What's New with U-Boot?

ELC 2019
San Diego, CA

Simon Glass
Chrome OS Boulder

Goals of this talk

- Tell you things useful for using U-Boot
- Update you on the current state of U-Boot
  - Changed that are completed
  - Changes still in progress
  - Potential future changes
- Point a little way into the future

- What is 'new' for you depends on how closely you follow U-Boot
  - Many people just use the same version for years
  - I will mostly focus on things in the last few years that I have some idea about...
    - ...without completely ignoring things 2-5 years old

U-Boot overview

- Tom Rini is head custodian across U-Boot (since 2012)
  - About 50 custodians for different architectures and subsystems
- About 2.2m lines of C, 35k lines of assembler
  - Various tools written mostly in C and Python
- Release cycle currently 3 months
  - Two weeks between each release candidate
- Very active and dynamic project
  - About 400 individual contributors and 6k commits in the last year
  - Many ongoing improvement efforts on code structure, testing
- Strong links to Linux and distributions
  - Some subsystems share code, also use device tree files
  - Fedora, Debian, Yocto

What's not new?

- Some things I won't talk about!

- Fast, small, simple, portable, configurable, flexible
- Wide architecture (13) and board (~1400) support
- Tertiary and Secondary Program Loader (TPL, SPL)
- Lots of support for loading different image types
  - Flexible Flat Image Tree (FIT) format with compression, hashing, signing
- 'Sandbox' architecture for rapid development/debugging on a host
- Command line with about 150 top-level commands
  - Many with sub-commands; console supports serial / video / USB console
- Wide partition, filesystem and networking support
- Subsystems and drivers for most types of peripherals

Some old new things (> 2 years)

- *Driver model*
- Device tree
- Kbuild
- Kconfig
- Verified / secure boot
- *Bootstage / trace*
- *Buildman*
- DFU / fastboot
- Coverity

Driver model

- Comprehensive and efficient driver model
- 'Uclasses' for most subsystems

```
/ {
        ...
        i2c_0: i2c@13860000 {
                #address-cells = <1>;
                #size-cells = <0>;
                compatible = "samsung,s3c2440-i2c";
};

&i2c_0 {
        max77686: max77686_pmic@09 {
                voltage-regulators {
                        ldo4_reg: LDO4 {
                                regulator-name = "VDDQ_MMC2_2.8V";
                                regulator-min-microvolt = <2800000>;
                                regulator-max-microvolt = <2800000>;
                        };
                };
        };
};
```

```
const char *mmc_regulators[] = {
        "VDDQ_EMMC_1.8V",
        "VDDQ_EMMC_2.8V",
        "TFLASH_2.8V",
        NULL,
};

if (regulator_list_autoset(mmc_regulators, NULL, true))
        pr_err("Unable to init all mmc regulators\n");
```

Bootstage and Tracing

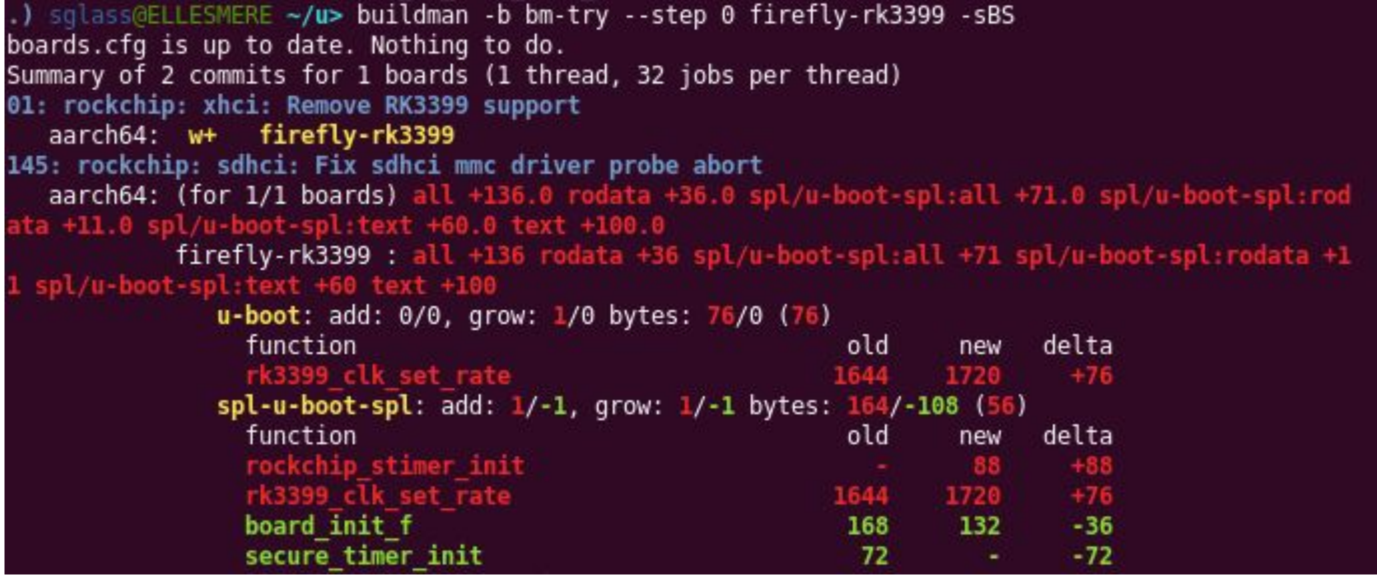
- Track boot time through all phases
  - TPL, SPL, U-Boot proper
  - Pass to Linux via device tree
- Track function calls and time

```
=> trace stats
        671,406 function sites
      1,279,450 function calls
              0 untracked function calls
        950,490 traced function calls
             16 maximum observed call depth
             15 call depth limit
      1,275,767 calls not traced due to depth
=> tftpput ${profbase} ${profoffset} 192.168.4.1:trace
```

```
bootstage report
Timer summary in microseconds:
       Mark    Elapsed  Stage
          0          0  reset
    100,000    100,000  spl_start
    842,156    742,156  board_init_f
    899,769     57,613  board_init_r
    902,927      3,158  board_init
    927,905     24,978  board_init_done
    945,247     17,342  id=64
    950,104      4,857  main_loop
    950,104          0  main_loop

Accumulated time:
               188,378  lcd
```

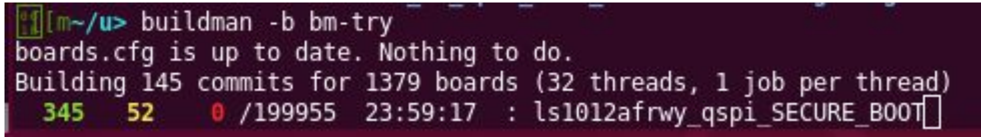
Google

Buildman



```
.) sglass@ELLESMERE ~/u> buildman -b bm-try --step 0 firefly-rk3399 -sBS
boards.cfg is up to date. Nothing to do.
Summary of 2 commits for 1 boards (1 thread, 32 jobs per thread)
01: rockchip: xhci: Remove RK3399 support
   aarch64:  w+   firefly-rk3399
145: rockchip: sdhci: Fix sdhci mmc driver probe abort
   aarch64: (for 1/1 boards) all +136.0 rodata +36.0 spl/u-boot-spl:all +71.0 spl/u-boot-spl:rod
ata +11.0 spl/u-boot-spl:text +60.0 text +100.0
            firefly-rk3399 : all +136 rodata +36 spl/u-boot-spl:all +71 spl/u-boot-spl:rodata +1
1 spl/u-boot-spl:text +60 text +100
               u-boot: add: 0/0, grow: 1/0 bytes: 76/0 (76)
                 function                                   old     new   delta
                 rk3399_clk_set_rate                       1644    1720     +76
               spl-u-boot-spl: add: 1/-1, grow: 1/-1 bytes: 164/-108 (56)
                 function                                   old     new   delta
                 rockchip_stimer_init                         -      88     +88
                 rk3399_clk_set_rate                       1644    1720     +76
                 board_init_f                               168     132     -36
                 secure_timer_init                           72       -     -72
```

- Multi-threaded build / analysis tool for U-Boot
- Automatic toolchain download
- Builds any/all boards / arch
- Supports building multiple commits, with analysis:
  - Which commits introduce or fix errors
  - Overall (and per-function '--bloat') code size changes
  - CONFIG changes, environment change



```
[m~/u> buildman -b bm-try
boards.cfg is up to date. Nothing to do.
Building 145 commits for 1379 boards (32 threads, 1 job per thread)
  345    52     0 /199955  23:59:17  : ls1012afrwy_qspi_SECURE_BOOT
```

Google

New things (< 2 years)

- Device-tree overlays
- Live tree
- OF-platdata / dtoc
- Android, OP-TEE
- Gitlab
- New hardware / automated testing
- EFI
- Documentation format
- Lots of board/arch things I won't mention (e.g. RISC-V)

Device-tree overlays

- U-Boot can do this
  - Provide a completed DT to linux
- SPL in progress

Base board

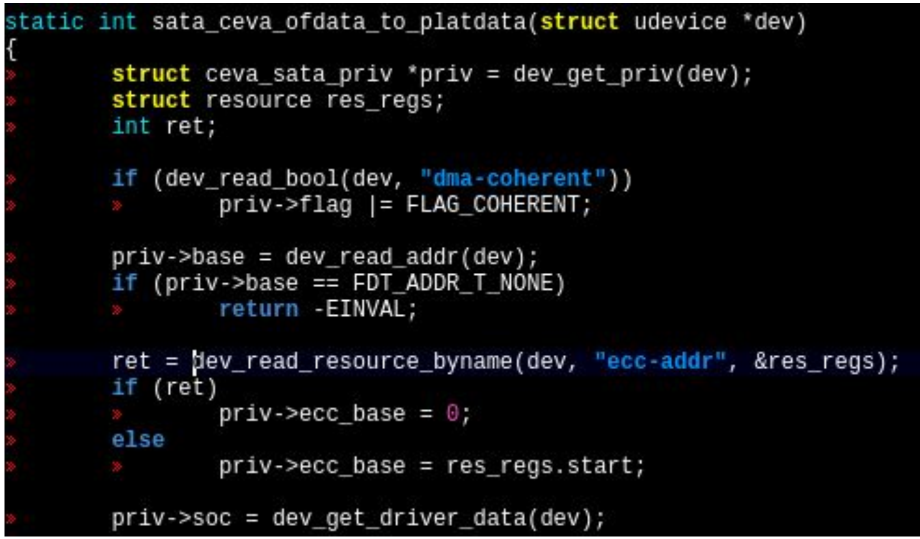
Relay board

```
=> host load hostfs - 0 /tmp/b/rpi_2/dts/dt.dtb
18837 bytes read in 0 ms
=> host load hostfs - 10000
/home/sjg/c/rpi/hd44780-lcd.dtbo
1662 bytes read in 1 ms (1.6 MiB/s)
=> fdt addr 0
=> fdt list /auxdisplay
libfdt fdt_path_offset() returned FDT_ERR_NOTFOUND
=> fdt resize
=> fdt apply 10000
=> fdt list /auxdisplay
auxdisplay {
      phandle = <0x0000005a>;
      display-width-chars = <0x00000010>;
      display-height-chars = <0x00000002>;
      rs-gpios = <0x00000016 0x00000014 0x00000000>;
      enable-gpios = <0x00000016 0x00000015 0x00000000>;
      compatible = "hit,hd44780";
   ...
};
=>
```

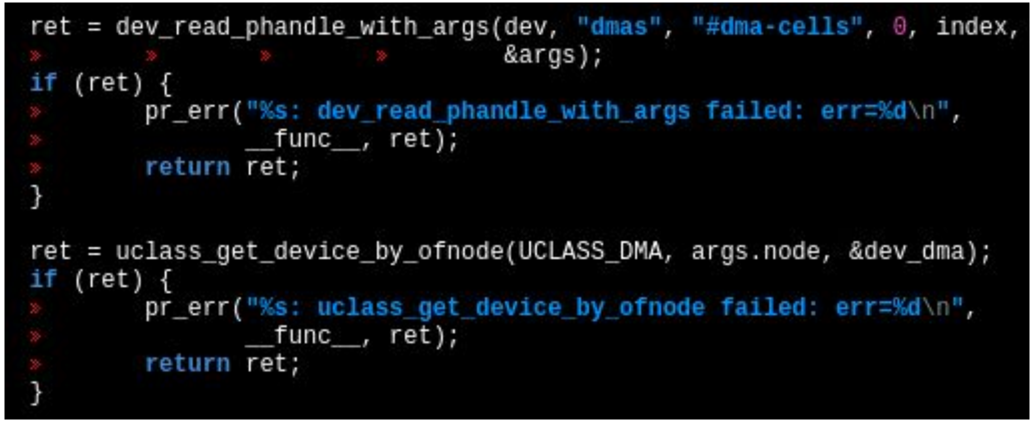
Google

Live tree

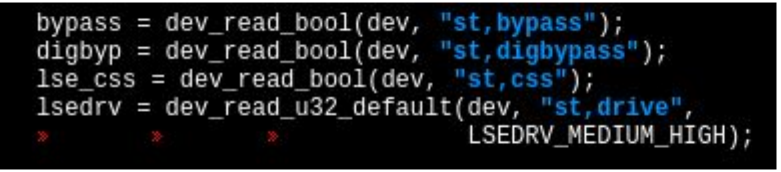
- CONFIG_OF_LIVE
- New dev_read_...() API
    - Supports flat and live tree transparently
- Tree is 'unflattened' during relocation
    - Live tree used after that



```
static int sata_ceva_ofdata_to_platdata(struct udevice *dev)
{
	struct ceva_sata_priv *priv = dev_get_priv(dev);
	struct resource res_regs;
	int ret;

	if (dev_read_bool(dev, "dma-coherent"))
		priv->flag |= FLAG_COHERENT;

	priv->base = dev_read_addr(dev);
	if (priv->base == FDT_ADDR_T_NONE)
		return -EINVAL;

	ret = dev_read_resource_byname(dev, "ecc-addr", &res_regs);
	if (ret)
		priv->ecc_base = 0;
	else
		priv->ecc_base = res_regs.start;

	priv->soc = dev_get_driver_data(dev);
```



```
ret = dev_read_phandle_with_args(dev, "dmas", "#dma-cells", 0, index,
				 &args);
if (ret) {
	pr_err("%s: dev_read_phandle_with_args failed: err=%d\n",
	       __func__, ret);
	return ret;
}

ret = uclass_get_device_by_ofnode(UCLASS_DMA, args.node, &dev_dma);
if (ret) {
	pr_err("%s: uclass_get_device_by_ofnode failed: err=%d\n",
	       __func__, ret);
	return ret;
}
```



```
bypass = dev_read_bool(dev, "st,bypass");
digbyp = dev_read_bool(dev, "st,digbypass");
lse_css = dev_read_bool(dev, "st,css");
lsedrv = dev_read_u32_default(dev, "st,drive",
			      LSEDRV_MEDIUM_HIGH);
```

Google

```
&dmc {
        rockchip,memory-schedule = <DMC_MSCH_CBDR>;
        rockchip,ddr-frequency = <800000000>;
        rockchip,ddr-speed-bin = <DDR3_1600K>;
};
```

OF-platdata / dtoc

- libfdt and DT add 6-7KB size to U-Boot SPL
- OF-platdata avoids this
- dtoc generates C structures from selected DT nodes automatically

```
};
struct dtd_rockchip_rk3368_dmc {
        fdt64_t         reg[4];
        fdt32_t         rockchip_cru;
        fdt32_t         rockchip_ddr_frequency;
        fdt32_t         rockchip_ddr_speed_bin;
        fdt32_t         rockchip_grf;
        fdt32_t         rockchip_memory_schedule;
        fdt32_t         rockchip_msch;
};
```

```
static const struct dtd_rockchip_rk3368_dmc dtv_dmc_at_ff610000 =
        .reg                     = {0xff610000, 0x400, 0xff620000,
        .rockchip_cru            = 0xb,
        .rockchip_ddr_frequency = 0x2faf0800,
        .rockchip_ddr_speed_bin = 0xc,
        .rockchip_grf            = 0xc,
        .rockchip_memory_schedule = 0x0,
        .rockchip_msch           = 0xd,
};
U_BOOT_DEVICE(dmc_at_ff610000) = {
        .name           = "rockchip_rk3368_dmc",
        .platdata       = &dtv_dmc_at_ff610000,
        .platdata_size  = sizeof(dtv_dmc_at_ff610000),
};
```

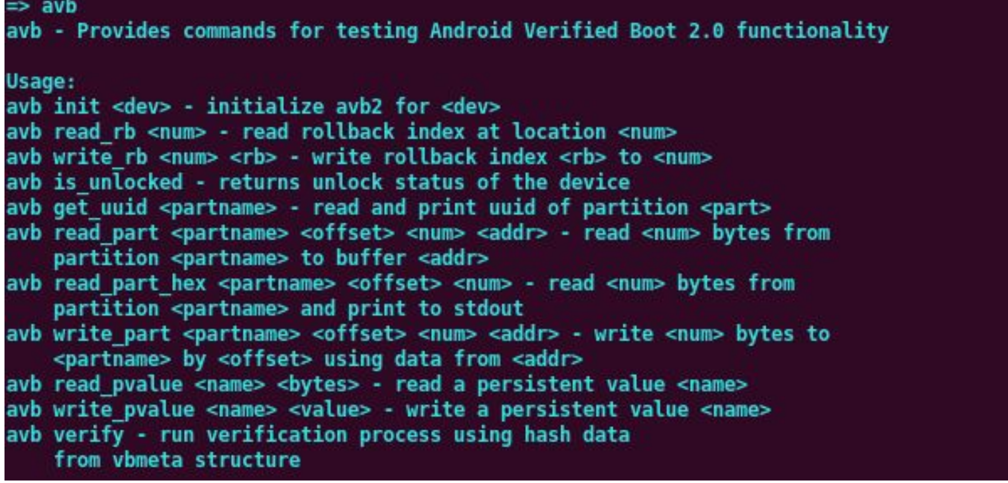
Example of-platdata driver code

```
#if CONFIG_IS_ENABLED(OF_PLATDATA)
static int conv_of_platdata(struct udevice *dev)
{
      struct rk3368_sdram_params *plat = dev_get_platdata(dev);
      struct dtd_rockchip_rk3368_dmc *of_plat = &plat->of_plat;

      plat->ddr_freq = of_plat->rockchip_ddr_frequency;
      plat->ddr_speed_bin = of_plat->rockchip_ddr_speed_bin;
      plat->memory_schedule = of_plat->rockchip_memory_schedule;

      return 0;
}
#endif

static int rk3368_dmc_probe(struct udevice *dev)
{
      struct dram_info *priv = dev_get_priv(dev);

#if CONFIG_IS_ENABLED(OF_PLATDATA)
      ret = conv_of_platdata(dev);
      if (ret)
            return ret;
#endif
```

```
struct rk3368_sdram_params {
#if CONFIG_IS_ENABLED(OF_PLATDATA)
      struct dtd_rockchip_rk3368_dmc of_plat;
#endif
      struct rk3288_sdram_pctl_timing
pctl_timing;
      u32 trefi_mem_ddr3;
      struct rk3288_sdram_channel chan;
      struct regmap *map;
};
```

Android and OP-TEE

- Based on Chrome OS verified boot
  - Which is partly based on Android...
- libavb incorporated into U-Boot
- New 'avb' command
- New 'tee' uclass (no command yet)

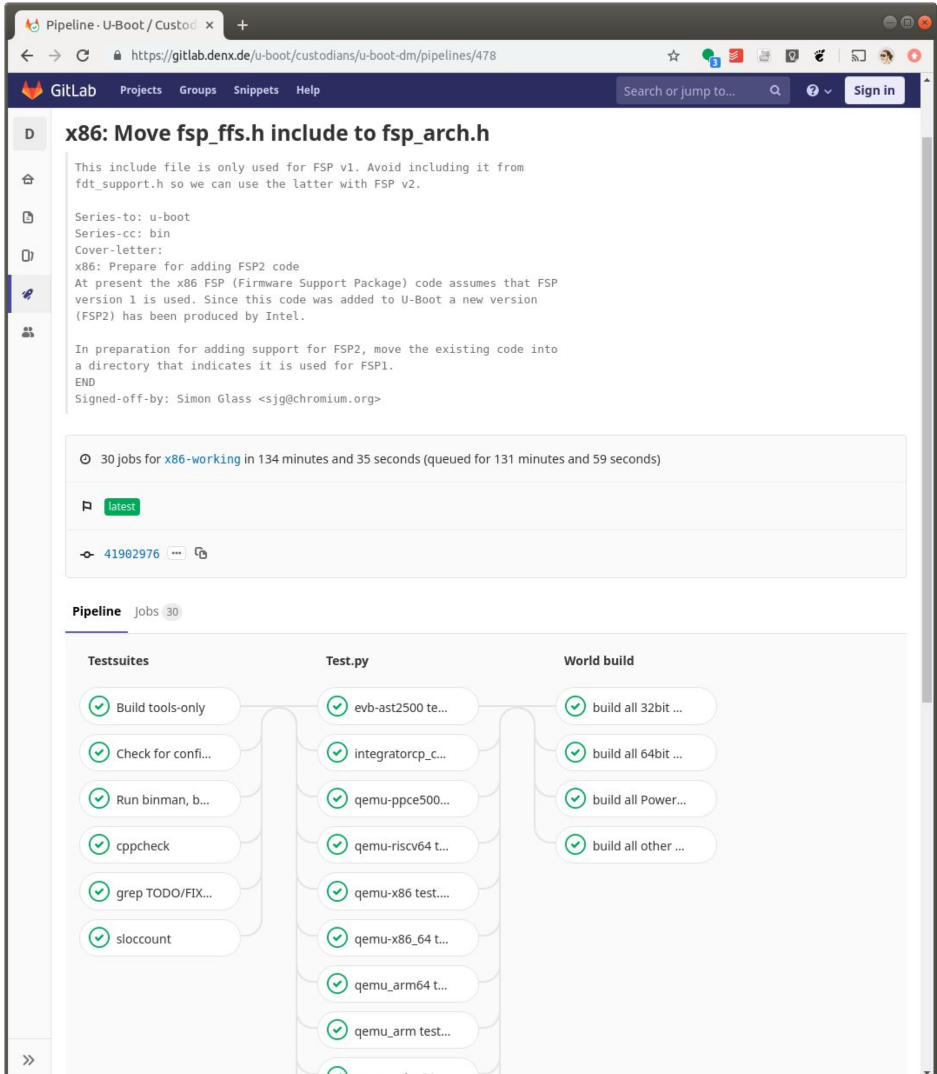


```
=> avb
avb - Provides commands for testing Android Verified Boot 2.0 functionality

Usage:
avb init <dev> - initialize avb2 for <dev>
avb read_rb <num> - read rollback index at location <num>
avb write_rb <num> <rb> - write rollback index <rb> to <num>
avb is_unlocked - returns unlock status of the device
avb get_uuid <partname> - read and print uuid of partition <part>
avb read_part <partname> <offset> <num> <addr> - read <num> bytes from
    partition <partname> to buffer <addr>
avb read_part_hex <partname> <offset> <num> - read <num> bytes from
    partition <partname> and print to stdout
avb write_part <partname> <offset> <num> <addr> - write <num> bytes to
    <partname> by <offset> using data from <addr>
avb read_pvalue <name> <bytes> - read a persistent value <name>
avb write_pvalue <name> <value> - write a persistent value <name>
avb verify - run verification process using hash data
    from vbmeta structure
```

```
avb_verify=avb init $mmcdev; avb verify;
if run avb_verify; then
   echo AVB verification OK. Continue boot;
   set bootargs $bootargs $avb_bootargs;
else
   echo AVB verification failed;
   exit;
fi;
```

Gitlab

- U-Boot custodian trees moved to to Gitlab in mid 2019
  - Travis-CI still maintained for now
- Automatic builds / notifications
  - Help out by adding a build server to increase capacity



Pipeline · U-Boot / Custod
https://gitlab.denx.de/u-boot/custodians/u-boot-dm/pipelines/478
GitLab
Projects
Groups
Snippets
Help
Search or jump to...
Sign in

x86: Move fsp_ffs.h include to fsp_arch.h

```
This include file is only used for FSP v1. Avoid including it from
fdt_support.h so we can use the latter with FSP v2.

Series-to: u-boot
Series-cc: bin
Cover-letter:
x86: Prepare for adding FSP2 code
At present the x86 FSP (Firmware Support Package) code assumes that FSP
version 1 is used. Since this code was added to U-Boot a new version
(FSP2) has been produced by Intel.

In preparation for adding support for FSP2, move the existing code into
a directory that indicates it is used for FSP1.
END
Signed-off-by: Simon Glass <sjg@chromium.org>
```

30 jobs for x86-working in 134 minutes and 35 seconds (queued for 131 minutes and 59 seconds)

latest

41902976

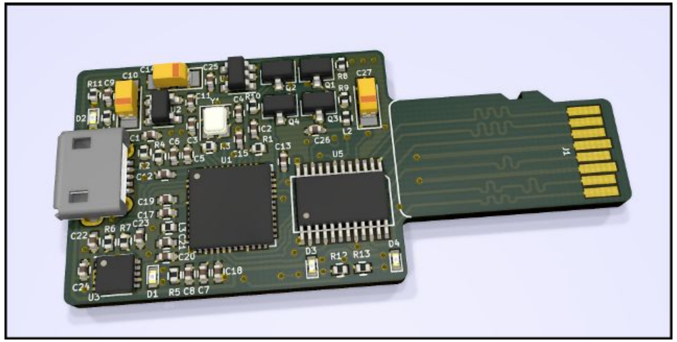
Pipeline | Jobs 30

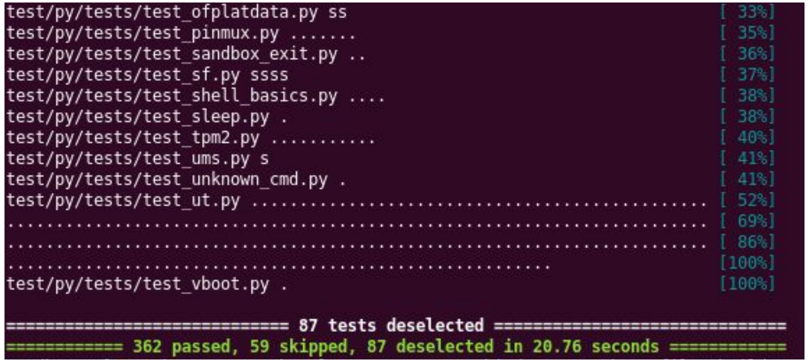
Testsuites	Test.py	World build
✓ Build tools-only	✓ evb-ast2500 te...	✓ build all 32bit ...
✓ Check for confi...	✓ integratorcp_c...	✓ build all 64bit ...
✓ Run binman, b...	✓ qemu-ppce500...	✓ build all Power...
✓ cppcheck	✓ qemu-riscv64 t...	✓ build all other ...
✓ grep TODO/FIX...	✓ qemu-x86 test....	
✓ sloccount	✓ qemu-x86_64 t...	
	✓ qemu_arm64 t...	
	✓ qemu_arm test...	

Google

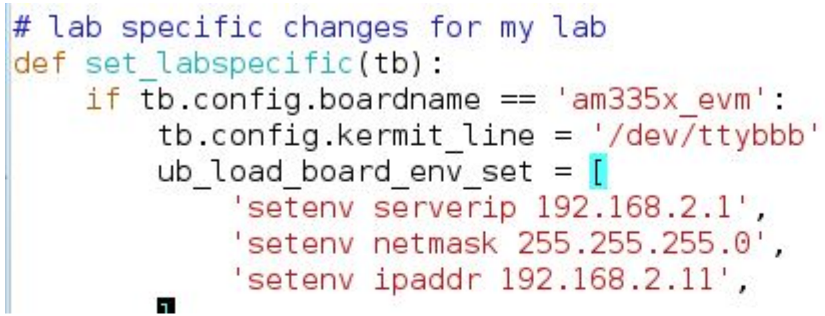
New hardware / automated testing

- pytest
- tbot
- Target control - FlashAir, SDWire
- Planning to connect to gitlab





```
test/py/tests/test_ofplatdata.py ss                                   [ 33%]
test/py/tests/test_pinmux.py .......                                  [ 35%]
test/py/tests/test_sandbox_exit.py ..                                 [ 36%]
test/py/tests/test_sf.py ssss                                         [ 37%]
test/py/tests/test_shell_basics.py ....                               [ 38%]
test/py/tests/test_sleep.py .                                         [ 38%]
test/py/tests/test_tpm2.py ..........                                 [ 40%]
test/py/tests/test_ums.py s                                           [ 41%]
test/py/tests/test_unknown_cmd.py .                                   [ 41%]
test/py/tests/test_ut.py ............................................ [ 52%]
..................................................................... [ 69%]
..................................................................... [ 86%]
.....................................................                 [100%]
test/py/tests/test_vboot.py .                                         [100%]

=========================== 87 tests deselected ============================
=========== 362 passed, 59 skipped, 87 deselected in 20.76 seconds ===========
```



```
# lab specific changes for my lab
def set_labspecific(tb):
    if tb.config.boardname == 'am335x_evm':
        tb.config.kermit_line = '/dev/ttybbb'
        ub_load_board_env_set = [
            'setenv serverip 192.168.2.1',
            'setenv netmask 255.255.255.0',
            'setenv ipaddr 192.168.2.11',
```

Google

EFI

- U-Boot can run EFI programs
  - Used for some distributions (SUSE)
  - EFI support has grown significantly in the last few years
  - Replace UEFI in many cases
  - E.g. supports booting grub2
  - Includes storage, console, networking, etc.
- Good set of automated tests

- Also can boot U-Boot as an EFI payload
  - It loads as an EFI app and then takes over!

Documentation format

- U-Boot has a lot of features
  - About 550 files in doc/
- Recently moved to restructured text (.rst)
- Directory structure is starting to mirror code
  - doc/arch/…
  - doc/board/...

Random other things dear to my heart

- Binman
- x86 support
- Logging
- Patman

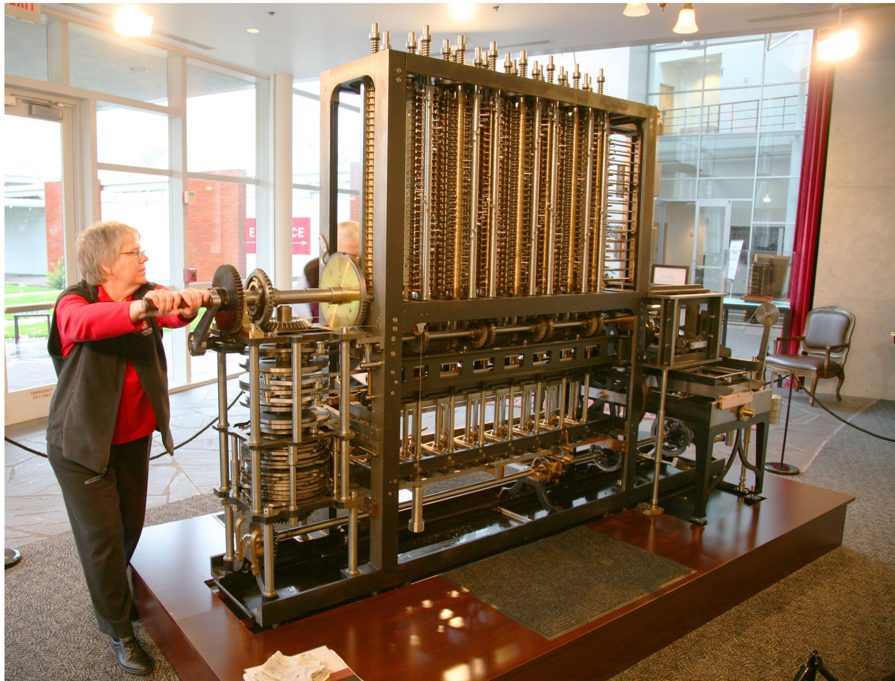
Binman

- Firmware packer
- Operates from a device-tree config
- Image info available to U-Boot at run-time
  - Position of each entry in the image
  - Via device tree or automatic linker symbols
- Supports signing, CBFS, related entries
- Easy to extend (written in Python)
- Fast (generally one pass)
- Tests provide 100% code coverage

```
#include <config.h>

/ {
   binman {
      filename = "image.rom";
      pad-byte = <0xff>;
      u-boot-spl {
      };
      u-boot-img {
         offset = <CONFIG_SPL_PAD_TO>;
      };
      cbfs {
         size = <0x10000>;
         intel-vga {
            cbfs-type = "raw";
         };
         intel-fsp-m {
            cbfs-type = "raw";
            cbfs-compress = "lz4";
         };
         intel-fsp-s {
            cbfs-type = "raw";
            cbfs-compress = "lz4";
         };
      };
   };
};
```

x86 support

- Supports bare-metal on about 10 SoCs (e.g. Broadwell, Apololake soon)
  - Supports booting from coreboot on most boards
- Intel FSP support for several platforms
  - FSP2 support in the works :-)
- Full use of driver model
- Binman provides image structure
  - Exquisitely complex
- New slimbootloader support



Google

Logging

```
U-Boot 2019.10-rc2-00016-g81fed78c0a (Aug 19 2019 - 07:28:05
-0600)

Model: sandbox
u-boot, a command line test interface to U-Boot

Usage: u-boot [options]
Options:
  -L, --log_level         <arg>   Set log level (0=panic, 7=debug)
```

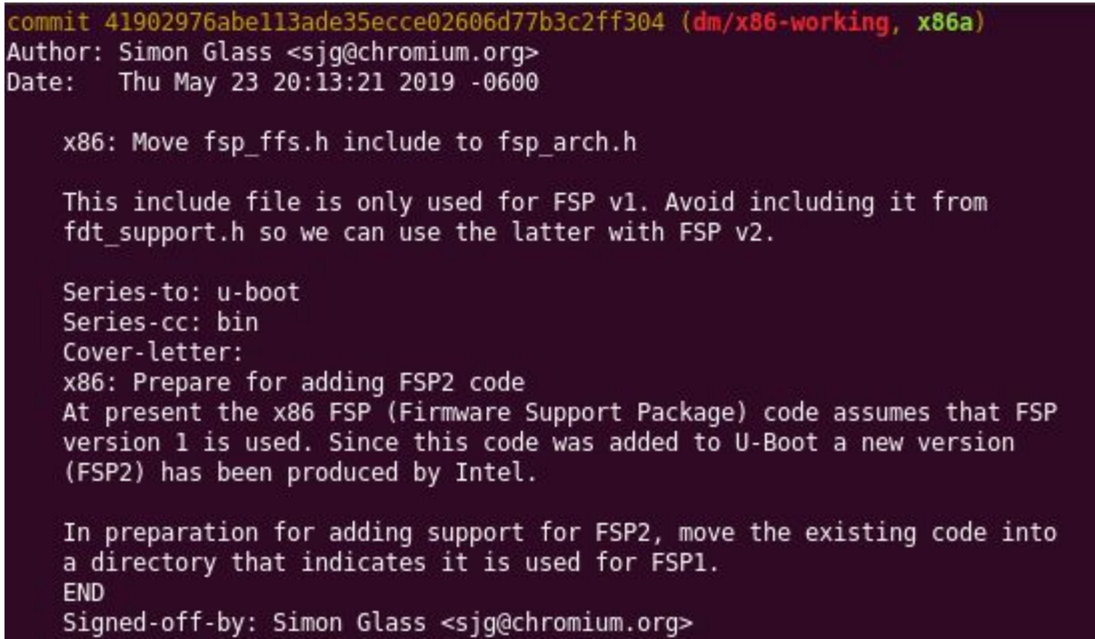
- Provides a way to log events
  - Either to console or your own driver
  - E.g. store in memory for passing to Linux
- log_debug(), log_warn(), etc.
- Supports log levels and log categories
  - Build-time and run-time filtering
- Can select log level to build with (to reduce code size)

```
if (hdr->magic != BLOBLIST_MAGIC)
      return log_msg_ret("Bad magic", -ENOENT);

log(LOGC_BLOBLIST, LOGL_DEBUG, "Found existing bloblist\n");
```

Google

Patman

- Easily check and sent patches to mailing lists
  - U-Boot, kernel and others
- Manages change logs and cover letter
- Avoids common user errors
- Little demo if time



```
commit 41902976abe113ade35ecce02606d77b3c2ff304 (dm/x86-working, x86a)
Author: Simon Glass <sjg@chromium.org>
Date:   Thu May 23 20:13:21 2019 -0600

    x86: Move fsp_ffs.h include to fsp_arch.h

    This include file is only used for FSP v1. Avoid including it from
    fdt_support.h so we can use the latter with FSP v2.

    Series-to: u-boot
    Series-cc: bin
    Cover-letter:
    x86: Prepare for adding FSP2 code
    At present the x86 FSP (Firmware Support Package) code assumes that FSP
    version 1 is used. Since this code was added to U-Boot a new version
    (FSP2) has been produced by Intel.

    In preparation for adding support for FSP2, move the existing code into
    a directory that indicates it is used for FSP1.
    END
    Signed-off-by: Simon Glass <sjg@chromium.org>
```

Google

How might U-Boot look in a few years?

- U-Boot's direction is set by its contributors
- Contributions often come out of the blue
  - "I wish U-Boot could..."
  - "My architecture needs to be able to..."
  - "The xxx implementation is terrible..."
  - "We need a new way to define...."
- U-Boot exists to solve the booting problem
  - As needs evolve, so will U-Boot

- But since you asked...

Google

How might U-Boot look in a few years?

- Most custodians will have little automated test farms
  - At present not very many (Denx, Consulko, Nvidia, Linaro, Samsung…?)
  - Faster release cycle, fewer regressions
- Driver-model migration complete
  - Deadlines in 2019 include MMC, USB, BLK, SATA, SPI, PCI, VIDEO
  - And perhaps Kconfig (~4500 completed so far)
  - Perhaps more driver-model support on the command line?
- More Linux code in U-Boot
- All new code comes with tests
  - At present this is true with driver model, filesystems, EFI, but is far from universal
- Reduced image size

Thank you for listening

- U-Boot is an open-source firmware project
- We are a friendly and welcoming bunch!
    - (if not, please let me know 👿)
- Go forth and U-Boot
    - Please send patches

- My details
    - Simon Glass
    - to: u-boot@lists.denx.de
    - cc: sjg@chromium.org

Google

Links (1)

- Driver model
  - https://elinux.org/images/c/c4/Order_at_last_-_U-Boot_driver_model_slides_%282%29.pdf
- Device tree
  - https://elinux.org/Device_Tree_Reference
- Kbuild
  - https://www.kernel.org/doc/Documentation/kbuild/makefiles.txt
- Kconfig
  - https://www.kernel.org/doc/Documentation/kbuild/kconfig-language.txt
- Custodian trees https://gitlab.denx.de/u-boot/custodians?page=1
- Android verified boot and OP-TEE
  - https://www.slideshare.net/GlobalLogicUkraine/uboot-and-android-verified-boot-20
  - http://connect.linaro.org.s3.amazonaws.com/hkg18/presentations/hkg18-124.pdf

Links (2)

- Verified boot
  - https://lwn.net/Articles/571031/
  - https://www.denx.de/wiki/pub/U-Boot/MiniSummitELCE2013/U-Boot_verified_RSA_boot_flow_on_arm_target.pdf
  - https://events.static.linuxfound.org/sites/events/files/slides/elce-2014.pdf
  - https://www.slideshare.net/GlobalLogicUkraine/uboot-and-android-verified-boot-20
  - https://ai.google/research/pubs/pub42038
- buildman - 'buildman -H'
- DFU
  - http://www.ti.com/lit/an/sprac65a/sprac65a.pdf
  - https://www.denx.de/wiki/pub/U-Boot/MiniSummitELCE2013/dfu_elce_u-boot.pdf
-

Links (3)

- Fastboot
  - https://www.denx.de/en/pub/Documents/Presentations/EWC2012_Roeder_Zundel_Fastboot.pdf
- Device-tree overlays
  - https://learn.adafruit.com/introduction-to-the-beaglebone-black-device-tree/device-tree-overlays
- Tizen SDWire https://wiki.tizen.org/SDWire
- Tbot https://github.com/hsdenx/tbot
- EFI
  - https://www.suse.com/media/article/UEFI_on_Top_of_U-Boot.pdf
  - http://events17.linuxfoundation.org/sites/events/files/slides/Marrying%20U-Boot%2C%20UEFI%20and%20grub.pdf
- OP-TEE https://www.op-tee.org/