the report. If you're lucky, it might not be needed. And if it is, someone might help you to get things going. Also be aware this is just one of several ways to decode kernel stack traces. Sometimes different steps will be required to retrieve the relevant details. Don't worry about that, if that's needed in your case, developers will tell you what to do.

Special care for regressions

> *If your problem is a regression, try to narrow down when the issue was introduced as much as possible.*

Linux lead developer Linus Torvalds insists that the Linux kernel never worsens, that's why he deems regressions as unacceptable and wants to see them fixed quickly. That's why changes that introduced a regression are often promptly reverted if the issue they cause can't get solved quickly any other way. Reporting a regression is thus a bit like playing a kind of trump card to get something quickly fixed. But for that to happen the change that's causing the regression needs to be known. Normally it's up to the reporter to track down the culprit, as maintainers often won't have the time or setup at hand to reproduce it themselves.

To find the change there is a process called 'bisection' which the document Documentation/admin-guide/bug-bisect.rst describes in detail. That process will often require you to build about ten to twenty kernel images, trying to reproduce the issue with each of them before building the next. Yes, that takes some time, but don't worry, it works a lot quicker than most people assume. Thanks to a 'binary search' this will lead you to the one commit in the source code management system that's causing the regression. Once you find it, search the net for the subject of the change, its commit id and the shortened commit id (the first 12 characters of the commit id). This will lead you to existing reports about it, if there are any.

Note, a bisection needs a bit of know-how, which not everyone has, and quite a bit of effort, which not everyone is willing to invest. Nevertheless, it's highly recommended performing a bisection yourself. If you really can't or don't want to go down that route at least find out which mainline kernel introduced the regression. If something for example breaks when switching from 5.5.15 to 5.8.4, then try at least all the mainline releases in that area (5.6, 5.7 and 5.8) to check when it first showed up. Unless you're trying to find a regression in a stable or longterm kernel, avoid testing versions which number has three sections (5.6.12, 5.7.8), as that makes the outcome hard to interpret, which might render your testing useless. Once you found the major version which introduced the regression, feel free to move on in the reporting process. But keep in mind: it depends on the issue at hand if the developers will be able to help without knowing the culprit. Sometimes they might recognize from the report want went wrong and can fix it; other times they will be unable to help unless you perform a bisection.

When dealing with regressions make sure the issue you face is really caused by the kernel and not by something else, as outlined above already.

In the whole process keep in mind: an issue only qualifies as regression if the older and the newer kernel got built with a similar configuration. This can be achieved by using `make olddefconfig`, as explained in more detail by Documentation/admin-guide/reporting-regressions.rst; that document also provides a good deal of other information about regressions you might want to be aware of.

Write and send the report

> *Start to compile the report by writing a detailed description about the issue. Always mention a few things: the latest kernel version you installed for reproducing, the Linux Distribution used, and your notes on how to reproduce the issue. Ideally, make the kernel's build configuration (.config) and the output from ``dmesg`` available somewhere on the net and link to it. Include or upload all other information that might be relevant, like the output/screenshot of an Oops or the output from ``lspci``. Once you wrote this main part, insert a normal length paragraph on top of it outlining the issue and the impact quickly. On top of this add one sentence that briefly describes the problem and gets people to read on. Now give the thing a descriptive title or subject that yet again is shorter. Then you're ready to send or file the report like the MAINTAINERS file told you, unless you are dealing with one of those 'issues of high priority': they need special care which is explained in 'Special handling for high priority issues' below.*

Now that you have prepared everything it's time to write your report. How to do that is partly explained by the three documents linked to in the preface above. That's why this text will only mention a few of the essentials as well as things specific to the Linux kernel.

There is one thing that fits both categories: the most crucial parts of your report are the title/subject, the first sentence, and the first paragraph. Developers often get quite a lot of mail. They thus often just take a few seconds to skim a mail before deciding to move on or look closer. Thus: the better the top section of your report, the higher are the chances that someone will look into it and help you. And that is why you should ignore them for now and write the detailed report first. ;-)

Things each report should mention

Describe in detail how your issue happens with the fresh vanilla kernel you installed. Try to include the step-by-step instructions you wrote and optimized earlier that outline how you and ideally others can reproduce the issue; in those rare cases where that's impossible try to describe what you did to trigger it.

Also include all the relevant information others might need to understand the issue and its environment. What's actually needed depends a lot on the issue, but there are some things you should include always:

- the output from `cat /proc/version`, which contains the Linux kernel version number and the compiler it was built with.
- the Linux distribution the machine is running (`hostnamectl | grep "Operating System"`)
- the architecture of the CPU and the operating system (`uname -mi`)
- if you are dealing with a regression and performed a bisection, mention the subject and the commit-id of the change that is causing it.

In a lot of cases it's also wise to make two more things available to those that read your report:

- the configuration used for building your Linux kernel (the '.config' file)
- the kernel's messages that you get from `dmesg` written to a file. Make sure that it starts with a line like 'Linux version 5.8-1 (foobar@example.com) (gcc (GCC) 10.2.1, GNU ld version 2.34) #1 SMP Mon Aug 3 14:54:37 UTC 2020' If it's missing, then important messages from the first boot phase already got discarded. In this case instead consider using `journalctl -b 0 -k`; alternatively you can also reboot, reproduce the issue and call `dmesg` right afterwards.

These two files are big, that's why it's a bad idea to put them directly into your report. If you are filing the issue in a bug tracker then attach them to the ticket. If you report the issue by mail do not attach them, as that makes the mail too large; instead do one of these things:

- Upload the files somewhere public (your website, a public file paste service, a ticket created just for this purpose on bugzilla.kernel.org, ...) and include a link to them in your report. Ideally use something where the files stay available for years, as they could be useful to someone many years from now; this for example can happen if five or ten years from now a developer works on some code that was changed just to fix your issue.
- Put the files aside and mention you will send them later in individual replies to your own mail. Just remember to actually do that once the report went out. ;-)

Things that might be wise to provide

Depending on the issue you might need to add more background data. Here are a few suggestions what often is good to provide:

- If you are dealing with a 'warning', an 'OOPS' or a 'panic' from the kernel, include it. If you can't copy'n'paste it, try to capture a netconsole trace or at least take a picture of the screen.
- If the issue might be related to your computer hardware, mention what kind of system you use. If you for example have problems with your graphics card, mention its manufacturer, the card's model, and what chip is uses. If it's a laptop mention its name, but try to make sure it's meaningful. 'Dell XPS 13' for example is not, because it might be the one from 2012; that one looks not that different from the one sold today, but apart from that the two have nothing in common. Hence, in such cases add the exact model number, which for example are '9380' or '7390' for XPS 13 models introduced during 2019. Names like 'Lenovo Thinkpad T590' are also somewhat ambiguous: there are variants of this laptop with and without a dedicated graphics chip, so try to find the exact model name or specify the main components.
- Mention the relevant software in use. If you have problems with loading modules, you want to mention the versions of kmod, systemd, and udev in use. If one of the DRM drivers misbehaves, you want to state the versions of libdrm and Mesa; also specify your Wayland compositor or the X-Server and its driver. If you have a filesystem issue, mention the version of corresponding filesystem utilities (e2fsprogs, btrfs-progs, xfsprogs, ...).
- Gather additional information from the kernel that might be of interest. The output from `lspci -nn` will for example help others to identify what hardware you use. If you have a problem with hardware you even might want to make the output from `sudo lspci -vvv` available, as that provides insights how the components were configured. For some issues it might be good to include the contents of files like `/proc/cpuinfo`, `/proc/ioports`, `/proc/iomem`, `/proc/modules`, or `/proc/scsi/scsi`. Some subsystem also offer tools to collect relevant information. One such tool is `alsa-info.sh` which the audio/sound subsystem developers provide.

Those examples should give your some ideas of what data might be wise to attach, but you have to think yourself what will be helpful for others to know. Don't worry too much about forgetting something, as developers will ask for additional details they need. But making everything important available from the start increases the chance someone will take a closer look.

The important part: the head of your report

Now that you have the detailed part of the report prepared let's get to the most important section: the first few sentences. Thus go to the top, add something like 'The detailed description:' before the part you just wrote and insert two newlines at the top. Now write one normal length paragraph that describes the issue roughly. Leave out all boring details and focus on the crucial parts readers need to know to understand what this is all about; if you think this bug affects a lot of users, mention this to get people interested.

Once you did that insert two more lines at the top and write a one sentence summary that explains quickly what the report is about. After that you have to get even more abstract and write an even shorter subject/title for the report.

Now that you have written this part take some time to optimize it, as it is the most important parts of your report: a lot of people will only read this before they decide if reading the rest is time well spent.

Now send or file the report like the :ref:`MAINTAINERS <maintainers>` file told you, unless it's one of those 'issues of high priority' outlined earlier: in that case please read the next subsection first before sending the report on its way.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\(linux-master)(Documentation)(admin-guide)reporting-issues.rst`, line 1220); *backlink*

Unknown interpreted text role "ref".

Special handling for high priority issues

Reports for high priority issues need special handling.

Severe issues: make sure the subject or ticket title as well as the first paragraph makes the severeness obvious.

Regressions: make the report's subject start with '[REGRESSION]'.

In case you performed a successful bisection, use the title of the change that introduced the regression as the second part of your subject. Make the report also mention the commit id of the culprit. In case of an unsuccessful bisection, make your report mention the latest tested version that's working fine (say 5.7) and the oldest where the issue occurs (say 5.8-rc1).

When sending the report by mail, CC the Linux regressions mailing list (regressions@lists.linux.dev). In case the report needs to be filed to some web tracker, proceed to do so. Once filed, forward the report by mail to the regressions list; CC the maintainer and the mailing list for the subsystem in question. Make sure to inline the forwarded report, hence do not attach it. Also add a short note at the top where you mention the URL to the ticket.

When mailing or forwarding the report, in case of a successful bisection add the author of the culprit to the recipients; also CC everyone in the signed-off-by chain, which you find at the end of its commit message.

Security issues: for these issues your will have to evaluate if a short-term risk to other users would arise if details were publicly disclosed. If that's not the case simply proceed with reporting the issue as described. For issues that bear such a risk you will need to adjust the reporting process slightly:

- If the MAINTAINERS file instructed you to report the issue by mail, do not CC any public mailing lists.
- If you were supposed to file the issue in a bug tracker make sure to mark the ticket as 'private' or 'security issue'. If the bug tracker does not offer a way to keep reports private, forget about it and send your report as a private mail to the maintainers instead.

In both cases make sure to also mail your report to the addresses the MAINTAINERS file lists in the section 'security contact'. Ideally directly CC them when sending the report by mail. If you filed it in a bug tracker, forward the report's text to these addresses; but on top of it put a small note where you mention that you filed it with a link to the ticket.

See Documentation/admin-guide/security-bugs.rst for more information.

Duties after the report went out

Wait for reactions and keep the thing rolling until you can accept the outcome in one way or the other. Thus react publicly and in a timely manner to any inquiries. Test proposed fixes. Do proactive testing: retest with at least every first release candidate (RC) of a new mainline version and report your results. Send friendly reminders if things stall. And try to help yourself, if you don't get any help or if it's unsatisfying.

If your report was good and you are really lucky then one of the developers might immediately spot what's causing the issue; they then might write a patch to fix it, test it, and send it straight for integration in mainline while tagging it for later backport to stable and longterm kernels that need it. Then all you need to do is reply with a 'Thank you very much' and switch to a version with the fix once it gets released.

But this ideal scenario rarely happens. That's why the job is only starting once you got the report out. What you'll have to do depends on the situations, but often it will be the things listed below. But before digging into the details, here are a few important things you need to keep in mind for this part of the process.

General advice for further interactions

Always reply in public: When you filed the issue in a bug tracker, always reply there and do not contact any of the developers privately about it. For mailed reports always use the 'Reply-all' function when replying to any mails you receive. That includes mails with any additional data you might want to add to your report: go to your mail applications 'Sent' folder and use 'reply-all' on your mail with the report. This approach will make sure the public mailing list(s) and everyone else that gets involved over time stays in the loop; it also keeps the mail thread intact, which among others is really important for mailing lists to group all related mails together.

There are just two situations where a comment in a bug tracker or a 'Reply-all' is unsuitable:

- Someone tells you to send something privately.
- You were told to send something, but noticed it contains sensitive information that needs to be kept private. In that case it's okay to send it in private to the developer that asked for it. But note in the ticket or a mail that you did that, so everyone else knows you honored the request.

Do research before asking for clarifications or help: In this part of the process someone might tell you to do something that requires a skill you might not have mastered yet. For example, you might be asked to use some test tools you never have heard of yet; or you might be asked to apply a patch to the Linux kernel sources to test if it helps. In some cases it will be fine sending a reply asking for instructions how to do that. But before going that route try to find the answer own your own by searching the internet; alternatively consider asking in other places for advice. For example ask a friend or post about it to a chatroom or forum you normally hang out.

Be patient: If you are really lucky you might get a reply to your report within a few hours. But most of the time it will take longer, as maintainers are scattered around the globe and thus might be in a different time zone â€“ one where they already enjoy their night away from keyboard.

In general, kernel developers will take one to five business days to respond to reports. Sometimes it will take longer, as they might be busy with the merge windows, other work, visiting developer conferences, or simply enjoying a long summer holiday.

The 'issues of high priority' (see above for an explanation) are an exception here: maintainers should address them as soon as possible; that's why you should wait a week at maximum (or just two days if it's something urgent) before sending a friendly reminder.

Sometimes the maintainer might not be responding in a timely manner; other times there might be disagreements, for example if an issue qualifies as regression or not. In such cases raise your concerns on the mailing list and ask others for public or private replies how to move on. If that fails, it might be appropriate to get a higher authority involved. In case of a WiFi driver that would be the wireless maintainers; if there are no higher level maintainers or all else fails, it might be one of those rare situations where it's okay to get Linus Torvalds involved.

Proactive testing: Every time the first pre-release (the 'rc1') of a new mainline kernel version gets released, go and check if the issue is fixed there or if anything of importance changed. Mention the outcome in the ticket or in a mail you sent as reply to your report (make sure it has all those in the CC that up to that point participated in the discussion). This will show your commitment and that you are willing to help. It also tells developers if the issue persists and makes sure they do not forget about it. A few other occasional retests (for example with rc3, rc5 and the final) are also a good idea, but only report your results if something relevant changed or if you are writing something anyway.

With all these general things off the table let's get into the details of how to help to get issues resolved once they were reported.

Inquires and testing request

Here are your duties in case you got replies to your report:

Check who you deal with: Most of the time it will be the maintainer or a developer of the particular code area that will respond to your report. But as issues are normally reported in public it could be anyone that's replying â€" including people that want to help, but in the end might guide you totally off track with their questions or requests. That rarely happens, but it's one of many reasons why it's wise to quickly run an internet search to see who you're interacting with. By doing this you also get aware if your report was heard by the right people, as a reminder to the maintainer (see below) might be in order later if discussion fades out without leading to a satisfying solution for the issue.

Inquiries for data: Often you will be asked to test something or provide additional details. Try to provide the requested information soon, as you have the attention of someone that might help and risk losing it the longer you wait; that outcome is even likely if you do not provide the information within a few business days.

Requests for testing: When you are asked to test a diagnostic patch or a possible fix, try to test it in timely manner, too. But do it properly and make sure to not rush it: mixing things up can happen easily and can lead to a lot of confusion for everyone involved. A common mistake for example is thinking a proposed patch with a fix was applied, but in fact wasn't. Things like that happen even to experienced testers occasionally, but they most of the time will notice when the kernel with the fix behaves just as one without it.

What to do when nothing of substance happens

Some reports will not get any reaction from the responsible Linux kernel developers; or a discussion around the issue evolved, but faded out with nothing of substance coming out of it.

In these cases wait two (better: three) weeks before sending a friendly reminder: maybe the maintainer was just away from keyboard for a while when your report arrived or had something more important to take care of. When writing the reminder, kindly ask if anything else from your side is needed to get the ball running somehow. If the report got out by mail, do that in the first lines of a mail that is a reply to your initial mail (see above) which includes a full quote of the original report below: that's on of those few situations where such a 'TOFU' (Text Over, Fullquote Under) is the right approach, as then all the recipients will have the details at hand immediately in the proper order.

After the reminder wait three more weeks for replies. If you still don't get a proper reaction, you first should reconsider your approach. Did you maybe try to reach out to the wrong people? Was the report maybe offensive or so confusing that people decided to completely stay away from it? The best way to rule out such factors: show the report to one or two people familiar with FLOSS issue reporting and ask for their opinion. Also ask them for their advice how to move forward. That might mean: prepare a better report and make those people review it before you send it out. Such an approach is totally fine; just mention that this is the second and improved report on the issue and include a link to the first report.

If the report was proper you can send a second reminder; in it ask for advice why the report did not get any replies. A good moment for this second reminder mail is shortly after the first pre-release (the 'rc1') of a new Linux kernel version got published, as you should retest and provide a status update at that point anyway (see above).

If the second reminder again results in no reaction within a week, try to contact a higher-level maintainer asking for advice: even busy maintainers by then should at least have sent some kind of acknowledgment.

Remember to prepare yourself for a disappointment: maintainers ideally should react somehow to every issue report, but they are only obliged to fix those 'issues of high priority' outlined earlier. So don't be too devastating if you get a reply along the lines of 'thanks for the report, I have more important issues to deal with currently and won't have time to look into this for the foreseeable future'.

It's also possible that after some discussion in the bug tracker or on a list nothing happens anymore and reminders don't help to motivate anyone to work out a fix. Such situations can be devastating, but is within the cards when it comes to Linux kernel development. This and several other reasons for not getting help are explained in 'Why some issues won't get any reaction or remain unfixed after being reported' near the end of this document.

Don't get devastated if you don't find any help or if the issue in the end does not get solved: the Linux kernel is FLOSS and thus you can still help yourself. You for example could try to find others that are affected and team up with them to get the issue resolved. Such a team could prepare a fresh report together that mentions how many you are and why this is something that in your option should get fixed. Maybe together you can also narrow down the root cause or the change that introduced a regression, which often makes developing a fix easier. And with a bit of luck there might be someone in the team that knows a bit about programming and might be able to write a fix.

Reference for "Reporting regressions within a stable and longterm kernel line"

This subsection provides details for the steps you need to perform if you face a regression within a stable and longterm kernel line.

Make sure the particular version line still gets support

> *Check if the kernel developers still maintain the Linux kernel version line you care about: go to the front page of kernel.org and make sure it mentions the latest release of the particular version line without an '[EOL]' tag.*

Most kernel version lines only get supported for about three months, as maintaining them longer is quite a lot of work. Hence, only one per year is chosen and gets supported for at least two years (often six). That's why you need to check if the kernel developers still support the version line you care for.

Note, if kernel.org lists two stable version lines on the front page, you should consider switching to the newer one and forget about the older one: support for it is likely to be abandoned soon. Then it will get a "end-of-life" (EOL) stamp. Version lines that reached that point still get mentioned on the kernel.org front page for a week or two, but are unsuitable for testing and reporting.

Search stable mailing list

> *Check the archives of the Linux stable mailing list for existing reports.*

Maybe the issue you face is already known and was fixed or is about to. Hence, search the archives of the Linux stable mailing list for reports about an issue like yours. If you find any matches, consider joining the discussion, unless the fix is already finished and scheduled to get applied soon.

Reproduce issue with the newest release

> *Install the latest release from the particular version line as a vanilla kernel. Ensure this kernel is not tainted and still shows the problem, as the issue might have already been fixed there. If you first noticed the problem with a vendor kernel, check a vanilla build of the last version known to work performs fine as well.*

Before investing any more time in this process you want to check if the issue was already fixed in the latest release of version line you're interested in. This kernel needs to be vanilla and shouldn't be tainted before the issue happens, as detailed outlined already above in the section "Install a fresh kernel for testing".

Did you first notice the regression with a vendor kernel? Then changes the vendor applied might be interfering. You need to rule that out by performing a recheck. Say something broke when you updated from 5.10.4-vendor.42 to 5.10.5-vendor.43. Then after testing the latest 5.10 release as outlined in the previous paragraph check if a vanilla build of Linux 5.10.4 works fine as well. If things are broken there, the issue does not qualify as upstream regression and you need switch back to the main step-by-step guide to report the issue.

Report the regression

> *Send a short problem report to the Linux stable mailing list (stable@vger.kernel.org) and CC the Linux regressions mailing list (regressions@lists.linux.dev); if you suspect the cause in a particular subsystem, CC its maintainer and its mailing list. Roughly describe the issue and ideally explain how to reproduce it. Mention the first version that shows the problem and the last version that's working fine. Then wait for further instructions.*

When reporting a regression that happens within a stable or longterm kernel line (say when updating from 5.10.4 to 5.10.5) a brief report is enough for the start to get the issue reported quickly. Hence a rough description to the stable and regressions mailing list is all it takes; but in case you suspect the cause in a particular subsystem, CC its maintainers and its mailing list as well, because that will speed things up.

And note, it helps developers a great deal if you can specify the exact version that introduced the problem. Hence if possible within a reasonable time frame, try to find that version using vanilla kernels. Lets assume something broke when your distributor released a update from Linux kernel 5.10.5 to 5.10.8. Then as instructed above go and check the latest kernel from that version line, say 5.10.9. If it shows the problem, try a vanilla 5.10.5 to ensure that no patches the distributor applied interfere. If the issue doesn't manifest itself there, try 5.10.7 and then (depending on the outcome) 5.10.8 or 5.10.6 to find the first version where things broke. Mention it in the report and state that 5.10.9 is still broken.

What the previous paragraph outlines is basically a rough manual 'bisection'. Once your report is out your might get asked to do a proper one, as it allows to pinpoint the exact change that causes the issue (which then can easily get reverted to fix the issue quickly). Hence consider to do a proper bisection right away if time permits. See the section 'Special care for regressions' and the document Documentation/admin-guide/bug-bisect.rst for details how to perform one. In case of a successful bisection add the author of the culprit to the recipients; also CC everyone in the signed-off-by chain, which you find at the end of its commit message.

Reference for "Reporting issues only occurring in older kernel version lines"

This section provides details for the steps you need to take if you could not reproduce your issue with a mainline kernel, but want to see it fixed in older version lines (aka stable and longterm kernels).

Some fixes are too complex

> *Prepare yourself for the possibility that going through the next few steps might not get the issue solved in older releases: the fix might be too big or risky to get backported there.*

Even small and seemingly obvious code-changes sometimes introduce new and totally unexpected problems. The maintainers of the stable and longterm kernels are very aware of that and thus only apply changes to these kernels that are within rules outlined in Documentation/process/stable-kernel-rules.rst.

Complex or risky changes for example do not qualify and thus only get applied to mainline. Other fixes are easy to get backported to the newest stable and longterm kernels, but too risky to integrate into older ones. So be aware the fix you are hoping for might be one of those that won't be backported to the version line your care about. In that case you'll have no other choice then to live with the issue or switch to a newer Linux version, unless you want to patch the fix into your kernels yourself.

Common preparations

> *Perform the first three steps in the section "Reporting issues only occurring in older kernel version lines" above.*

You need to carry out a few steps already described in another section of this guide. Those steps will let you:

- Check if the kernel developers still maintain the Linux kernel version line you care about.
- Search the Linux stable mailing list for exiting reports.
- Check with the latest release.

Check code history and search for existing discussions

> *Search the Linux kernel version control system for the change that fixed the issue in mainline, as its commit message might tell you if the fix is scheduled for backporting already. If you don't find anything that way, search the appropriate mailing lists for posts that discuss such an issue or peer-review possible fixes; then check the discussions if the fix was deemed unsuitable for backporting. If backporting was not considered at all, join the newest discussion, asking if it's in the cards.*

In a lot of cases the issue you deal with will have happened with mainline, but got fixed there. The commit that fixed it would need to get backported as well to get the issue solved. That's why you want to search for it or any discussions abound it.

- First try to find the fix in the Git repository that holds the Linux kernel sources. You can do this with the web interfaces on kernel.org or its mirror on GitHub; if you have a local clone you alternatively can search on the command line with `git log --grep=<pattern>`.

  If you find the fix, look if the commit message near the end contains a 'stable tag' that looks like this:

  > Cc: <stable@vger.kernel.org> # 5.4+

  If that's case the developer marked the fix safe for backporting to version line 5.4 and later. Most of the time it's getting applied there within two weeks, but sometimes it takes a bit longer.
- If the commit doesn't tell you anything or if you can't find the fix, look again for discussions about the issue. Search the net with your favorite internet search engine as well as the archives for the Linux kernel developers mailing list. Also read the section *Locate kernel area that causes the issue* above and follow the instructions to find the subsystem in question: its bug tracker or mailing list archive might have the answer you are looking for.
- If you see a proposed fix, search for it in the version control system as outlined above, as the commit might tell you if a backport can be expected.
  - Check the discussions for any indicators the fix might be too risky to get backported to the version line you care about. If that's the case you have to live with the issue or switch to the kernel version line where the fix got applied.
  - If the fix doesn't contain a stable tag and backporting was not discussed, join the discussion: mention the version where you face the issue and that you would like to see it fixed, if suitable.

Ask for advice

> *One of the former steps should lead to a solution. If that doesn't work out, ask the maintainers for the subsystem that seems to be causing the issue for advice; CC the mailing list for the particular subsystem as well as the stable mailing list.*

If the previous three steps didn't get you closer to a solution there is only one option left: ask for advice. Do that in a mail you sent to the maintainers for the subsystem where the issue seems to have its roots; CC the mailing list for the subsystem as well as the stable mailing list (stable@vger.kernel.org).

Why some issues won't get any reaction or remain unfixed after being reported

When reporting a problem to the Linux developers, be aware only 'issues of high priority' (regressions, security issues, severe problems) are definitely going to get resolved. The maintainers or if all else fails Linus Torvalds himself will make sure of that. They and the other kernel developers will fix a lot of other issues as well. But be aware that sometimes they can't or won't help; and sometimes there isn't even anyone to send a report to.

This is best explained with kernel developers that contribute to the Linux kernel in their spare time. Quite a few of the drivers in the kernel were written by such programmers, often because they simply wanted to make their hardware usable on their favorite operating system.

These programmers most of the time will happily fix problems other people report. But nobody can force them to do, as they are contributing voluntarily.

Then there are situations where such developers really want to fix an issue, but can't: sometimes they lack hardware programming documentation to do so. This often happens when the publicly available docs are superficial or the driver was written with the help of reverse engineering.

Sooner or later spare time developers will also stop caring for the driver. Maybe their test hardware broke, got replaced by something more fancy, or is so old that it's something you don't find much outside of computer museums anymore. Sometimes developer stops caring for their code and Linux at all, as something different in their life became way more important. In some cases nobody is willing to take over the job as maintainer â€“ and nobody can be forced to, as contributing to the Linux kernel is done on a voluntary basis. Abandoned drivers nevertheless remain in the kernel: they are still useful for people and removing would be a regression.

The situation is not that different with developers that are paid for their work on the Linux kernel. Those contribute most changes these days. But their employers sooner or later also stop caring for their code or make its programmer focus on other things. Hardware vendors for example earn their money mainly by selling new hardware; quite a few of them hence are not investing much time and energy in maintaining a Linux kernel driver for something they stopped selling years ago. Enterprise Linux distributors often care for a longer time period, but in new versions often leave support for old and rare hardware aside to limit the scope. Often spare time contributors take over once a company orphans some code, but as mentioned above: sooner or later they will leave the code behind, too.

Priorities are another reason why some issues are not fixed, as maintainers quite often are forced to set those, as time to work on Linux is limited. That's true for spare time or the time employers grant their developers to spend on maintenance work on the upstream kernel. Sometimes maintainers also get overwhelmed with reports, even if a driver is working nearly perfectly. To not get completely stuck, the programmer thus might have no other choice than to prioritize issue reports and reject some of them.

But don't worry too much about all of this, a lot of drivers have active maintainers who are quite interested in fixing as many issues as possible.

Closing words

Compared with other Free/Libre & Open Source Software it's hard to report issues to the Linux kernel developers: the length and complexity of this document and the implications between the lines illustrate that. But that's how it is for now. The main author of this text hopes documenting the state of the art will lay some groundwork to improve the situation over time.