Slides: https://tinyurl.com/lpc22-gpio (https://docs.google.com/presentation/d/18jdugObCQHgFjT9_1-xp67WN2iXxjdEScaSdZvzENag/edit?usp=sharing)

The slides are just meant to be a starting point. Please everyone join in with comments, questions, and ideas!

Export GPIO consumers PID to userspace:
- Bart has sent new mail today - need to rethink PID in the context of forking: https://lore.kernel.org/linux-gpio/20220909121329.42004-1-brgl@bgdev.pl/T/#mb54408a1b166f4196c933d6c787b8906f5c3cc8a
- daemon (probably over D-BUS) is planned by Bart to manage GPIO
- on fork, the kernel would need close the parents fd's, and the child would keep them
- change the fork flags being used by the gpio cdev... but this would have to go through the vfs... not pratical to land such a change?

libgpiod v2 status
- C api will be more complex, more calls to do actions, BUT this will make the higher level bindings like Python to be easier for Python programmers

libgpiod utilites:
- speed of sysfs echo vs. gpioset utility
- would python prompt be faster than sysfs... is process creation (fork) the source of the overhead?
- need for speed when bitbanging - does this compare to what the IR drivers do?
- linus: writing C program with gpiod is fast, but forking utilities like gpioset slow because of process creation

process info:
- jan: /proc/fdinfo (?) would show what has a line open. possible to annotate gpio information in those fields. i.e. like https://lore.kernel.org/all/69924bc6-d249-35b2-a942-a43a9293558e@amd.com/T/#m807be960bbbe3a6faf8a411521a1dd11d020b1d7
- jan: it might be difficult to expose pids now that there are namespaces, etc

pinctrl topic: pinmux-select
- replace functionality in bone-pinmux-helper
- device tree overalys in uboot is the most robust solution for changing pinmux
- breadboarding is maybe not a storng usecase
- marek: but FPGAs have a real runtime use case
- linusw: if security is not a concern, then debugfs
- linus: need to have threat model for gpio lines to decide whether or not it matters
- don't enable debugfs on secure systems
- security of device tree overlay contents: is it possible to have some type of signing? (but doesn't this require admin cap or root anyways)
- gpio aggregator was created with security in mind - prevent vm guest from accessing all the lines on a chip, only the lines that it needs
-

why a daemon?

- in sysfs uapi, the exported pins maintained their state, not tied to process lifetime
- bart's idea: centeral daemon to keep the lines open, preference to use d-bus as most common for systems he interacts with, and should be simplest for implementation
- jan: some use cases where d-bus not possible, other options for centeral authority
- chris: what about protobufs for low level
- jan: for low level sysetms, preference to not deal with complex interface just to handle what happens to line when process crashes
- kent working on libgpiod gpioset daemon - send commands as input
- jan: would be useful to have ability to put line into a defined state when the fd closes
- cengiz: thoughts on dbus, worked on system to use dbus for CAN and IoT systems, but the dbus overhead was too much time sensitive. 600 ms delay for button press in experiment through dbus.
- bart: in previous project found dbus latency was quite low so 600 ms seems high - maybe config issue... but also debugging dbus may be complex
- marek: dbus could be useful for virtual machine guest to interact with gpio

gpiod: how to have safe state on line fd close?
- jan: would be useful to have ability to put line into a defined state when the fd closes
- behavior would be driver dependent
- also need to set that default behavior for that line from userspace
- what about having the line go back to hog state when fd closes? (can line be opened when defined as a hog?)
- define the default state of a line in DT? this is the state when an FD is not open.
- gpiolib could call the revelant callbacks upon the fd close and release
- bart: could protoytype idea unless ideas?
- are we crossing the line of no policy in DT?
- maybe call it something like "safe state" property
- todo: dicuss with DT folks about this idea
- isn't gpio hog also policy and it is DT already... so "safe default state" should be ok?
- geert: maybe gpio-idle-state ? this is not _software_ policy, it is a hardware policy ,e.g. what the schematic dictates
- TODO: bart will start a discussion about implementing

Driver testing:
- possible to get coverage across gpio driver like a kselftest that could be used in kernelci
- chris: zephyr does this