Introduction

The Linux DRM layer contains code intended to support the needs of complex graphics devices, usually containing programmable pipelines well suited to 3D graphics acceleration. Graphics drivers in the kernel may make use of DRM functions to make tasks like memory management, interrupt handling and DMA easier, and provide a uniform interface to applications.

A note on versions: this guide covers features found in the DRM tree, including the TTM memory manager, output configuration and mode setting, and the new vblank internals, in addition to all the regular features found in current kernels.

[Insert diagram of typical DRM stack here]

Style Guidelines

For consistency this documentation uses American English. Abbreviations are written as all-uppercase, for example: DRM, KMS, IOCTL, CRTC, and so on. To aid in reading, documentations make full use of the markup characters kerneldoc provides: @parameter for function parameters, @member for structure members (within the same structure), &struct structure to reference structures and function() for functions. These all get automatically hyperlinked if kerneldoc for the referenced objects exists. When referencing entries in function vtables (and structure members in general) please use &vtable_name.vfunc. Unfortunately this does not yet yield a direct link to the member, only the structure.

Except in special situations (to separate locked from unlocked variants) locking requirements for functions aren't documented in the kerneldoc. Instead locking should be check at runtime using e.g. `WARN_ON(!mutex_is_locked(...));`. Since it's much easier to ignore documentation than runtime noise this provides more value. And on top of that runtime checks do need to be updated when the locking rules change, increasing the chances that they're correct. Within the documentation the locking rules should be explained in the relevant structures: Either in the comment for the lock explaining what it protects, or data fields need a note about which lock protects them, or both.

Functions which have a non-`void` return value should have a section called "Returns" explaining the expected return values in different cases and their meanings. Currently there's no consensus whether that section name should be all upper-case or not, and whether it should end in a colon or not. Go with the file-local style. Other common section names are "Notes" with information for dangerous or tricky corner cases, and "FIXME" where the interface could be cleaned up.

Also read the :ref:`guidelines for the kernel documentation at large <doc_guide>`.

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\[linux-master][Documentation][gpu]introduction.rst`, line 52); *backlink***
>
> Unknown interpreted text role "ref".

Documentation Requirements for kAPI

All kernel APIs exported to other modules must be documented, including their datastructures and at least a short introductory section explaining the overall concepts. Documentation should be put into the code itself as kerneldoc comments as much as reasonable.

Do not blindly document everything, but document only what's relevant for driver authors: Internal functions of drm.ko and definitely static functions should not have formal kerneldoc comments. Use normal C comments if you feel like a comment is warranted. You may use kerneldoc syntax in the comment, but it shall not start with a /** kerneldoc marker. Similar for data structures, annotate anything entirely private with `/* private: */` comments as per the documentation guide.

Getting Started

Developers interested in helping out with the DRM subsystem are very welcome. Often people will resort to sending in patches for various issues reported by checkpatch or sparse. We welcome such contributions.

Anyone looking to kick it up a notch can find a list of janitorial tasks on the :ref:`TODO list <todo>`.

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\[linux-master][Documentation][gpu]introduction.rst`, line 77); *backlink***
>
> Unknown interpreted text role "ref".

Contribution Process

Mostly the DRM subsystem works like any other kernel subsystem, see :ref:`the main process guidelines and documentation

<process_index>` for how things work. Here we just document some of the specialities of the GPU subsystem.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\[linux-master][Documentation][gpu]introduction.rst, line 83); *backlink***
>
> Unknown interpreted text role "ref".

Feature Merge Deadlines

All feature work must be in the linux-next tree by the -rc6 release of the current release cycle, otherwise they must be postponed and can't reach the next merge window. All patches must have landed in the drm-next tree by latest -rc7, but if your branch is not in linux-next then this must have happened by -rc6 already.

After that point only bugfixes (like after the upstream merge window has closed with the -rc1 release) are allowed. No new platform enabling or new drivers are allowed.

This means that there's a blackout-period of about one month where feature work can't be merged. The recommended way to deal with that is having a -next tree that's always open, but making sure to not feed it into linux-next during the blackout period. As an example, drm-misc works like that.

Code of Conduct

As a freedesktop.org project, dri-devel, and the DRM community, follows the Contributor Covenant, found at: https://www.freedesktop.org/wiki/CodeOfConduct

Please conduct yourself in a respectful and civilised manner when interacting with community members on mailing lists, IRC, or bug trackers. The community represents the project as a whole, and abusive or bullying behaviour is not tolerated by the project.