第 12 章

FinSH Shell 系统

RT-Thread 的 shell 系统——finsh，提供了一套供用户在命令行操作的接口，主要用于调试、查看系统信息。finsh 被设计成一个不同于传统命令行的解释器，由于很多嵌入式系统都是采用 C 语言来编写，所以 finsh 的输入对象也类似于 C 语言表达式的风格：它能够解析执行大部分 C 语言的表达式，也能够使用类似于 C 语言的函数调用方式访问系统中的函数及全局变量，此外它也能够通过命令行方式创建变量。

12.1 基本数据类型

finsh 支持基本的 C 语言数据类型，包括：
`void` – 空数据格式，只用于创建指针变量
`char, unsigned char` –（带符号）字符型数据
`short, unsigned int` – (带符号)整数型数据
`long, unsigned long` –(带符号)长整型数据
此外，finsh 也支持指针类型（`void *`或 `int *`等声明方式），如果指针做为函数指针类型调用，将自动按照函数方式执行。

12.2 内置命令

finsh 中内建了一些命令函数，可以在命令行中调用：
`list()`
显示系统中存在的命令及变量，在 lumit4510 平台上执行结果如下

```
--Function List:
hello
version
list
list_thread
list_sem
list_mutex
list_event
list_mb
list_mq
```

```
list_memp
list_timer
--Variable List:
```

`--Function List` 表示的是函数列表；
`--Variable List` 表示的是变量列表。

12.3 RT-Thread 内置命令

针对 RT-Thread RTOS，finsh 也提供了一些基本的函数命令：

- list_thread()

列表显示当前系统中线程状态，lumit4510 显示结果如下：

```
thread  pri  status      sp     stack size max used   left tick  error
-------- ---- ------- ---------- ---------- ---------- ---------- ---
tidle    0xff ready   0x00000074 0x00000100 0x00000074 0x0000003b 000
tshell   0x14 ready   0x0000024c 0x00000800 0x00000418 0x00000064 000
```

`thread` 字段表示线程的名称
`pri` 字段表示线程的优先级
`status` 字段表示线程当前的状态
`sp` 字段表示线程当前的栈位置
`stack size` 字段表示线程的栈大小
`max used` 字段表示线程历史上使用的最大栈位置
`left tick` 字段表示线程剩余的运行节拍数
`error` 字段表示线程的错误号

- list_sem()

列表显示系统中信号量状态，lumit4510 显示结果如下：

```
semaphore v   suspend thread
--------  --- --------------
uart      000 0
```

`semaphore` 字段表示信号量的名称
`v` 字段表示信号量的当前值
`suspend thread` 字段表示等在这个信号量上的线程数目

- list_mb()

列表显示系统中信箱状态，lumit4510 显示结果如下：

```
mailbox  entry size suspend thread
-------- ----  ---- --------------
```

`mailbox` 字段表示信箱的名称
`entry` 字段表示信箱中包含的信件数目
`size` 字段表示信箱能够容纳的最大信件数目
`suspend thread` 字段表示等在这个信箱上的线程数目

- list_mq()

列表显示系统中消息队列状态，lumit4510 显示结果如下：

```
msgqueue entry suspend thread
-------- ----  --------------
```

semaphore 字段表示消息队列的名称
entry 字段表示消息队列中当前包含的消息数目
size 字段表示消息队列能够容纳的最大消息数目
suspend thread 字段表示等在这个消息队列上的线程数目

- list_event()

列表显示系统中事件状态，lumit4510 显示结果如下：

```
event    set        suspend thread
-------- ---------- --------------
```

event 字段表示事件的名称
set 字段表示事件的值
suspend thread 字段表示等在这个事件上的线程数目

- list_timer()

列表显示系统中定时器状态，lumit4510 显示结果如下：

```
timer    periodic   timeout    flag
-------- ---------- ---------- -----------
tidle    0x00000000 0x00000000 deactivated
tshell   0x00000000 0x00000000 deactivated
current tick:0x00000d7e
```

timer 字段表示定时器的名称
periodic 字段表示定时器是否是周期性的
timeout 字段表示定时器超时时的节拍数
flag 字段表示定时器的状态，activated 表示活动的，deactivated 表示不活动的
current tick 表示当前系统的节拍

12.4 应用程序接口

finsh 的应用程序接口提供了上层注册函数或变量的接口，使用时应作如下头文件包含：

```
#include <finsh.h>
```

- 原型：void finsh_syscall_append(const char* name, syscall_func func)

在 finsh 中添加一个函数。
name – 函数在 finsh shell 中访问的名称
func – 函数的地址

- 原型：void finsh_sysvar_append(const char* name, u_char type, void* addr)

在 finsh 中添加一个变量。
name – 变量在 finsh shell 中访问的名称
type – 数据类型，由枚举类型 finsh_type 给出 finsh 支持的数据类型：

```
enum finsh_type {
```

```
    finsh_type_unknown = 0,
    finsh_type_void,        /** void            */
    finsh_type_voidp,       /** void pointer    */
    finsh_type_char,        /** char            */
    finsh_type_uchar,       /** unsigned char   */
    finsh_type_charp,       /** char pointer    */
    finsh_type_short,       /** short           */
    finsh_type_ushort,      /** unsigned short  */
    finsh_type_shortp,      /** short pointer   */
    finsh_type_int,         /** int             */
    finsh_type_uint,        /** unsigned int    */
    finsh_type_intp,        /** int pointer     */
    finsh_type_long,        /** long            */
    finsh_type_ulong,       /** unsigned long   */
    finsh_type_longp        /** long pointer    */
};
```

addr – 变量的地址

12.5 移植

由于 finsh 完全采用 ANSI C 编写，具备极好的移植性，同时在内存占用上也非常小，如果不使用上述提到的 API 函数，整个 finsh 将不会动态申请内存。

finsh shell 线程：
每次的命令执行都是在 finsh shell 线程的上下文中完成的，finsh shell 线程在函数 finsh_system_init()中创建，它将一直等待 uart_sem 信号量的释放。

finsh 的输出：
finsh 的输出依赖于系统的输出，在 RT-Thread 中依赖的是 rt_kprintf 输出。

finsh 的输入：
finsh shell 线程在获得了 uart_sem 信号量后调用 rt_serial_getc()函数从串口中获得一个字符然后处理。所以 finsh 的移植需要 rt_serial_getc()函数的实现。而 uart_sem 信号量的释放通过调用 finsh_notify()函数以完成对 finsh shell 线程的输入通知。

通常的过程是，当串口接收中断发生时（即串口中有输入），接收中断服务例程调用 finsh_notify()函数通知 finsh shell 线程有输入；而后 finsh shell 线程获取串口输入最后做相应的命令处理。

12.6 选项

要开启 finsh 的支持，在 RT-Thread 的配置中必须定义 RT_USING_FINSH 宏。