Depthcharge

The ChromeOS bootloader

Design goals

- Boot ChromeOS
  - One focused goal
- Small
  - Fast load times
  - Easy to learn
  - Small attack surface
- Simple
  - Fast execution time
  - Easy to reason about
  - Better developer productivity
- Efficient

Google

Code flow

1. Depthcharge starts
    a. Enable consoles
    b. Initialize timestamps
    c. Call initialization callbacks (mostly just the board init function)
2. Call into verified boot (vboot) library.
    a. VbInit
        i. Initialize the library
    b. VbSelectFirmware
        i. Choose firmware version
    c. VbSelectAndLoadKernel
        i. Select a kernel
        ii. Developer mode
        iii. Recovery mode
    d. Calls back into depthcharge for platform specific functionality
3. Unpack, set up, and boot the chosen kernel

Google

libpayload

Library provided by coreboot for payloads, BSD licensed

Support code

- malloc/free, printf, "string" functions, rand, *delay, etc.
- LZMA compression
- PCI
- CBFS

Drivers

- USB 1, 2, 3, HID, mass storage, hubs
- cbmem/serial/framebuffer consoles
- CMOS
- PC keyboard

Google

Code structure

Header files alongside implementation

- board - board config settings, image layout
- src
  - arch - architecture specific code
  - base - generic support code
  - board - board specific setup code
  - boot - code for booting the kernel
  - drivers - drivers
  - image - code for interacting with the firmware image
  - net - network stack (netboot image)
  - netboot - DHCP, TFTP (netboot image)
  - vboot - interface to and from vboot

Drivers

- Generic interface structures
  - Structure of function pointer callbacks
  - First parameter is pointer to interface structure
  - Embedded in driver structure
  - Driver structure retrieved with container_of()
- Be as lazy as possible
  - Only do what you have to do when you have to do it
  - Don’t set up the device until someone tries to use it
- Driver structure returned by new_* function
  - Parameterizes drivers
  - Don’t touch the device, just describe it

Driver - code structure

- src/driver
  - [class]
    - [class].h - Generic interface structure, API functions
    - [class].c - Optional - Generic support code
    - Driver implementations

Google

Driver example - generic header

```
typedef struct PowerOps
{
        int (*cold_reboot)(struct PowerOps *me);
        int (*power_off)(struct PowerOps *me);
} PowerOps;

void power_set_ops(PowerOps *ops);

/* Cold reboot the machine */
int cold_reboot(void);

/* Power off the machine */
int power_off(void);
```

Google

Driver example - generic support code

```
static PowerOps *power_ops;

void power_set_ops(PowerOps *ops)
{
	die_if(power_ops, "%s: Power ops already set.\n", __func__);
	power_ops = ops;
}

int cold_reboot(void)
{
	die_if(!power_ops, "%s: No power ops set.\n", __func__);
	assert(power_ops->cold_reboot);

	if (run_cleanup_funcs(CleanupOnReboot))
		return -1;

	printf("Rebooting...\n");
	return power_ops->cold_reboot(power_ops);
}
```

Google

Driver example - driver header

```
typedef struct As3722Power
{
	PowerOps ops;
	I2cOps *bus;
	uint8_t chip;
} As3722Pmic;

As3722Pmic *new_as3722_pmic(I2cOps *bus, uint8_t chip);
```

Google

Driver example - driver implementation

```
static int as3722_cold_reboot(PowerOps *me)
{
	As3722Pmic *pmic = container_of(me, As3722Pmic, ops);
	as3722_set_bit(pmic->bus, pmic->chip, AS3722_RESET_CONTROL,
			AS3722_RESET_CONTROL_FORCE_RESET);
	halt();
}

static int as3722_power_off(PowerOps *me)
{
	As3722Pmic *pmic = container_of(me, As3722Pmic, ops);
	as3722_set_bit(pmic->bus, pmic->chip, AS3722_RESET_CONTROL,
			AS3722_RESET_CONTROL_POWER_OFF);
	halt();
}

As3722Pmic *new_as3722_pmic(I2cOps *bus, uint8_t chip)
{
	As3722Pmic *pmic = xzalloc(sizeof(*pmic));
	pmic->ops.cold_reboot = &as3722_cold_reboot;
	pmic->ops.power_off = &as3722_power_off;
	pmic->bus = bus;
	pmic->chip = chip;
	return pmic;
}
```

Google

Board setup function

```
static int board_setup(void)
{
	sysinfo_install_flags();

	PchGpio *ec_in_rw = new_pantherpoint_gpio_input(0, 21);
	flag_install(FLAG_ECINRW, &ec_in_rw->ops);

	CrosEcLpcBus *cros_ec_lpc_bus = new_cros_ec_lpc_bus();
	cros_ec_set_bus(&cros_ec_lpc_bus->ops);

	flash_set_ops(&new_mem_mapped_flash(0xff800000, 0x800000)->ops);

	sound_set_ops(&new_hda_codec()->ops);

	AhciCtrlr *ahci = new_ahci_ctrlr(PCI_DEV(0, 31, 2));
	list_insert_after(&ahci->ctrlr.list_node, &fixed_block_dev_controllers);

	power_set_ops(&pch_power_ops);

	tpm_set_ops(&new_lpc_tpm((void *)(uintptr_t)0xfed40000)->ops);

	return 0;
}
```

Google

Configuration

Configured using kconfig

All settings are off/inert by default

Override defaults in board/[board name]/defconfig

Google

Configuration example

```
# Arch
CONFIG_ARCH_X86=y

# Board
CONFIG_BOARD="link"

# Image
CONFIG_FMAP_OFFSET=0x610000

# Vboot
CONFIG_EC_SOFTWARE_SYNC=y
CONFIG_OPROM_MATTERS=y
CONFIG_RO_NORMAL_SUPPORT=y
CONFIG_VIRTUAL_DEV_SWITCH=y

CONFIG_CROSSYSTEM_ACPI=y
CONFIG_NV_STORAGE_CMOS=y

# Kernel format
CONFIG_KERNEL_ZIMAGE=y
```

```
# Drivers
CONFIG_DRIVER_AHCI=y
CONFIG_DRIVER_EC_CROS=y
CONFIG_DRIVER_EC_CROS_LPC=y
CONFIG_DRIVER_FLASH_MEMMAPPED=y
CONFIG_DRIVER_GPIO_PANTHERPOINT=y
CONFIG_DRIVER_INPUT_PS2=y
CONFIG_DRIVER_INPUT_USB=y
CONFIG_DRIVER_NET_ASIX=y
CONFIG_DRIVER_POWER_PCH=y
CONFIG_DRIVER_SOUND_HDA=y
CONFIG_DRIVER_TPM_LPC=y
```

Google

Support code - src/base

init_funcs
: Called when depthcharge starts. Don't touch hardware

cleanup_funcs
: Functions called when depthcharge exits

list
: Linked list implementation

container_of
: Like the one in the kernel

device_tree
: Code for manipulating the kernel's device tree on ARM

timestamp
: Timestamps for performance measurement

Google

New board steps

- Add fmap.dts to describe the image layout
- Add src/board directory, wire it into kconfig
- Add defconfig and enable the required settings/drivers
- Add and enable any missing drivers
- Write a setup_board function to describe board hardware