USB Raw Gadget

USB Raw Gadget is a gadget driver that gives userspace low-level control over the gadget's communication process.

Like any other gadget driver, Raw Gadget implements USB devices via the USB gadget API. Unlike most gadget drivers, Raw Gadget does not implement any concrete USB functions itself but requires userspace to do that.

Raw Gadget is currently a strictly debugging feature and should not be used in production. Use GadgetFS instead.

Enabled with CONFIG_USB_RAW_GADGET.

Comparison to GadgetFS

Raw Gadget is similar to GadgetFS but provides more direct access to the USB gadget layer for userspace. The key differences are:

1. Raw Gadget passes every USB request to userspace to get a response, while GadgetFS responds to some USB requests internally based on the provided descriptors. Note that the UDC driver might respond to some requests on its own and never forward them to the gadget layer.
2. Raw Gadget allows providing arbitrary data as responses to USB requests, while GadgetFS performs sanity checks on the provided USB descriptors. This makes Raw Gadget suitable for fuzzing by providing malformed data as responses to USB requests.
3. Raw Gadget provides a way to select a UDC device/driver to bind to, while GadgetFS currently binds to the first available UDC. This allows having multiple Raw Gadget instances bound to different UDCs.
4. Raw Gadget explicitly exposes information about endpoints addresses and capabilities. This allows the user to write UDC-agnostic gadgets.
5. Raw Gadget has an ioctl-based interface instead of a filesystem-based one.

Userspace interface

The user can interact with Raw Gadget by opening `/dev/raw-gadget` and issuing ioctl calls; see the comments in include/uapi/linux/usb/raw_gadget.h for details. Multiple Raw Gadget instances (bound to different UDCs) can be used at the same time.

A typical usage scenario of Raw Gadget:

1. Create a Raw Gadget instance by opening `/dev/raw-gadget`.
2. Initialize the instance via `USB_RAW_IOCTL_INIT`.
3. Launch the instance with `USB_RAW_IOCTL_RUN`.
4. In a loop issue `USB_RAW_IOCTL_EVENT_FETCH` to receive events from Raw Gadget and react to those depending on what kind of USB gadget must be implemented.

Note that some UDC drivers have fixed addresses assigned to endpoints, and therefore arbitrary endpoint addresses cannot be used in the descriptors. Nevertheless, Raw Gadget provides a UDC-agnostic way to write USB gadgets. Once `USB_RAW_EVENT_CONNECT` is received via `USB_RAW_IOCTL_EVENT_FETCH`, `USB_RAW_IOCTL_EPS_INFO` can be used to find out information about the endpoints that the UDC driver has. Based on that, userspace must choose UDC endpoints for the gadget and assign addresses in the endpoint descriptors correspondingly.

Raw Gadget usage examples and a test suite:

https://github.com/xairy/raw-gadget

Internal details

Every Raw Gadget endpoint read/write ioctl submits a USB request and waits until its completion. This is done deliberately to assist with coverage-guided fuzzing by having a single syscall fully process a single USB request. This feature must be kept in the implementation.

Potential future improvements

- Report more events (suspend, resume, etc.) through `USB_RAW_IOCTL_EVENT_FETCH`.
- Support `O_NONBLOCK` I/O. This would be another mode of operation, where Raw Gadget would not wait until the completion of each USB request.
- Support USB 3 features (accept SS endpoint companion descriptor when enabling endpoints; allow providing `stream_id` for bulk transfers).
- Support ISO transfer features (expose `frame_number` for completed requests).