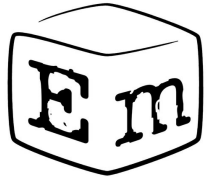


Опыт создания операционной платформы на основе существующих проектов с открытым кодом

Антон Бондарев

OS Day, 24 мая 2017

Система vs. Платформа

Ядро́ (kernel) — центральная часть операционной системы (ОС), обеспечивающая приложениям координированный доступ к ресурсам компьютера, таким как процессорное время, память, внешнее аппаратное обеспечение, внешнее устройство ввода и вывода информации.

Система vs. Платформа

Операцио́нная систе́ма, сокр. **ОС** (англ. *operating system, OS*) — комплекс взаимосвязанных программ, предназначенных для управления ресурсами компьютера и организации взаимодействия с пользователем.

Система vs. Платформа

Операционная платформа — комплекс программ, предназначенных для управления ресурсами компьютера, организации взаимодействия с пользователем и другими компьютерами, взаимодействия между программами, а также разработки и отладки программ.

Стадии развития проекта Embox

Стадия	**Драйвера**	**Приложения**	**API**
Маленький загрузчик	Напрямую	Напрямую	Напрямую
Студенческий проект	Вынесли интерфейс	Вызов драйверов напрямую	Какая-то своя библиотека
Первые промышленные применения	Разработка драйверов вручную	Разработка вручную через POSIX	Реализация стандартной библиотеки
Используемый проект	Попытка использовать сторонние проекты	По возможности используем сторонние проекты	Попытки заимствовать стандартную библиотеку

Интерфейс

Было:

```
EMBOX_CMD(bt_main);
static int bt_main(int argc, char **argv) {
```

Стало:

```
int main(int argc, char **argv) {
```

Система сборки

```
@AutoCmd
@Cmd(name = "df",
    help = "report file system disk space usage",
    man = '''
      NAME
           df - report file system disk space usage
      SYNOPSIS
           cd [dir]
      ...
    ''')
module df {
    source "df.c"
    depends embox.compat.posix.fs.statvfs
}
```

Система сборки (ExtBuild)

```
PKG_NAME := nano
PKG_VER  := 2.2.6
PKG_SOURCES := http://www.nano-editor.org/dist/v2.2/$(PKG_NAME)-$(PKG_VER).tar.gz
PKG_MD5   := 03233ae480689a008eb98feb1b599807
PKG_PATCHES := pkg_patch.txt
include $(EXTBLD_LIB)
$(CONFIGURE) :
    export EMBOX_GCC_LINK=full; \
    cd $(PKG_SOURCE_DIR) && ( \
      ./configure --host=$(AUTOCONF_TARGET_TRIPLET) \
           --target=$(AUTOCONF_TARGET_TRIPLET) \
     ...
    )
    touch $@
```

Система сборки (ExtBuild)

```
$(BUILD) :
    cd $(PKG_SOURCE_DIR) && ( \
    $(MAKE) MAKEFLAGS='$(EMBOX_IMPORTED_MAKEFLAGS)'; \
    )
    touch $@

$(INSTALL) :
    cp $(PKG_SOURCE_DIR)/src/nano $(PKG_INSTALL_DIR)/nano.o
    touch $@
```

Система сборки (ExtBuild)

```
@App
@AutoCmd
@Build(stage=2,script="$(EXTERNAL_MAKE)")
@Cmd(name = "nano",
    help = "Text editor",
    man = '''
      NAME
           nano - Nano's ANOther editor
...
    ''')
module nano {
    source "^BUILD/extbld/^MOD_PATH/install/nano.o"
    @NoRuntime depends embox.compat.posix.regex
    depends embox.compat.posix.LibCurses
}
```

Сторонние BSP

```
PKG_NAME := stm32f7

PKG_SOURCES :=
https://www.dropbox.com/s/2u5cuh3gmrv375m/stm32cubef7.zip?dl=0
PKG_ARCHIVE_NAME :=stm32cubef7.zip
PKG_MD5   := 15f04dadded7602d1406cc0623ba0e22

include $(EXTBLD_LIB)
```

Сторонние BSP

```
package third_party.bsp.stmf7cube


@Build(stage=1,script="$(EXTERNAL_MAKE) download extract patch")
@BuildArtifactPath(cppflags="-DSTM32F746xx
-I$(ROOT_DIR)/third-party/bsp/stmf7cube/ $(addprefix
-I$(EXTERNAL_BUILD_DIR)/third_party/bsp/stmf7cube/core/STM32Cube_FW
_F7_V1.5.0/,Drivers/STM32F7xx_HAL_Driver/Inc
Drivers/BSP/STM32746G-Discovery/
Drivers/CMSIS/Device/ST/STM32F7xx/Include Drivers/CMSIS/Include
Projects/STM32746G-Discovery/Examples/BSP/Inc)")
static module core extends third_party.bsp.st_bsp_api {
```

Сторонние BSP

```
@BuildDepends(third_party.bsp.stmf7cube.core)
module stm32f7cube_eth {
    source "stm32f7cube_eth.c"
    source "stm32cube_eth.c"
    @IncludeExport(path="drivers/net",
                   target_name="stm32cube_eth.h")
    source"stm32f7cube_eth.h"
    option number log_level=0
    depends embox.net.skbuff
    ...
}
```

Сторонние BSP

```
#include <embox/unit.h>
EMBOX_UNIT_INIT(stm32eth_init);

static int stm32eth_init(void) {
    int res;
    struct net_device *nic;
    res = irq_attach(nic->irq, stm32eth_interrupt, 0,
                     stm32eth_netdev, "");
    ...
    return inetdev_register_dev(nic);
}
```

Сторонние BSP

```
static irq_return_t stm32eth_interrupt(unsigned int irq_num,
    void *dev_id) {
    struct net_device *nic_p = dev_id;
    struct sk_buff *skb;
    ETH_HandleTypeDef *heth = &stm32_eth_handler;
    ...
    return IRQ_HANDLED;
}
```

Сторонние BSP

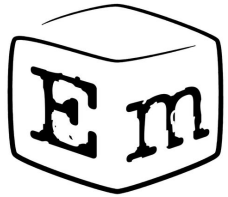
```
static int stm32eth_xmit(struct net_device *dev,
                         struct sk_buff *skb);
static int stm32eth_open(struct net_device *dev);
static int stm32eth_set_mac(struct net_device *dev,
                            const void *addr);

static const struct net_driver stm32eth_ops = {
    .xmit = stm32eth_xmit,
    .start = stm32eth_open,
    .set_macaddr = stm32eth_set_mac,
};
```

Выводы

Использование СПО:

- Уменьшает количество ошибок
- Уменьшает время разработки
- Уменьшает стоимость поддержки

Контакты



Страница проекта

http://emboxing.ru

Репозиторий проекта

https://github.com/embox/

Антон Бондарев

anton.bondarev2310@gmail.com