udev Rules

udev stands for *userspace /dev* and is a device manager for the Linux kernel. The `/dev` path is ordinarily where device information is accessed from, *udev rules* allow you to alter this information in a predictable and permanent manner. For example, what drivers to use/blacklist for a specific device being plugged in.

udev rules are checked when:

1. Booting up, all udev rules are parsed and a rules database is built in memory
2. When an event happens, it checks its rule database and performs the appropriate actions.

In a nutshell, udev rules do always have two major parts:

- **Matches** ("If the condition(s) matches"...). The operator for matches are == and `!=`
- **Actions** (..."then do this!"). The operator for actions is: =

udev rules in Batocera

Batocera supports booting up and event udev rules like with any ordinary operating system. These can be manipulated in the usual `/etc/udev/rules.d/` directory. However, changes cannot be permanently stored there due to Batocera's filesystem (overlays can be used, but are forgotten between upgrades).

But if you only need event udev rules, then the `/userdata/system/udev/rules.d/` path can be used instead. The path must be initially created manually.

For example, if you wanted to enable the Bluetooth passthrough udev rule of Dolphin for your own Bluetooth dongle (list of compatible adapters **for Dolphin, not necessarily Batocera** is further down that page), you would use this file:

52-dolphin.rules

```
SUBSYSTEM=="usb", ATTRS{idVendor}=="YOURVID",
ATTRS{idProduct}=="YOURPID", TAG+="uaccess"
```

Where the values `YOURVID` and `YOURPID` are replaced according to your specific device found in the list mentioned above. Then save it and copy it to `/userdata/system/udev/rules.d/52-dolphin.rules` and reboot Batocera.

If the udev rule does not function in this path, then it is a booting up udev rule and must use the aforementioned `/etc/udev/rules.d/` directory.

If you need something activated earlier in the boot process or something more intimate to the system, you can instead use boot scripts.

Udev doesn't strictly need the VID or PID to identify a device, its name attribute can be used instead (however this is more susceptible to being applied to the similar devices connected).

Deeper explanation of udev rules

As an example, let's take the udev rule `52-dolphin.rules` from above to dive a little deeper into udev rules:

SUBSYSTEM, `ATTRS{idVendor}` and `ATTRS{idProduct}` are all variables that do all have a == between them and their assigned values. This means they are all *matches* (conditions). On the other hand, the last variable TAG does have a +=, which adds itself to any other TAG action being executed on this specific event (e.g. by an additional udev rule matching the same event).

This udev rule can be read as follows:

If the detected device is a USB device with the vendor ID 'YOURVID' and the product ID 'YOURPID', then, in addition to any other TAG action on this specific event, tag the device with 'uaccess' (which provides access to the `/dev/` file for the logged in user (in case of Batocera, mostly **root** user)).

Viewing and editing pre-existing udev rules

Once Batocera has booted you can view the existing udev rules for your system in `/etc/udev/rules.d/`. You can do this from the file manager (`[F1]` on the system list) and going up one level, or via SSH. You cannot access this location from the network share, as it is outside of the userdata partition.

Create your own udev rule

In order to create a udev rule either its name (or other uniquely identifiable attribute) or the VID and PID need to be discovered.

1. Connect to Batocera via SSH
2. Run the list command for the device in question:
   - For input devices, this can be achieved with `evtest`
   - For USB devices in general: `lsusb`
   - For devices in general (or if you don't know what subsystem it uses), search through the `/dev/` folder in root: `ls /dev` and then `ls /dev/<subsystem>`
3. Make an appropriate `##-<any-name-you-want>.rules` where `##` is a number (udev will apply the rules in this specified order, most of the time this can be 99 with no issues)
4. If not making a boot udev rule, it can be tested immediately by reloading the udev service:

```
udevadm control --reload-rules && udevadm trigger
```

- Once this is done, run the following to test if the rule has been parsed and applied correctly:

```
udevadm info -q all -n /dev/<subsystem>/<device>
```

Transform a keyboard into a pad

The keyboard can be identified with `evtest` (in this example, `event3`):

```
# evtest
No device specified, trying to scan all of /dev/input/event*
Available devices:
/dev/input/event0:  Lid Switch
/dev/input/event1:  Sleep Button
/dev/input/event2:  Power Button
/dev/input/event3:  AT Translated Set 2 keyboard
/dev/input/event4:  Video Bus
/dev/input/event5:  Video Bus
/dev/input/event6:  HDA Intel PCH Headphone
/dev/input/event7:  HDA Intel HDMI HDMI/DP,pcm=3
/dev/input/event8:  HDA Intel HDMI HDMI/DP,pcm=7
/dev/input/event9:  HDA Intel HDMI HDMI/DP,pcm=8
/dev/input/event10: FocalTechPS/2 FocalTech Touchpad
/dev/input/event11: Asus WMI hotkeys
/dev/input/event12: 8Bitdo  8BitDo N30 Pro 2
/dev/input/event13: USB2.0 HD UVC WebCam: USB2.0 HD
Select the device event number [0-13]:
```

Create a udev rule to set this event as a joystick. This can be achieved with this one-line command (where `AT Translated Set 2 keyboard` is the name of your keyboard):

```
echo 'SUBSYSTEM=="input", ATTRS{name}=="AT Translated Set 2 keyboard",
MODE="0666", ENV{ID_INPUT_JOYSTICK}="1", ENV{ID_INPUT_KEYBOARD}="0"' >
/etc/udev/rules.d/99-custom.rules
```

To make sure it's work, reload the rules and check if the flag is present:

```
udevadm control --reload-rules && udevadm trigger
udevadm info -q all -n /dev/input/event3
```

Troubleshooting

If you find that your udev rule isn't working after reboot for any reason, you can alternatively try putting your `*.rules` file directly into `/etc/udev/rules.d/` and running `batocera-save-overlay` afterward. This will call it slightly earlier in the boot process. Remember: Everything saved to the boot partition via `batocera-save-overlay` will get lost after updating Batocera to a new version.