Bisecting a bug

Last updated: 28 October 2016

Introduction

Always try the latest kernel from kernel.org and build from source. If you are not confident in doing that please report the bug to your distribution vendor instead of to a kernel developer.

Finding bugs is not always easy. Have a go though. If you can't find it don't give up. Report as much as you have found to the relevant maintainer. See MAINTAINERS for who that is for the subsystem you have worked on.

Before you submit a bug report read 'Documentation/admin-guide/reporting-issues.rst'.

Devices not appearing

Often this is caused by udev/systemd. Check that first before blaming it on the kernel.

Finding patch that caused a bug

Using the provided tools with `git` makes finding bugs easy provided the bug is reproducible.

Steps to do it:

- build the Kernel from its git source
- start bisect with [1]:

  ```
  $ git bisect start
  ```

- mark the broken changeset with:

  ```
  $ git bisect bad [commit]
  ```

- mark a changeset where the code is known to work with:

  ```
  $ git bisect good [commit]
  ```

- rebuild the Kernel and test
- interact with git bisect by using either:

  ```
  $ git bisect good
  ```

  or:

  ```
  $ git bisect bad
  ```

  depending if the bug happened on the changeset you're testing
- After some interactions, git bisect will give you the changeset that likely caused the bug.
- For example, if you know that the current version is bad, and version 4.8 is good, you could do:

  ```
  $ git bisect start
  $ git bisect bad                 # Current version is bad
  $ git bisect good v4.8
  ```

[1] You can, optionally, provide both good and bad arguments at git start with `git bisect start [BAD] [GOOD]`

For further references, please read:

- The man page for `git-bisect`
- Fighting regressions with git bisect
- Fully automated bisecting with "git bisect run"
- Using Git bisect to figure out when brokenness was introduced