Sync File API Guide

Author: Gustavo Padovan <gustavo at padovan dot org>

This document serves as a guide for device drivers writers on what the sync_file API is, and how drivers can support it. Sync file is the carrier of the fences(struct dma_fence) that are needed to synchronize between drivers or across process boundaries.

The sync_file API is meant to be used to send and receive fence information to/from userspace. It enables userspace to do explicit fencing, where instead of attaching a fence to the buffer a producer driver (such as a GPU or V4L driver) sends the fence related to the buffer to userspace via a sync_file.

The sync_file then can be sent to the consumer (DRM driver for example), that will not use the buffer for anything before the fence(s) signals, i.e., the driver that issued the fence is not using/processing the buffer anymore, so it signals that the buffer is ready to use. And vice-versa for the consumer -> producer part of the cycle.

Sync files allows userspace awareness on buffer sharing synchronization between drivers.

Sync file was originally added in the Android kernel but current Linux Desktop can benefit a lot from it.

in-fences and out-fences

Sync files can go either to or from userspace. When a sync_file is sent from the driver to userspace we call the fences it contains 'out-fences'. They are related to a buffer that the driver is processing or is going to process, so the driver creates an out-fence to be able to notify, through dma_fence_signal(), when it has finished using (or processing) that buffer. Out-fences are fences that the driver creates.

On the other hand if the driver receives fence(s) through a sync_file from userspace we call these fence(s) 'in-fences'. Receiving in-fences means that we need to wait for the fence(s) to signal before using any buffer related to the in-fences.

Creating Sync Files

When a driver needs to send an out-fence userspace it creates a sync_file.

Interface:

```
struct sync_file *sync_file_create(struct dma_fence *fence);
```

The caller pass the out-fence and gets back the sync_file. That is just the first step, next it needs to install an fd on sync_file->file. So it gets an fd:

```
fd = get_unused_fd_flags(O_CLOEXEC);
```

and installs it on sync_file->file:

```
fd_install(fd, sync_file->file);
```

The sync_file fd now can be sent to userspace.

If the creation process fail, or the sync_file needs to be released by any other reason fput(sync_file->file) should be used.

Receiving Sync Files from Userspace

When userspace needs to send an in-fence to the driver it passes file descriptor of the Sync File to the kernel. The kernel can then retrieve the fences from it.

Interface:

```
struct dma_fence *sync_file_get_fence(int fd);
```

The returned reference is owned by the caller and must be disposed of afterwards using dma_fence_put(). In case of error, a NULL is returned instead.

References:

1. struct sync_file in include/linux/sync_file.h
2. All interfaces mentioned above defined in include/linux/sync_file.h