附录 C

RT-Thread/lumit4510 on SkyEye 说明

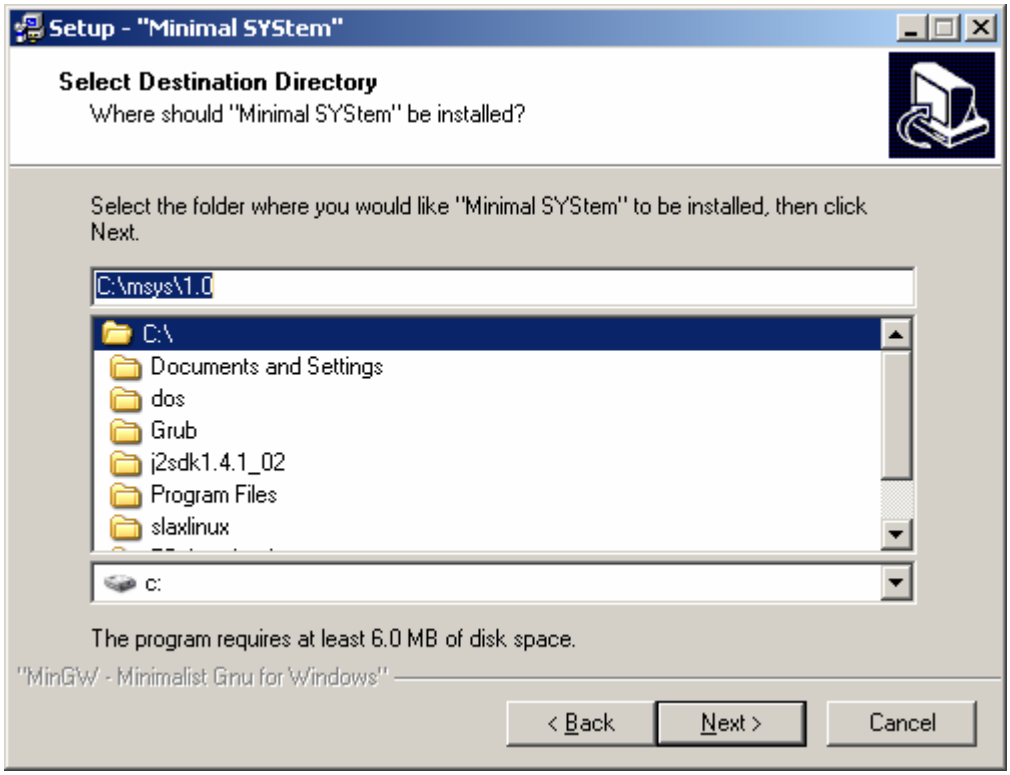
本文将说明如何在 Win32 环境下搭建 RT-Thread 的开发环境及如何使用 skyeye（http://www.skyeye.org）来模拟运行/调试 RT-Thread。

Win32 开发环境的搭建

安装 MinGW （Msys）

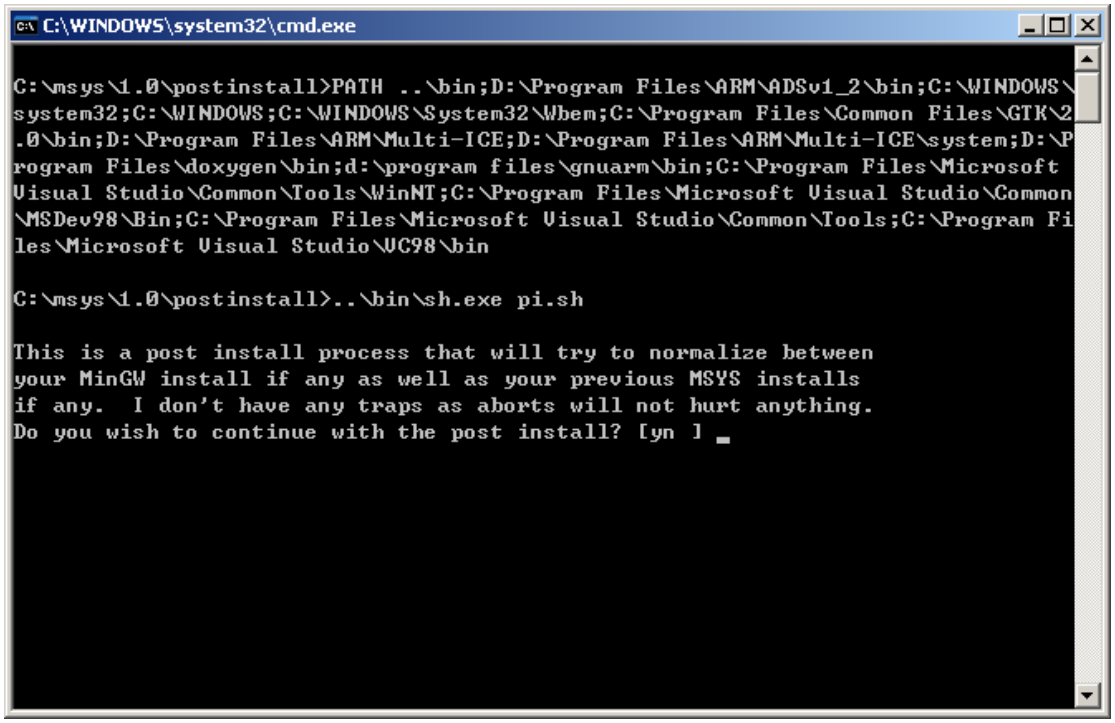
在 Windows 平台上为了获得 GNU 的环境，需要安装额外的软件，为了简化安装，这里选择了 Windows 平台下最小的 GNU 环境：MSys，先请安装 Msys，下载地址：http://prdownloads.sf.net/mingw/MSYS-1.0.11-2004.04.30-1.exe?download (请将这个地址复制到 IE 地址栏中，然后开启下载)

安装截图如下：

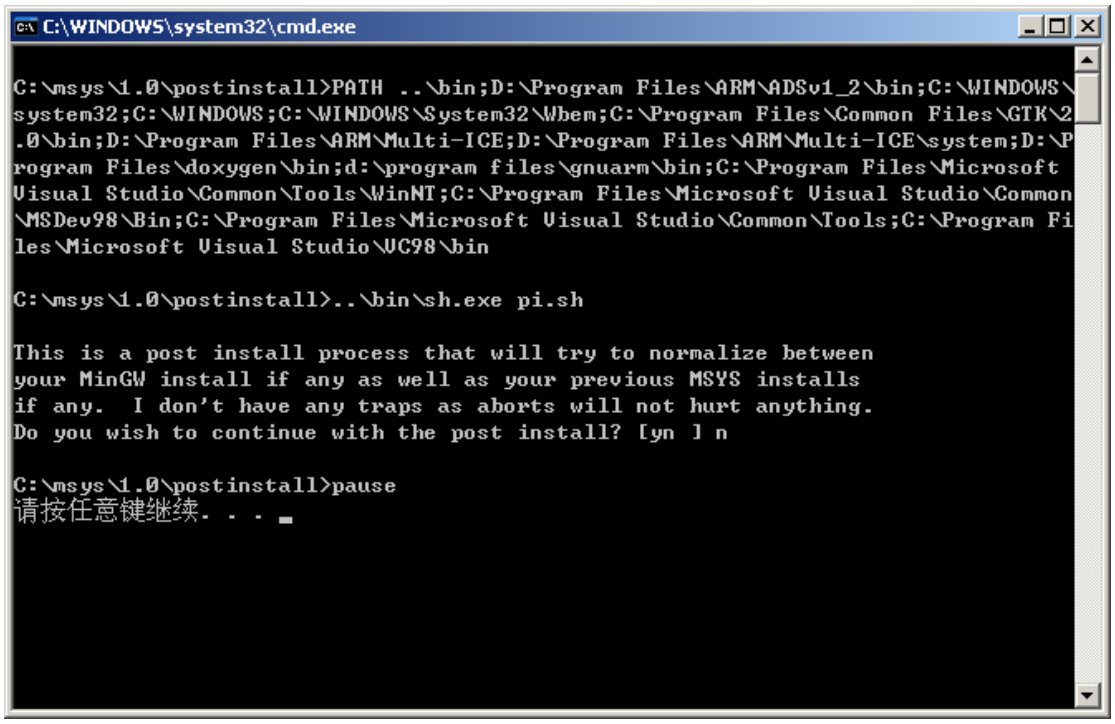


对于初次使用 Msys，建议采用默认选择，安装在 C 盘。

安装过程中会跳出一个命令行窗口，如下：



请直接选择 n



然后随意按一键继续。

MSys 就算安装完成了。

它会在开始菜单中生成一个菜单：

运行 msys，会出现类似 linux rxvt 的终端窗口：

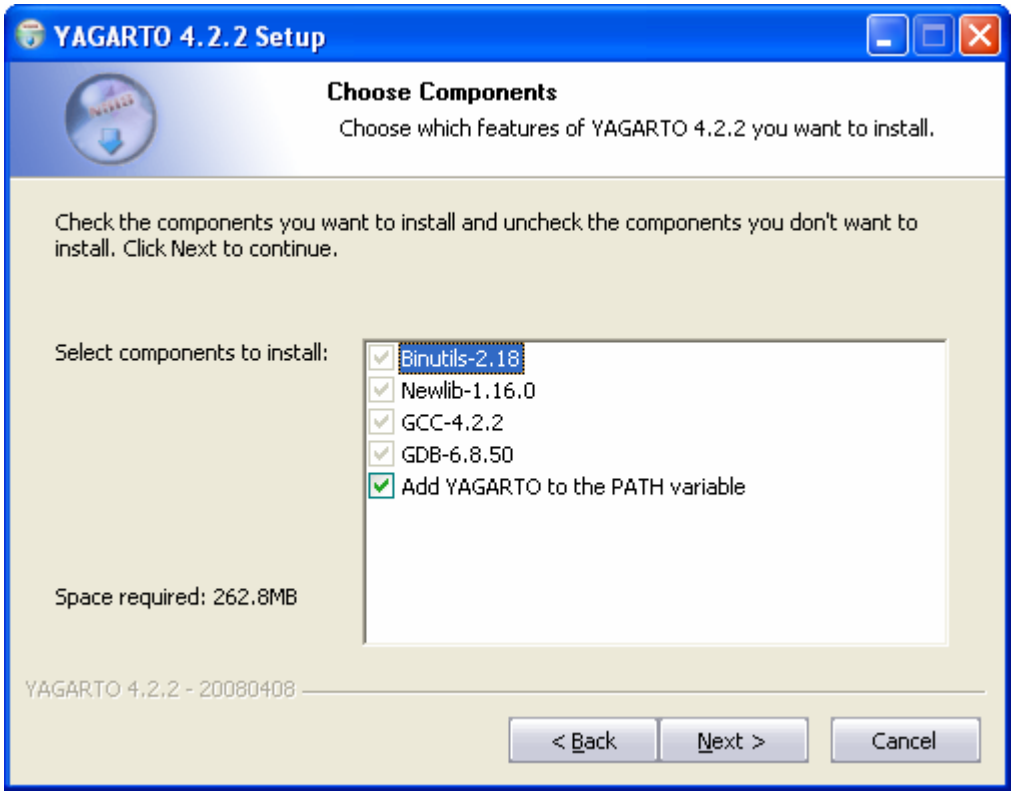
安装 Python

目前 RT-Thread 支持多种平台以及 CPU，针对不同的平台类型，需要用 Python 来生成对应的编译配置文件。到 http://www.python.org/download/ 下载最新版的 Python Windows installer。

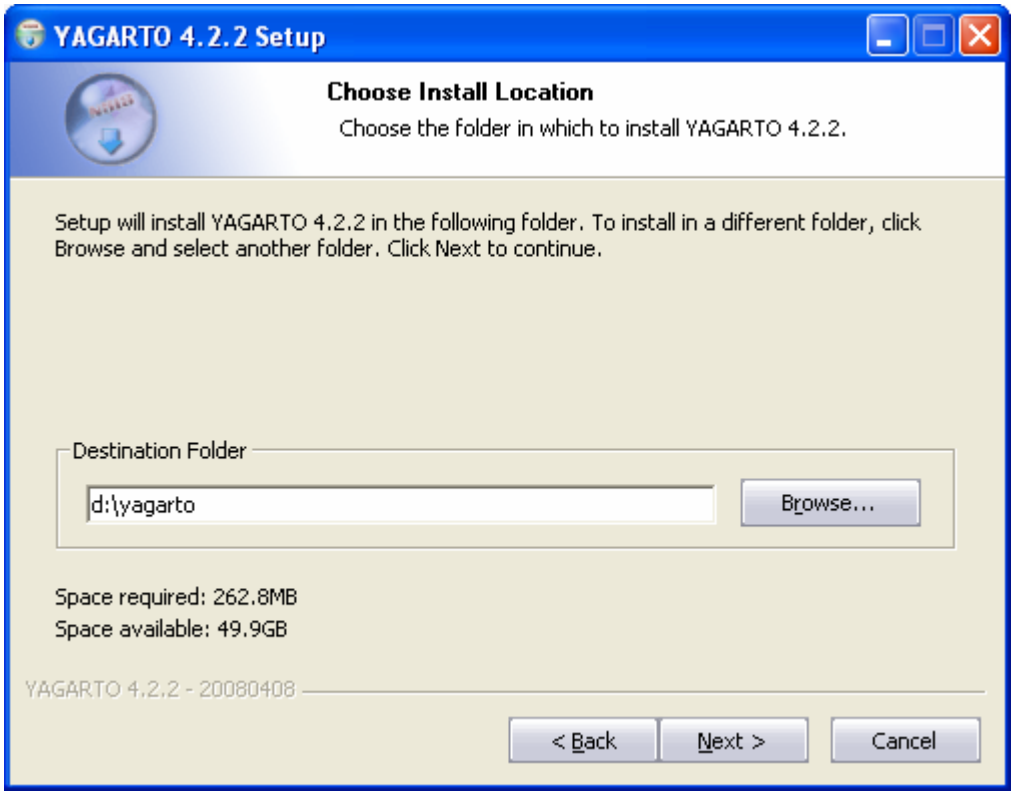
安装 ARM 交叉编译器

ARM 交叉编译器是配合 Msys 一起使用的，请到如下网址下载：
http://downloads.sourceforge.net/yagarto/yagarto-bu-2.18_gcc-4.3.2-c-c%2B%2B_nl-1.16.0_gi-6.8.50_20080928.exe

安装截图：



选择安装路径：



安装完成后，在 Msys 终端窗口中输入/d/yagarto/bin/arm-elf-gcc –v 应该会有如下输出：

```
MINGW32:~

bernard@rt-thread.org ~
$ /d/yagarto/bin/arm-elf-gcc -v
Using built-in specs.
Target: arm-elf
Configured with: ../gcc-4.2.2/configure --target=arm-elf --prefix=/home/yagarto/
install --disable-nls --disable-shared --disable-threads --with-gcc --with-gnu-l
d --with-gnu-as --with-dwarf2 --enable-languages=c,c++ --enable-interwork --enab
le-multilib --with-newlib --with-headers=../newlib-1.16.0/newlib/libc/include --
disable-libssp --disable-libstdcxx-pch --disable-libmudflap --disable-libgomp -v
Thread model: single
gcc version 4.2.2

bernard@rt-thread.org ~
$ 
```

准备编译运行 RT-Thread

请先下载 RT-Thread v0.2.3 源代码，下载地址：
http://www.rt-thread.org/rt-thread/rtt-0.2.3.tar.bz2

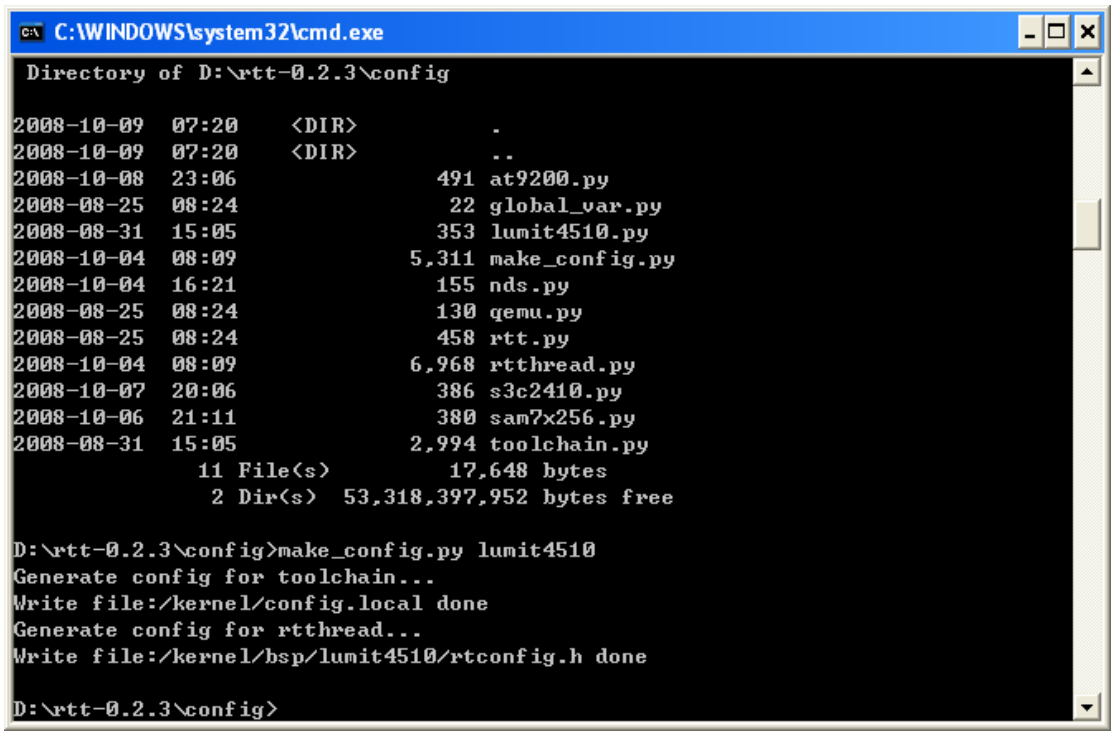
放置到 D:盘根目录下，在 Msys 终端中请运行如下命令行来解压缩源代码：
cd /d/
tar jxvf rtt-0.2.3.tar.bz2

```
MINGW32:/d

bernard@rt-thread.org ~
$ cd /d/

bernard@rt-thread.org /d
$ tar jxvf rtt-0.2.3.tar.bz2
rtt-0.2.3/
rtt-0.2.3/AUTHORS
rtt-0.2.3/ChangeLog
rtt-0.2.3/config/
rtt-0.2.3/config/at9200.py
rtt-0.2.3/config/global_var.py
rtt-0.2.3/config/lumit4510.py
rtt-0.2.3/config/make_config.py
rtt-0.2.3/config/nds.py
rtt-0.2.3/config/qemu.py
rtt-0.2.3/config/rtt.py
rtt-0.2.3/config/rtthread.py
rtt-0.2.3/config/s3c2410.py
rtt-0.2.3/config/sam7x256.py
rtt-0.2.3/config/toolchain.py
rtt-0.2.3/COPYING
rtt-0.2.3/kernel/
rtt-0.2.3/kernel/bsp/
rtt-0.2.3/kernel/bsp/AT9200/
```

编译 RT-Thread

预配置

打开 windows 命令行窗口，在 d:\rtt-0.2.3\config 目录下执行：
make_config.py lumit4510



至此，为 lumit4510 平台的预配置就完成了，接下来开始编译 RT-Thread。

编译

进入到 RT-Thraed 的目录，设置好 ARM 交叉编译器的路径，运行 make 即可：
export PATH=/d/yagarto/bin:$PATH
cd /d/rtt-0.2.3/
make

```
MINGW32:/d/rtt-0.2.3
bernard@rt-thread.org /d
$ export PATH=/d/yagarto/bin:$PATH

bernard@rt-thread.org /d
$ cd rtt-0.2.3

bernard@rt-thread.org /d/rtt-0.2.3
$ make
make -C kernel
make[1]: Entering directory `/d/rtt-0.2.3/kernel'
for dir in src libcpu finsh cplusplus bsp; do                          \
        make -C $dir  RT_USING_FINSH=1;                  \
done
make[2]: Entering directory `/d/rtt-0.2.3/kernel/src'
Makefile:22: .depend: No such file or directory
arm-elf-gcc -Wall -I../finsh  -I../include -I../bsp/lumit4510 -Wall -nostdinc -f
no-builtin -ggdb -M kservice.c clock.c object.c ipc.c timer.c irq.c scheduler.c
thread.c idle.c mempool.c mem.c slab.c device.c > .depend
make[2]: Leaving directory `/d/rtt-0.2.3/kernel/src'
make[2]: Entering directory `/d/rtt-0.2.3/kernel/src'
arm-elf-gcc -c -Wall -I../finsh  -I../include -I../bsp/lumit4510 -Wall -nostdinc
 -fno-builtin -ggdb -o kservice.o kservice.c
arm-elf-gcc -c -Wall -I../finsh  -I../include -I../bsp/lumit4510 -Wall -nostdinc
 -fno-builtin -ggdb -o clock.o clock.c
arm-elf-gcc -c -Wall -I../finsh  -I../include -I../bsp/lumit4510 -Wall -nostdinc
```

稍微等待几分钟，编译出来的 RT-Thread 操作系统映像文件将成功的产生。

```
MINGW32:/d/rtt-0.2.3
make[3]: Entering directory `/d/rtt-0.2.3/kernel/bsp/lumit4510'
arm-elf-gcc -c -Wall -I../../finsh  -I../../include -I../../bsp/lumit4510 -Wall
-nostdinc -fno-builtin -ggdb -I../../libcpu/arm/s3c4510 -o board.o board.c
arm-elf-gcc -c -Wall -I../../finsh  -I../../include -I../../bsp/lumit4510 -Wall
-nostdinc -fno-builtin -ggdb -I../../libcpu/arm/s3c4510 -o startup.o startup.c
arm-elf-ar r ../../lib/libbsp_lumit4510.a board.o startup.o
d:\yagarto\bin\arm-elf-ar.exe: creating ../../lib/libbsp_lumit4510.a
arm-elf-ranlib ../../lib/libbsp_lumit4510.a
arm-elf-gcc -c -Wall -I../../finsh  -I../../include -I../../bsp/lumit4510 -Wall
-nostdinc -fno-builtin -ggdb -I../../libcpu/arm/s3c4510 -o application.o applica
tion.c
arm-elf-ld -static -nostdlib -T ../../bsp/lumit4510/lumit4510_ram.lds -Ttext 0x8
000 ../../lib/start_s3c4510.o board.o startup.o application.o -nostdlib -L../../
lib --start-group -lcpu_s3c4510 -lkernel_arm -lbsp_lumit4510 -lfinsh_arm -lrtt++
_arm --end-group -o rtthread-lumit4510.elf
arm-elf-objcopy -O binary rtthread-lumit4510.elf rtthread-lumit4510.bin
arm-elf-size rtthread-lumit4510.elf
   text    data     bss     dec     hex filename
  79156     672   13192   93020   16b5c rtthread-lumit4510.elf
make[3]: Leaving directory `/d/rtt-0.2.3/kernel/bsp/lumit4510'
make[2]: Leaving directory `/d/rtt-0.2.3/kernel/bsp'
make[1]: Leaving directory `/d/rtt-0.2.3/kernel'

bernard@rt-thread.org /d/rtt-0.2.3
$
```

使用 Skyeye 运行 RT-Thread

在 RT-Thread-0.2.3 的发行包当中，已经附带了 skyeye 的 windows 可执行文件：
执行 d:\rtt-0.2.3\tools 目录下的 run-lumit4510.bat：

```
C:\WINDOWS\system32\cmd.exe
uart_mod:0, desc_in:, desc_out:, converter:
SKYEYE: use arm7100 mmu ops
exec file "..\kernel\bsp\lumit4510\rtthread-lumit4510.elf"'s format is elf32-lit
tle.
load section .text: addr = 0x00008000  size = 0x00012bd0.
load section .rodata: addr = 0x0001abd0  size = 0x00000964.
load section .data: addr = 0x0001b534  size = 0x000002a0.
not load section .bss: addr = 0x0001b7d4  size = 0x00003388 .
not load section .comment: addr = 0x00000000  size = 0x00000240 .
not load section .debug_abbrev: addr = 0x00000000  size = 0x00002553 .
not load section .debug_info: addr = 0x00000000  size = 0x0000ded4 .
not load section .debug_line: addr = 0x00000000  size = 0x0000205f .
not load section .debug_pubnames: addr = 0x00000000  size = 0x000018c8 .
not load section .debug_aranges: addr = 0x00000000  size = 0x00000440 .
not load section .ARM.attributes: addr = 0x00000000  size = 0x00000010 .
not load section .debug_frame: addr = 0x00000000  size = 0x00002b34 .
not load section .debug_loc: addr = 0x00000000  size = 0x000035fa .
not load section .debug_str: addr = 0x00000000  size = 0x0000034a .
not load section .debug_ranges: addr = 0x00000000  size = 0x00000018 .
start addr is set to 0x00008000 by exec file.
 \ | /
- RT - Thread Operating System
 / | \ 0.2.3 build Oct  9 2008
 2006, 2007, 2008 Copyright by rt-thread team
finsh>>
```

现在 RT-Thread 就启动起来了

（执行的是 d:\rtt-0.2.3\kernel\bsp\lumit4510\rtthread-lumit4510.elf）。

由于默认 RT-Thread/lumit4510 默认会启动 finsh shell，所以可以在上面运行一些 finsh 内建的函数：

```
C:\WINDOWS\system32\cmd.exe
load section .rodata: addr = 0x0001abd0  size = 0x00000964.
load section .data: addr = 0x0001b534  size = 0x000002a0.
not load section .bss: addr = 0x0001b7d4  size = 0x00003388 .
not load section .comment: addr = 0x00000000  size = 0x00000240 .
not load section .debug_abbrev: addr = 0x00000000  size = 0x00002553 .
not load section .debug_info: addr = 0x00000000  size = 0x0000ded4 .
not load section .debug_line: addr = 0x00000000  size = 0x0000205f .
not load section .debug_pubnames: addr = 0x00000000  size = 0x000018c8 .
not load section .debug_aranges: addr = 0x00000000  size = 0x00000440 .
not load section .ARM.attributes: addr = 0x00000000  size = 0x00000010 .
not load section .debug_frame: addr = 0x00000000  size = 0x00002b34 .
not load section .debug_loc: addr = 0x00000000  size = 0x000035fa .
not load section .debug_str: addr = 0x00000000  size = 0x0000034a .
not load section .debug_ranges: addr = 0x00000000  size = 0x00000018 .
start addr is set to 0x00008000 by exec file.
 \ | /
- RT - Thread Operating System
 / | \ 0.2.3 build Oct  9 2008
 2006, 2007, 2008 Copyright by rt-thread team
 thread   pri  status      sp     stack size max used   left tick  error
-------- ---- ------- ---------- ---------- ---------- ---------- ---
tidle    0xff ready   0x00000074 0x00000100 0x00000074 0x00000005 000
tshell   0x14 ready   0x0000024c 0x00000800 0x00000418 0x00000063 000
        0, 0x0000
finsh>>
```

详细的命令，请参考 RT-Thread Shell 文档。