The Android binderfs Filesystem

Android binderfs is a filesystem for the Android binder IPC mechanism. It allows to dynamically add and remove binder devices at runtime. Binder devices located in a new binderfs instance are independent of binder devices located in other binderfs instances. Mounting a new binderfs instance makes it possible to get a set of private binder devices.

Mounting binderfs

Android binderfs can be mounted with:

```
mkdir /dev/binderfs
mount -t binder binder /dev/binderfs
```

at which point a new instance of binderfs will show up at `/dev/binderfs`. In a fresh instance of binderfs no binder devices will be present. There will only be a `binder-control` device which serves as the request handler for binderfs. Mounting another binderfs instance at a different location will create a new and separate instance from all other binderfs mounts. This is identical to the behavior of e.g. `devpts` and `tmpfs`. The Android binderfs filesystem can be mounted in user namespaces.

Options

max

binderfs instances can be mounted with a limit on the number of binder devices that can be allocated. The `max=<count>` mount option serves as a per-instance limit. If `max=<count>` is set then only `<count>` number of binder devices can be allocated in this binderfs instance.

stats

Using `stats=global` enables global binder statistics. `stats=global` is only available for a binderfs instance mounted in the initial user namespace. An attempt to use the option to mount a binderfs instance in another user namespace will return a permission error.

Allocating binder Devices

To allocate a new binder device in a binderfs instance a request needs to be sent through the `binder-control` device node. A request is sent in the form of an ioctl().

What a program needs to do is to open the `binder-control` device node and send a `BINDER_CTL_ADD` request to the kernel. Users of binderfs need to tell the kernel which name the new binder device should get. By default a name can only contain up to `BINDERFS_MAX_NAME` chars including the terminating zero byte.

Once the request is made via an ioctl() passing a `struct binder_device` with the name to the kernel it will allocate a new binder device and return the major and minor number of the new device in the struct (This is necessary because binderfs allocates a major device number dynamically.). After the ioctl() returns there will be a new binder device located under /dev/binderfs with the chosen name.

Deleting binder Devices

Binderfs binder devices can be deleted via unlink(). This means that the rm() tool can be used to delete them. Note that the `binder-control` device cannot be deleted since this would make the binderfs instance unusable. The `binder-control` device will be deleted when the binderfs instance is unmounted and all references to it have been dropped.

Binder features

Assuming an instance of binderfs has been mounted at `/dev/binderfs`, the features supported by the binder driver can be located under `/dev/binderfs/features/`. The presence of individual files can be tested to determine whether a particular feature is supported by the driver.

Example:

```
cat /dev/binderfs/features/oneway_spam_detection
1
```