

2024.04.15-2024.04.21-work-log

工作进展

本阶段完成的任务有：将前三周编写的rtsmart-std、marco_main等库进行测试。编译了一个简单的rust程序，观察代码能否成功编译。并在qemu上测试一下，观察能否正常运行。

测试项目代码

Cargo.toml

```
[package]
name = "hello"
version = "0.1.0"
edition = "2021"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/reference/manifest.html
[dependencies]
marco_main = { path = "../marco_main" }
rtsmart-std = { path = "../rtsmart-std" }
```

需要引用原先编写的rtsmart-std和marco_main库作为依赖。

同时在.cargo文件夹中添加一个config.toml文件，在里面添加如下内容：

```
[build]
target = "aarch64-unknown-rtsmart"

[target.aarch64-unknown-rtsmart]
linker = "aarch64-linux-musleabi-gcc"
```

指定编译目标和链接器，这样就不需要在编译的时候指定--target=aarch64-unknown-rtsmart参数，直接cargo xbuild即可

main.rs

```
#![no_std]
#![no_main]

use marco_main::marco_main_use;
use rtsmart_std::println;

#[marco_main_use(appname = "rust_hello", desc = "Rust example app.")]
fn main(_param: Param) {
    println!("hello world");
}
```

仍需加上#![no_std]和#![no_main]标注，否则编译器会自动去寻找标准库，而我们并未对Rust编译器内的标准库添加aarch64-unknown-rtsmart平台的支持，因此必然会报错。

然后需要使用`marco_main_use`这一过程宏。除此之外，与正常使用Rust标准库类似，调用`println!()`宏对内容进行输出

编译过程

宏展开

首先我们可以观察一下宏展开后的代码是什么情况，因此我们下载`cargo-expand`工具

```
cargo install cargo-expand
```

然后通过如下命令观察宏展开后的Rust代码

```
cargo expand -Zbuild-std=core,alloc
```

运行后命令行窗口输出如下内容：

```
#![feature(prelude_import)]
#![no_std]
#![no_main]
#[prelude_import]
use core::prelude::rust_2021::*;
#[macro_use]
extern crate core;
extern crate compiler_builtins as _;
use marco_main::marco_main_use;
use rtsmart_std::println;
#[no_mangle]
pub extern "C" fn main(_argc: isize, _argv: *const *const u8) -> usize {
    {
        {
            ::rtsmart_std::out::_print(format_args!("hello world\n"));
        };
    }
    0
}
```

这是声明宏和过程宏同时展开后的结果，与我们之前编写的测试libc的应用程序代码相似，都是通过将用户编写的`main`函数转换为以C ABI 调用约定为基础的 `main` 函数，作为程序的入口点，来进行编译运行的。


编译

使用如下命令编译应用程序


```
cargo xbuild -Zbuild-std=core,alloc
```


由于有了`config.toml`文件中的配置，因此不需要在命令中指定`target`


编译成功后在`target/aarch64-unknown-rtsmart/debug`里能找到编译好的应用程序`hello`


✓  .cargo


✓  src


 main.rs


✓  target


✓  aarch64-unknown-rtsmart


✓  debug

>  .fingerprint


>  build

>  deps

 examples


>  incremental


≡ .cargo-lock


 hello

≡ hello.d


JSP CACHEDIR.TAG


>  debug


>  sysroot

 .rustc_info.json

JSP CACHEDIR.TAG

 .gitignore

 Cargo.lock

 Cargo.toml

之后通过2024.03.17-2024.03.23-work-log中描述的方法，将应用程序通过文件系统转移到qemu虚拟机中。

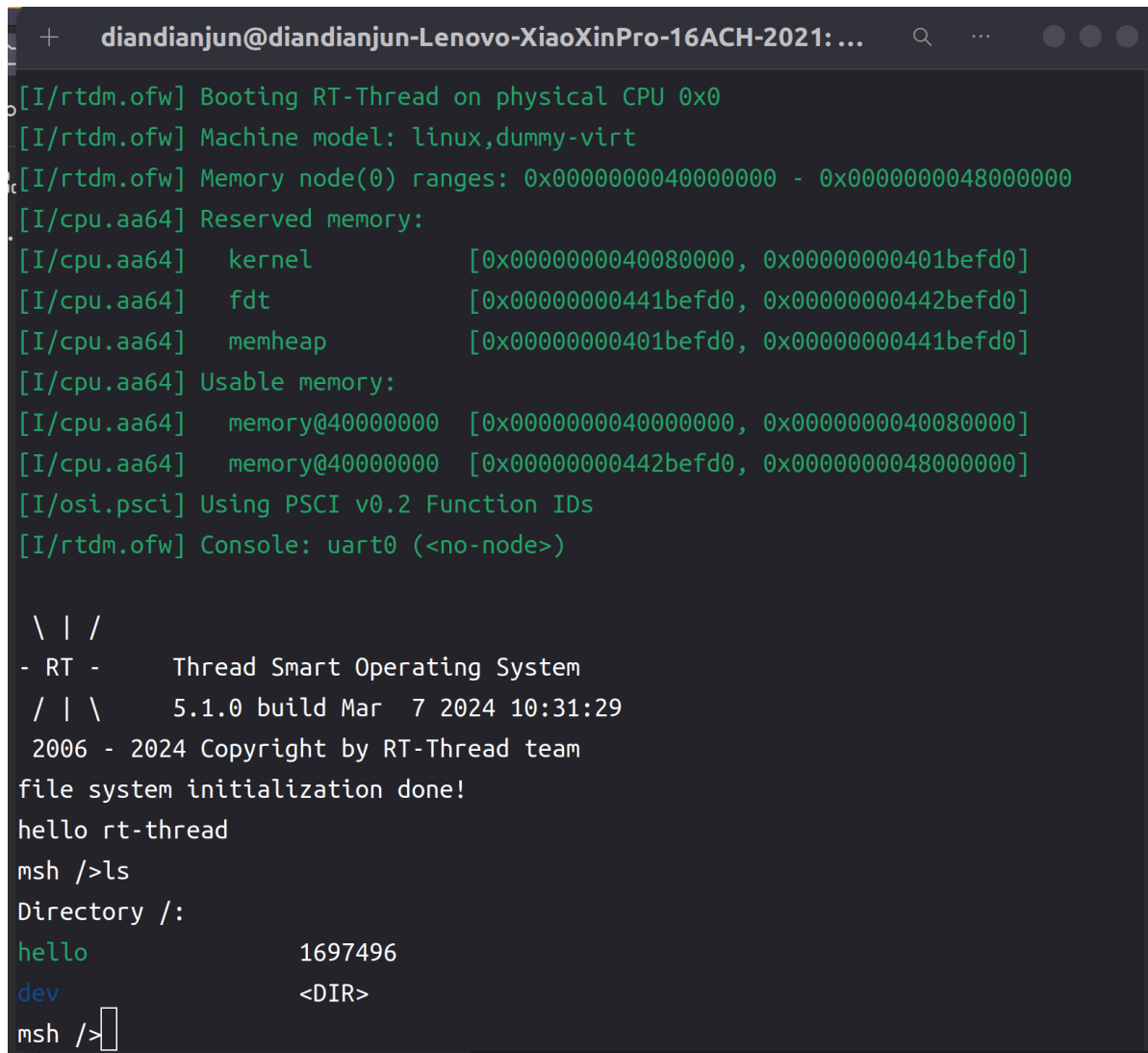
运行

```
./qemu.sh
```

运行虚拟机，使用命令

```
ls
```

可以观察到新放入的应用程序hello



```
diandianjun@diandianjun-Lenovo-XiaoXinPro-16ACH-2021: ...  
[I/rtdm.ofw] Booting RT-Thread on physical CPU 0x0  
[I/rtdm.ofw] Machine model: linux,dummy-virt  
[I/rtdm.ofw] Memory node(0) ranges: 0x0000000040000000 - 0x0000000048000000  
[I/cpu.aa64] Reserved memory:  
[I/cpu.aa64]   kernel      [0x0000000040080000, 0x00000000401befd0]  
[I/cpu.aa64]   fdt        [0x00000000441befd0, 0x00000000442befd0]  
[I/cpu.aa64]   memheap    [0x00000000401befd0, 0x00000000441befd0]  
[I/cpu.aa64] Usable memory:  
[I/cpu.aa64]   memory@40000000 [0x0000000040000000, 0x0000000040080000]  
[I/cpu.aa64]   memory@40000000 [0x00000000442befd0, 0x0000000048000000]  
[I/osi.psci] Using PSCI v0.2 Function IDs  
[I/rtdm.ofw] Console: uart0 (<no-node>)  
  
 \ | /  
- RT -   Thread Smart Operating System  
 / | \   5.1.0 build Mar 7 2024 10:31:29  
2006 - 2024 Copyright by RT-Thread team  
file system initialization done!  
hello rt-thread  
msh />ls  
Directory /:  
hello          1697496  
dev            <DIR>  
msh />
```

由于本次编译将rtsmart-std和marco_main等新编写的库都通过静态链接的方式编译到应用程序中了，因此本次的应用程序大小为1.7MB，比上次测试libc的hello应用程序的170KB大了不少

运行应用程序

```
./hello
```

```
+ diandianjun@diandianjun-Lenovo-XiaoXinPro-16ACH-2021: ...
[I/rtdm.ofw] Memory node(0) ranges: 0x0000000040000000 - 0x0000000048000000
[I/cpu.aa64] Reserved memory:
[I/cpu.aa64]   kernel      [0x0000000040080000, 0x00000000401befd0]
[I/cpu.aa64]   fdt        [0x00000000441befd0, 0x00000000442befd0]
[I/cpu.aa64]   memheap    [0x00000000401befd0, 0x00000000441befd0]
[I/cpu.aa64] Usable memory:
[I/cpu.aa64]   memory@40000000 [0x0000000040000000, 0x0000000040080000]
[I/cpu.aa64]   memory@40000000 [0x00000000442befd0, 0x0000000048000000]
[I/osi.psci] Using PSCI v0.2 Function IDs
[I/rtdm.ofw] Console: uart0 (<no-node>)

\ | /
- RT -   Thread Smart Operating System
/ | \    5.1.0 build Mar  7 2024 10:31:29
2006 - 2024 Copyright by RT-Thread team
file system initialization done!
hello rt-thread
msh />ls
Directory /:
hello          1697496
dev            <DIR>
msh />./hello
msh />hello world
█
```

观察到，能够正常输出hello world

总结

本周的工作进展主要是把前三周写的东西进行测试，验证了我们这种，新编写的仿制标准库+属性宏改写main函数的方式是可行的，能够正常调用libc，编译运行在 `aarch64-unknown-rtsmart` 平台上。

成功验证了我们的思路的正确性后，未来我们将继续完善当前标准库，提供更多的标准库函数给用户使用。