

User  
manual

# SylixOS Shell 用户手册

## SylixOS 用户

UM0010010001

V1.00

Date: 2016/12/3

用户手册

类别	内容
关键词	SylixOS Shell
摘 要	SylixOS 的 Shell 命令的使用及配置方法

## 修订历史

版本	日期	原因
V1.00	2016/12/3	梁增辉 创建文档

## 目 录

1. 添加与配置.....	1
1.1. 介绍.....	1
1.2. 配置.....	1
1.3. 添加.....	2
2. 系统命令.....	3
2.1. 介绍.....	3
2.2. 命令使用.....	4
2.2.1. aborts - 显示当前操作系统异常处理统计信息.....	4
2.2.2. affinity - 显示或设置线程或进程调度的 CPU 集合 .....	5
2.2.3. buss - 显示系统中挂载的所有总线信息.....	7
2.2.4. clear - 清除当前屏幕 .....	7
2.2.5. color - 根据 LS_COLORS 初始化配色方案 .....	8
2.2.6. cpuus - 查看 cpu 利用率 .....	9
2.2.7. devs - 显示系统中挂载的所有设备.....	9
2.2.8. drvlics - 显示系统中所有安装的驱动程序表许可证信息.....	11
2.2.9. drvs - 显示设备驱动程序表信息 .....	11
2.2.10. env - 查看当前的环境变量表 .....	12
2.2.11. echo - 回显命令，此命令将回显用户输入的参数 .....	13
2.2.12. eventset - 展示事件集信息 .....	14
2.2.13. exec - 执行一个程序.....	14
2.2.14. exit - 退出当前 shell 终端.....	15
2.2.15. free - 显示系统当前的内存信息 .....	15
2.2.16. help - 显示 ttinyShell 所有内建命令列表，如果带有参数，则可显示以参 数为名的命令介绍.....	16
2.2.17. hostname - 显示或设置当前 SylixOS 镜像主机名 .....	17
2.2.18. ints - 查看系统中断向量表信息 .....	18
2.2.19. kill - 向指定的线程或进程发送信号，默认发送 SIGKILL 信号.....	19
2.2.20. login - 切换用户，重新登录.....	20
2.2.21. loglevel - 显示或设置当前内核日志（printk）打印等级 .....	21
2.2.22. mems - 查看操作系统内核内存堆与系统内存堆内存使用情况 .....	21
2.2.23. part - 显示指定的内存分区信息 .....	22
2.2.24. pcis - 显示 PCI 总线的信息 .....	23
2.2.25. ps - 查看系统所有进程的信息 .....	24
2.2.26. reboot - 重新启动计算机 .....	25
2.2.27. renice - 设置指定进程的优先级 .....	26
2.2.28. restart - 重新启动线程 .....	26
2.2.29. sem - 显示信号信息 .....	27
2.2.30. shell - 使用 ttydevice 作为标准文件创建一个 shell .....	28
2.2.31. shstack - 显示或者设置 shell 任务堆栈大小 .....	29
2.2.32. shutdown - 关闭或重启系统.....	29
2.2.33. sigqueue - 发送信号给线程或进程 .....	30

2.2.34.	sleep - 当前线程睡眠指定的时间 .....	31
2.2.35.	sprio - 设置指定线程的优先级 .....	31
2.2.36.	ss - 查看系统中所有线程与中断系统堆栈使用情况 .....	32
2.2.37.	top - 查看 cpu 的使用率 .....	33
2.2.38.	tp - 查看系统中被阻塞的线程信息 .....	34
2.2.39.	ts - 查看系统线程信息 .....	35
2.2.40.	tty - 显示当前 shell 终端对应的 tty 文件 .....	36
2.2.41.	vardel - 删除一个指定的系统环境变量 .....	36
2.2.42.	varload - 从指定参数的文件中提取装载环境变量表 .....	37
2.2.43.	vars - 显示当前的环境变量 .....	38
2.2.44.	varsave - 保存当前的操作系统环境变量表 .....	39
2.2.45.	virtuals - 显示 vmm 虚拟存储器信息 .....	40
2.2.46.	which - 检查参数指定的文件位置 .....	41
2.2.47.	who - 查看当前登录用户身份 .....	41
2.2.48.	zones - 查看操作系统物理页面分区使用情况 .....	42
3.	文件命令 .....	43
3.1.	介绍 .....	43
3.2.	命令使用 .....	44
3.2.1.	cat - 显示文件内容 .....	44
3.2.2.	cd - 切换当前目录 .....	44
3.2.3.	ch - 改变目录 .....	45
3.2.4.	chmod - 改变文件或目录的访问权限 .....	45
3.2.5.	close - 关闭一个文件 .....	46
3.2.6.	cmp - 比较一个文件 .....	46
3.2.7.	cp - 拷贝一个文件 .....	47
3.2.8.	df - 查看指定目录的文件系统信息 .....	48
3.2.9.	dosfslabel - 查看或设置文件系统卷标 .....	48
3.2.10.	dsize - 计算一个指定的目录包含的所有文件信息 .....	49
3.2.11.	fatugid - 设置 fat 文件系统用户与组 id .....	49
3.2.12.	fdentrys - 列出操作系统所有正在操作的文件信息 .....	50
3.2.13.	fdisk - 显示或制作一个磁盘分区表 .....	50
3.2.14.	files - 列出操作系统所有正在操作的文件信息 .....	51
3.2.15.	gzip - 压缩或解压缩一个文件 .....	52
3.2.16.	ll - 显示指定目录下的文件详细信息 .....	53
3.2.17.	ln - 创建符号链接文件 .....	53
3.2.18.	logfileadd - 向内核日志打印函数加入指定的内核文件描述符 .....	54
3.2.19.	logfileclear - 从内核日志打印文件中清除指定的内核文件描述符 .....	54
3.2.20.	logfiles - 显示内核日志打印文件列表 .....	55
3.2.21.	ls - 显示指定目录下的文件名，默认当前目录 .....	55
3.2.22.	mkdir - 创建目录 .....	56
3.2.23.	mkfifo - 创建一个命名管道，只能在根文件系统设备下创建 .....	56
3.2.24.	mkfs - 格式化指定的磁盘 .....	57
3.2.25.	mmaps - 显示系统 mmap 信息 .....	57
3.2.26.	mount - 挂载一个卷 .....	58

3.2.27.	msgq – 显示消息队列的信息 .....	59
3.2.28.	mv – 移到或重命名一个文件 .....	60
3.2.29.	open – 打开一个文件 .....	60
3.2.30.	pwd – 查看当前的工作目录 .....	61
3.2.31.	rm – 删除一个文件 .....	61
3.2.32.	rmdir – 删除一个文件夹 .....	62
3.2.33.	showmount – 查看系统中所有已经挂载的卷 .....	62
3.2.34.	shfile – 执行指定的 shell 脚本 .....	63
3.2.35.	sync – 将所有系统缓存的信息写入到物理设备 .....	64
3.2.36.	tmpname – 获得一个可以创建的临时文件名 .....	64
3.2.37.	touch – 创建一个普通文件 .....	64
3.2.38.	umount – 卸载一个卷 .....	65
3.2.39.	untar - 解包或解压缩一个 tar 或 tar.gz 文件包 .....	66
3.2.40.	vi - 启动 vi 编辑器 .....	66
3.2.41.	yaffscmd - 显示、设置和擦除一个块 .....	67
3.2.42.	zlib - 添加一个 .gz 的压缩文件 .....	68
4.	用户命令 .....	69
4.1.	介绍 .....	69
4.2.	命令使用 .....	69
4.2.1.	gadd – 增加一个新的用户组 .....	69
4.2.2.	gdel – 删除一个用户组 .....	70
4.2.3.	group – 显示用户组的信息 .....	70
4.2.4.	pmod – 修改用户密码 .....	71
4.2.5.	uadd – 添加用户 .....	71
4.2.6.	udel – 删除用户 .....	72
4.2.7.	umod – 设置用户的模式 .....	72
4.2.8.	user - 显示用户信息或为用户生成密码 .....	73
5.	网络 .....	74
5.1.	介绍 .....	74
5.2.	命令使用 .....	74
5.2.1.	aodvs – 显示 aodv 路由表 .....	74
5.2.2.	arp – 添加、删除或查看 ARP 表 .....	75
5.2.3.	ftpdpath – 查看或设置 ftp 服务器初始化路径 .....	76
5.2.4.	ftpd – 显示 ftp 服务器信息 .....	76
5.2.5.	hosttable - 查看、添加或删除主机地址映射 .....	77
5.2.6.	ifconfig - 显示或配置网络配置信息 .....	77
5.2.7.	ifdown – 禁用一个网络接口 .....	78
5.2.8.	ifrout – 设置默认路由接口 .....	79
5.2.9.	ifup – 启用一个网络接口 .....	79
5.2.10.	ipv6 – 设置或显示 ipv6 .....	80
5.2.11.	nat – 启动、关闭或设置 NAT 虚拟网络地址服务 .....	81
5.2.12.	natalias – 添加或删除 NAT 别名 .....	81
5.2.13.	natmap – 添加或删除 NAT 映射 .....	82
5.2.14.	nats – 查看当前 NAT 虚拟地址服务状态 .....	82

5.2.15.	nbname – 显示或设置本机的 NetBIOS 的名字 .....	83
5.2.16.	netstat – 查看网络状态.....	84
5.2.17.	npfattach - 在指定网络接口上使能网络数据包过滤器 .....	85
5.2.18.	npfdetach - 在指定网络接口上禁能网络数据包过滤器.....	86
5.2.19.	npfruleadd – 添加一条网络数据包过滤器规则 .....	86
5.2.20.	npfruledel – 删除一条网络数据包过滤器规则 .....	87
5.2.21.	npfs – 查看网络数据包过滤器状态 .....	87
5.2.22.	ping - Ping 命令.....	88
5.2.23.	ping6 – IPv6 Ping 命令 .....	89
5.2.24.	route – 添加、删除、修改或查看系统路由表 .....	89
5.2.25.	tftp - 使用 tftp 命令接收或者发送一个文件 .....	91
5.2.26.	tftpdpath - 查看或设置 tftp 服务器本地路径 .....	92
5.2.27.	vlan - 显示、设置和删除 net 接口的 VLAN ID.....	92
5.2.28.	vpnclose - 删除一个虚拟网络接口? .....	93
5.2.29.	vpnopen - 创建一个虚拟网络接口? .....	93
6.	时间.....	94
6.1.	介绍.....	94
6.2.	命令使用.....	94
6.2.1.	date – 显示或设置系统当前时间.....	94
6.2.2.	hwclock – 显示或同步操作系统与硬件 RTC 时钟.....	95
6.2.3.	times – 显示 utc 或 local 时间 .....	96
6.2.4.	tzsync – 与环境变量 TZ 的时区同步 .....	96
7.	动态装载.....	97
7.1.	介绍.....	97
7.2.	命令使用.....	97
7.2.1.	debug – 调试一个进程.....	97
7.2.2.	dlconfig - 配置动态链接器工作参数.....	98
7.2.3.	leakchk – 内存泄漏检查.....	99
7.2.4.	leakchkstart – 启动内存泄漏跟踪器 .....	99
7.2.5.	leakchkstop – 关闭内存泄漏跟踪器.....	100
7.2.6.	lsmod - 查看系统装载的所有内核模块信息.....	100
7.2.7.	modulegcov – 生成内核模块代码文件 (*.gcda) .....	101
7.2.8.	modulereg – 注册一个模块 .....	101
7.2.9.	modules - 查看系统装载的所有内核模块与进程动态链接库信息 .....	102
7.2.10.	modulestat - 查看一个内核模块或动态链接库文件信息.....	103
7.2.11.	moduleunreg – 卸载一个模块 .....	103
8.	其他.....	104
8.1.	介绍.....	104
8.2.	命令使用.....	104
8.2.1.	args – 显示输入的全部参数，并且以空格为分隔符 .....	104
8.2.2.	crypt – 对数据进行加密.....	105
8.2.3.	perfrefresh – 更新性能信息统计.....	105
8.2.4.	perfs – 显示性能统计的信息.....	106
8.2.5.	perfstart – 启动性能分析工具.....	106

8.2.6.	perfstop – 停止性能分析工具 .....	107
8.2.7.	xmodemr - 使用 xmodem 协议接收一个文件 .....	107
8.2.8.	xmodems - 使用 xmodem 协议发送一个文件 .....	108

# 1. 添加与配置

## 1.1. 介绍

本章节将介绍关于怎样去添加或配置 shell 命名。

## 1.2. 配置

### 1.2.1. 配置 shell

SylixOS 的 Shell 支持可裁剪。如果不需要 shell 功能，只需要将 `LW_CFG_SHELL_EN` 配置宏置 0 即可。

配置 `LW_CFG_SHELL_EN` 为 0，找到 Base 工程下的 `/libsylixos/SylixOS/config/shell/shell_cfg.h` 只需要将 `LW_CFG_SHELL_EN` 置为 0，重新编译 Base 和 Bsp 工程，然后重新启动虚拟机，即可。

### 1.2.2. 配置某个 shell 命令

SylixOS 的 Shell 命令支持可裁剪。如果需要裁剪某个命令，只需要将具体命令的配置宏置位 0，就可以裁剪该命令。

以 `uadd` 命令为例，查看 `uadd` 命令介绍：

`uadd` – 添加用户

格式：

```
uadd name password enable[0/1] uid gid comment homedir
```

说明：

该命令用来添加一个用户。

返回值：

执行成功返回 0，失败返回-1。

备注：

用户 id 不能重复，添加用户时必须将其添加到一个存在的用户组。

样例：

配置：

当 `LW_CFG_SHELL_EN>0` 时，会允许操作系统提供 `tshell` 命令。

当(`LW_CFG_SHELL_USER_EN > 0`)，使能了 shell 用户管理工具，该命令将会被包含。

函数接口：

`uadd` 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellUserCmdUadd (INT iArgC, PCHAR ppcArgV[]);
```

配置 `LW_CFG_SHELL_USER_EN` 为 0，裁剪 `uadd` 命令。找到 `shell_cfg.h` 文件，将 `LW_CFG_SHELL_USER_EN` 置 0，重新编译 base 和 bsp 文件，然后重新启动虚拟机。(shell\_cfg.h 配置文件的路径为 `base/libsylixos/SylixOS/config/shell/shell_cfg.h`)

使用 `help` 查看是否有 `uadd` 命令。

```
[root@sylixos:/root]# help uadd
[root@sylixos:/root]# help udel
[root@sylixos:/root]# uadd
sh: command not found.
[root@sylixos:/root]# udel
sh: command not found.
[root@sylixos:/root]#
```



查看可知 uadd 命令的帮助文件和该命令都被裁剪了。

注意：配置 shell 命令应配置最小项宏，减小配置的范围。上例中配置了 uadd 命令，但相关的命令，例 udel、umod、gadd、gdel 等相关的命令也被裁剪了。

### 1.3. 添加

SylixOS 系统 shell 命令支持用户添加自己的命令。添加方式为：

在 ttintShellSysCmd.c 中写对应的接口函数，然后在 \_\_tshellSysCmdInit 添加 shell 命令。这里将用到三个系统接口。分别是：

```
LW_API ULONG API_TShellKeywordAdd (
    CPCHAR pcKeyword, PCOMMAND_START_ROUTINE pfuncCommand);
LW_API ULONG API_TShellFormatAdd (CPCHAR pcKeyword, CPCHAR pcFormat);
LW_API ULONG API_TShellHelpAdd (CPCHAR pcKeyword, CPCHAR pcFormat);
```

其中 API\_TShellKeywordAdd 用来添加 shell 命令，并将该 shell 命令与相应的处理函数对应起来。

API\_TShellFormatAdd 函数用来添加相应 shell 命令的使用格式。

TShellHelpAdd 用来添加响应 shell 命令的说明信息，介绍该命令要完成的相应功能。

例：添加一个 hello 的 shell 命令。

1、在 ttinyShellSysCmd.c 文件中添加 \_\_tshellCostomCmdHello 函数

```
static INT __tshellCostomCmdHello(INT iArgC, PCHAR ppcArgU[])
{
    if(iArgC > 2)
    {
        fprintf(stderr, "argument error.\n");
        return (-ERROR_TSHELL_EPARAM);
    }
    if (iArgC == 1)
    {
        printf("hello SylixOS\n");
    }
    if (iArgC == 2)
    {
        printf("hello %s\n", ppcArgU[1]);
    }
    return 0;
}
```

2、在 \_\_tshellSysCmdInit 函数中添加代码

```
VOID __tshellSysCmdInit (VOID)
{
    API_TShellKeywordAdd("hello", __tshellCostomCmdHello);
    API_TShellFormatAdd ("hello", " [message]");
    API_TShellHelpAdd ("hello", "show hello sylixOS\n");
}
```

3、重新编译 base 工程和 bsp 工程。然后启动虚拟机测试是否添加成功

```
[root@sylixos:/root]# help hello
show hello sylixOS
hello [message]
[root@sylixos:/root]# hello
hello SylixOS
[root@sylixos:/root]# hello world
hello world
[root@sylixos:/root]# hello SylixOS world
argument error.
parameter(s) error.
[root@sylixos:/root]#
```

测试得知添加成功。

## 2. 系统命令

### 2.1. 介绍

和系统相关的命令有：

- aborts - 显示当前操作系统异常处理统计信息
- affinity - 显示或设置线程或进程调度的 CPU 集合
- buss - 显示系统中挂载的所有总线信息
- clear - 清除当前屏幕
- color - 根据 LS\_COLORS 初始化配色方案
- cpuus - 查看 cpu 利用率
- devs - 显示系统中挂载的所有设备
- drvlics - 显示系统中所有安装的驱动程序表许可证信息
- drvs - 显示设备驱动程序表信息
- echo - 回显命令，此命令将回显用户输入的参数
- env - 查看当前的环境变量表
- eventset - 展示事件集信息
- exec - 执行一个程序
- exit - 退出当前 shell 终端
- free - 显示系统当前的内存信息
- help - 显示 ttinyShell 所有内建命令列表，如果带有参数，则可显示以参数为名的命令介绍
- hostname - 显示或设置当前 SylixOS 镜像主机名
- ints - 查看系统中断向量表信息
- kill - 向指定的线程或进程发送信号，默认发送 SIGKILL 信号
- login - 切换用户，重新登录
- loglevel - 显示或设置当前内核日志（printk）打印等级
- mems - 查看操作系统内核内存堆与系统内存堆内存使用情况
- part - 显示指定的内存分区信息
- pcis - 显示 PCI 总线的信息
- ps - 查看系统所有进程的信息
- reboot - 重新启动计算机
- renice - 设置指定进程的优先级
- restart - 重新启动线程

- sem - 显示信号信息
- shell - 使用 `ttydevice` 作为标准文件创建一个 shell
- shstack - 显示或者设置 shell 任务堆栈大小
- shutdown - 关闭或重启系统
- sigqueue - 发送信号给线程或进程
- sleep - 当前线程睡眠指定的时间
- sprio - 设置指定线程的优先级
- ss - 查看系统中所有线程与中断系统堆栈使用情况
- top - 查看 cpu 的使用率
- tp - 查看系统中被阻塞的线程信息
- ts - 查看系统线程信息
- tty - 显示当前 shell 终端对应的 tty 文件
- vardel - 删除一个指定的系统环境变量
- varload - 从指定参数的文件中提取装载环境变量表
- vars - 显示当前的环境变量
- varsave - 保存当前的操作系统环境变量表
- virtuals - 显示 vmm 虚拟存储器信息
- which - 检查参数指定的文件位置
- who - 查看当前登录用户身份
- zones - 查看操作系统物理页面分区使用情况

## 2.2. 命令使用

和系统相关 shell 命令的详细介绍和使用方法，主要有使用格式，说明，备注，使用样例，可裁剪配置，以及响应的接口。

### 2.2.1. aborts - 显示当前操作系统异常处理统计信息

格式：

aborts

说明：

该命令会显示系统当前内存访问异常、内存分配错误、分配物理页错误和页面映射错误次数。

返回值：

返回值是 0。

备注：

无。

样例：

```
[root@sylixos_station:/root]# aborts
vmm abort statistics infomation show >>
vmm abort (memory access error) counter : 0
vmm page fail (alloc success) counter   : 0
vmm alloc physical page error counter   : 0
vmm page map error counter               : 0
[root@sylixos_station:/root]#
```

配置：

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_VMM\_EN>0 时，对虚拟内存支持，该命令将会被包含。

函数接口：

aborts 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdAborts (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.2. affinity - 显示或设置线程或进程调度的 CPU 集合

格式：

```
affinity
```

```
affinity pid | thread id cpu id
```

```
affinity pid | thread id 'clear'
```

说明：

该命令有三种用法，affinity 不跟参数时，显示线程和进程调度的 cpu 集合；当 affinity 参数是线程 id 或进程 id、cpu id 时，设置该进程或线程调度的 cpu 集合；当 affinity 参数是线程 id 或进程 id、clear 时，清除该进程或线程对 cpu 集合的调度。

返回值：

执行成功返回 0，失败返回-1。

备注：

线程或进程调度 cpu 的信息存放在文件/proc/kernel/affinity 中。

样例：

```
[root@sylixos_station:/root]# affinity
      NAME      TID    PID  CPU
-----
t_idle0        4010000    0    0
t_itimer        4010001    0    *
t_isrdefer      4010002    0    *
t_except        4010003    0    *
t_log           4010004    0    *
t_power         4010005    0    *
t_hotplug       4010006    0    *
t_reclaim       4010008    0    *
t_netjob        4010009    0    *
t_netproto      401000a    0    *
t_tftpd         401000b    0    *
t_ftpd          401000c    0    *
t_telnetd       401000d    0    *
t_tshell        401000f    0    *
[root@sylixos_station:/root]# affinity 4010009 0
affinity set thread 0x4010009 to cpu 0 ok.
[root@sylixos_station:/root]# affinity
      NAME      TID    PID  CPU
-----
t_idle0        4010000    0    0
t_itimer        4010001    0    *
t_isrdefer      4010002    0    *
t_except        4010003    0    *
t_log           4010004    0    *
t_power         4010005    0    *
t_hotplug       4010006    0    *
t_reclaim       4010008    0    *
t_netjob        4010009    0    0
t_netproto      401000a    0    *
t_tftpd         401000b    0    *
t_ftpd          401000c    0    *
t_telnetd       401000d    0    *
t_tshell        401000f    0    *
[root@sylixos_station:/root]# affinity 4010009 clear
affinity clear thread 0x4010009 ok.
[root@sylixos_station:/root]# affinity
      NAME      TID    PID  CPU
-----
t_idle0        4010000    0    0
t_itimer        4010001    0    *
t_isrdefer      4010002    0    *
t_except        4010003    0    *
t_log           4010004    0    *
t_power         4010005    0    *
t_hotplug       4010006    0    *
t_reclaim       4010008    0    *
t_netjob        4010009    0    *
t_netproto      401000a    0    *
t_tftpd         401000b    0    *
t_ftpd          401000c    0    *
t_telnetd       401000d    0    *
t_tshell        401000f    0    *
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 (LW\_CFG\_SMP\_EN > 0) 和

(LW\_CFG\_POSIX\_EN > 0)时，系统对多处理器支持并且使能 posix 兼容库后，该命令将会被包含。

函数接口：

affinity 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdAffinity (INT iArgC, PCHAR ppcArgV[]);
```

### 2.2.3. buss – 显示系统中挂载的所有总线信息

格式：

buss

说明：

该命令用来显示系统中挂载的所有总线信息。

返回值：

执行返回 0。

备注：

无。

样例：

```
[root@sylixos_station:/root]# buss

  BUS NAME    DEV NUM
-----
/bus/i2c/0      0
[root@sylixos_station:/root]#
```

配置：

该命令属于系统提供的 tshell 命令。当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。

函数接口：

buss 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdBuss (INT iArgC, PCHAR ppcArgV[]);
```

### 2.2.4. clear – 清除当前屏幕

格式：

clear

说明：

该命令用来清除当前屏幕显示的信息。

返回值：

执行返回 0。

备注：

无。

样例：

8

color 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdColor (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.6. cpuus - 查看 cpu 利用率

格式：

```
cpuus
```

```
cpuus -n times -t wait_seconds
```

说明：

该命令有 2 种用法，其中 -n 后面跟的参数是检测 cpu 利用率的次数，-t 指明每次检查用的时间。cpuus 默认检测 1 次，检测的时间是 1s，cpuus -n times -t wait\_seconds 可以指明检测 times 次，每次检测的时间是 wait\_seconds 秒。规定检测时间不得超过 10 秒。

返回值：

执行返回 0。

备注：

无。

样例：

```
[root@sylixos_station:/etc]# cpuus
CPU usage checking, please wait...
CPU usage show (measurement accuracy 1.0%) >>

      NAME      TID  PRI   CPU   KERN
-----
t_tshell      401000f 150   0.0%   0.0%
t_telnetd     401000d 160   0.0%   0.0%
t_ftpd        401000c 160   0.0%   0.0%
t_tftpd       401000b 160   0.0%   0.0%
t_netproto    401000a 110   0.0%   0.0%
t_netjob      4010009 110   0.0%   0.0%
t_reclaim     4010008 253   0.0%   0.0%
t_hotplug     4010006 250   0.0%   0.0%
t_power       4010005 254   0.0%   0.0%
t_log         4010004  60   0.0%   0.0%
t_except      4010003  0    0.0%   0.0%
t_isrdefer    4010002  0    0.0%   0.0%
t_itimer      4010001  20   0.0%   0.0%
t_idle0       4010000 255  99.0%   0.0%

[root@sylixos_station:/etc]#
```

配置：

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

函数接口：

cpuus 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdCpuus (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.7. devs – 显示系统中挂载的所有设备

格式：



devs

devs -a

#### 说明:

该命令用于显示系统中挂载设备的信息，其中 `devs` 不带参数会显示挂载设备的设备号、打开情况和设备名字，`devs` 带参数 `-a` 会显示挂载设备全部的信息，包括设备号、打开情况、设备名字以及设备类型。

#### 返回值:

执行返回 0。

#### 备注:

无。

#### 样例:

```
[root@sylixos_station:/root]# devs
device show (minor device) >>
drv open name
32    0 /dev/input/xmse
32    0 /dev/input/xkbd
31    0 /dev/socket
30    0 /dev/netevent
28    1 /dev/input/touch0
29    0 /dev/fb0
26    0 /dev/yaffs2
17    0 /dev/ttyS2
17    0 /dev/ttyS1
17    1 /dev/ttyS0
14    0 /dev/urandom
14    0 /dev/random
13    0 /dev/shm
12    0 /proc
```

```
[root@sylixos_station:/root]# devs -a
device show (minor device) >>
drv open name          type
32    0 /dev/input/xmse  character
32    0 /dev/input/xkbd  character
31    0 /dev/socket      socket
30    0 /dev/netevent    character
28    2 /dev/input/touch0 character
29    0 /dev/fb0         character
26    0 /dev/yaffs2      directory
17    0 /dev/ttyS2       character
17    0 /dev/ttyS1       character
17    1 /dev/ttyS0       character
14    0 /dev/urandom     character
14    0 /dev/random      character
13    0 /dev/shm         directory
12    0 /proc            directory
```

#### 配置:

该命令属于系统提供的 `tshell` 命令，当 `LW_CFG_SHELL_EN>0` 时，会允许操作系统提供 `tshell` 命令，则包含该命令。

#### 函数接口:

`devs` 命令是通过 `c` 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdDevs (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.8. drvlics - 显示系统中所有安装的驱动程序表许可证信息

格式:

```
drvlics
```

说明:

该命令用于显示系统中安装设备的设备号、设备的描述、作者和版本号。

返回值:

执行返回 0。

备注:

无。

样例:

```
[root@sylixos_station:/root]# drvlics
driver License show (major device) >>

DRV      DESCRIPTION      AUTHOR      LICENSE
-----
0 null device driver.      Han.hui      GPL->Ver 2.0
1 zero device driver.      Han.hui      GPL->Ver 2.0
2 rootfs driver.           Han.hui      GPL->Ver 2.0
3 eventfd driver.          Han.hui      GPL->Ver 2.0
4 timerfd driver.          Han.hui      GPL->Ver 2.0
5 hstimerfd driver.        Han.hui      GPL->Ver 2.0
6 signalfd driver.         Han.hui      GPL->Ver 2.0
7 gpiofd driver.           Han.hui      GPL->Ver 2.0
8 blk io driver.           Han.hui      GPL->Ver 2.0
9 epoll driver.            Han.hui      GPL->Ver 2.0
10 hotplug message driver.  Han.hui      GPL->Ver 2.0
11 hardware rtc.           Han.hui      GPL->Ver 2.0
12 procfs driver.          Han.hui      GPL->Ver 2.0
```

配置:

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

函数接口:

drvlics 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellSysCmdDrvlics (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.9. drvs - 显示设备驱动程序表信息

格式:

```
drvs
```

说明:

该命令会显示设备驱动的驱动号 (drv) 及创建 (create)、删除 (delete)、打开 (open)、关闭 (close)、读 (read)、写 (write)、控制 (ioctl)、读设备扩展 (readex)、写设备扩展 (writeex)、选择功能 (select)、读写指针移动 (lseek)、建立链接文件、读取链接文件、文件映射函数的地址。

返回值:

执行返回 0。

备注:

无。

样例:

```
[root@sylixos_station:/root]# drvs
driver show (major device) >>
```

drv	create	delete	open	close	read	write	ioctl
1	301b9be0	0	301b9be0	301b9bd0	301b9c00	0	301b9cb4
2	3008b5f4	3008b4c4	3008b5f4	3008b024	3008ab90	3008ab78	3008aba8
3	0	0	3018db88	3018db20	3018d990	3018d828	3018d5d8
4	0	0	301c0c74	301c0bf8	301c0b14	0	301c0934
5	0	0	30190080	30190018	30190260	0	3018fe38
6	0	0	301c4428	301c3ebc	301c41b4	0	301c3f0c
7	0	0	3018e028	3018df9c	3018df40	3018deec	3018e20c
8	0	0	30081b94	30081b4c	30081c94	30081c28	3008137c
9	0	0	301b424c	301b4194	0	0	301b4358
10	301b52ec	0	301b52ec	301b5268	301b5160	301b4e70	301b4e88
11	30190b50	0	30190b50	30190b28	0	0	30190a28
12	30084414	0	30084414	300843a4	3008428c	30083b40	30083c2c
13	301afd00	301af664	301afd00	301af50c	301aec8c	301aec5c	301aeca4
14	301a5420	0	301a5420	301a54cc	301a5550	301a56e8	301a5700
15	301a4c18	0	301a4c18	301a4c3c	301a4c78	301a4d6c	301a4dc0
16	301a5000	0	301a5000	301a5024	301a5084	301a50b8	301a50ec

  

drv	readex	writeex	select	lseek	symlink	readlnk	mmap
1	0	0	0	0	0	0	0
2	0	0	0	0	3008b30c	3008b2c8	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0
8	30081914	30081670	0	0	0	0	0
9	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0
12	3008419c	30083b28	0	0	0	30083b58	0
13	301aec74	301aec44	0	301aec38	301afcd8	301af304	301af8b4
14	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0

配置:

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

函数接口:

drv 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellSysCmdDrvs (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.10. env – 查看当前的环境变量表

格式:

```
env
```

说明:

该命令用来显示系统全局的环境变量表。

返回值:

执行返回 0。

备注:

无。

样例:

```
[root@sylixos_station:/root]# env
variable show >>

-----
VARIABLE      REF      VALUE
-----
TERMCAP       /etc/termcap
TERM          vt100
LUA_CPATH     ?.so;/usr/local/lib/lua/?.so;/usr/lib/lua/?.so;/lib/lua/?.so
LUA_PATH      ?.lua;/usr/local/lib/lua/?.lua;/usr/lib/lua/?.lua;/lib/lua/?.lua
DEBUG_CPU     -1
PATH_LOCALE   /usr/share/locale
LC_ALL
LANG          C
LD_LIBRARY_PATH /usr/lib:/lib:/usr/local/lib
PATH          /usr/bin:/bin:/usr/pkg/sbin:/sbin:/usr/local/bin
NFS_CLIENT_PROTO udp
NFS_CLIENT_AUTH AUTH_UNIX
SYSLOGD_HOST  0.0.0.0:514
FIO_FLOAT     1
SO_MEM_PAGES  8192
TSLIB_CALIBFILE /etc/pointercal
TSLIB_TSDEVICE /dev/input/touch0
MOUSE         /dev/input/mouse0:/dev/input/touch0
KEYBOARD      /dev/input/keyboard0
TZ            CST-8:00:00
TMPDIR        /tmp/
LICENSE       SylixOS license: Commercial & GPL.
VERSION       1.3.5 (5)
SYSTEM        SylixOS kernel version: 1.3.5 (5) LongYuan(b)
```

#### 配置:

该命令属于系统提供的 tshell 命令。当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。

#### 函数接口:

env 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellSysCmdVars (INT iArgC, PCHAR ppcArgV[]);
```

### 2.2.11. echo - 回显命令，此命令将回显用户输入的参数

#### 格式:

```
echo message
```

#### 说明:

该命令 echo 后的参数回显。

#### 返回值:

执行返回 0。

#### 备注:

无。

#### 样例:

```
[root@sylixos_station:/root]# echo hello word
hello word
[root@sylixos_station:/root]#
```

#### 配置:

该命令属于系统提供的 tshell 命令。当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。

#### 函数接口:

echo 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellSysCmdEcho (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.12. eventset – 展示事件集信息

格式:

```
eventset eventset handle
```

说明:

该命令用于显示事件集的相关信息。

返回值:

执行成功返回 0，失败返回-1。

备注:

无。

样例:

使用 SylixOS 事件集操作函数编写程序，打印事件集 id 号。

```
[root@sylixos:/apps/mq]# eventset 20010009
event set show >>

event set name : event_set
event set event : 0x00000000

[root@sylixos:/apps/mq]#
```

配置:

该命令属于系统提供的 tshell 命令。当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。

函数接口:

eventset 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellSysCmdEventSet (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.13. exec – 执行一个程序

格式:

```
exec program file arguments
```

说明:

该命令用于执行一个程序，exec 后跟文件名，参数列表。

返回值:

执行成功返回 0，失败返回非 0 值。

备注:

无。

样例:

```
[root@sylixos_station:/root]# exec /apps/helloWord/helloWord
Hello SylixOS!
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_MODULELOADER\_EN>0，需要提供模块装载服务，该命令将会被包含。

函数接口:

exec 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellExec (INT iArgC, PCHAR *ppcArgV)
```

## 2.2.14. exit – 退出当前 shell 终端

格式:

exit

说明:

该命令用于退出当前 shell 终端。

返回值:

执行返回 0。

备注:

无。

样例:

```
[root@sylixos_station:/root]# exit
```

配置:

该命令属于系统提供的 tshell 命令。当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。

函数接口:

exit 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellSysCmdExit (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.15. free – 显示系统当前的内存信息

格式:

free

说明:

该命令用于显示当前内存的信息，如果对虚拟内存支持，会显示 vmm 物理存储器的信息。

返回值:

执行返回 0。

备注:

无。

样例:

```
[root@sylixos_station:/root]# free
heap show >>

      HEAP      TOTAL      USED      MAX USED  SEGMENT USED
-----
kersys      13279KB      3359KB      3376KB      1083  25%

vmm physical zone show >>

ZONE PHYSICAL  SIZE  PAGESIZE  PGD  FREEPAGE  DMA  USED
-----
  0 31800000   600000    1000 30bc8000    1348 true  12%
  1 31e00000  2200000    1000 30bc8000    8695 false  0%

ALL-Physical memory size: 64 MBytes (67108864 Bytes)
VMM-Physical memory size: 40 MBytes (41943040 Bytes)
VMM-Physical memory free: 39 MBytes (41136128 Bytes)
[root@sylixos_station:/root]#
```

#### 配置:

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

#### 函数接口:

free 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellSysCmdFree (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.16. help - 显示 ttinyShell 所有内建命令列表，如果带有参数，则可显示以参数为名的命令介绍

#### 格式:

help

help keyword

help -s keyword

#### 说明:

该命令只有 3 种用法，当 help 后面不跟参数时，显示 ttinyShell 所有内建命令列表，help keyword 与 help -s keyword 的功能相同，都是显示以 keyword 为名的命令介绍

#### 返回值:

执行成功返回 1，失败返回非 0 值。

#### 备注:

help 参数 keyword 支持所用通配符 (\*、?)，当使用通配符时，会列出匹配的内建命令列表

#### 样例:

```
[root@sylixos_station:/root]# help
ps
lsmod
modulegcov          [kernel module handle]
modules             [module name]
modulestat          [program file]
moduleunreg         [kernel module handle]
modulereg           [kernel module file *.ko]
dlconfig            {[share {en | dis}} | [refresh [*]]}
exec                [program file] [arguments...]
which               [program file]
mmaps
vpnclose            [netifname]
vpnopen             [configuration file]
npfdetach           [netifname]
npfattach           [netifname]
npfruledel          [netifname] [rule sequence num]
npfruleadd          [netifname] [rule] [args...]
npfs
nats
natmap              {[add] | [del]} [WAN port] [LAN port] [LAN IP] [protocol]
natalias            {[add] [alias] [LAN start] [LAN end]} | {[del] [alias]}
nat                 [stop] | {[LAN netif] [WAN netif]}
ftpdpath            [new path]

press ENTER to continue, 'Q' to quit.
```

```
[root@sylixos_station:/root]# help help
display help information.
help [-s keyword]
help [-s keyword]
[root@sylixos_station:/root]#
```

```
[root@sylixos_station:/root]# help l?
ln          [-s | -f] [actualpath] [sympath]
ll          [path name]
ls          [path name]
[root@sylixos_station:/root]#
```

### 配置：

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

### 函数接口：

help 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdHelp (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.17. hostname - 显示或设置当前 SylixOS 镜像主机名

### 格式：

```
hostname
hostname name
```

### 说明：

该命令有 2 种用法，当 hostname 后面没跟参数时，显示当前的 SylixOS 镜像主机名，如果跟了参数，则将 Sylix 镜像名改为该参数。

### 返回值：

执行成功返回 0，失败返回-1。



备注:

无。

样例:

```
[root@sylixos_station:/root]# hostname
hostname is sylixos_station
[root@sylixos_station:/root]# hostname shell
[root@shell:/root]# hostname
hostname is shell
[root@shell:/root]#
```

配置:

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

函数接口:

hostname 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellSysCmdHostname (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.18. ints – 查看系统中断向量表信息

格式:

ints

ints cupidstart

ints cupidstart cupidend

说明:

该命令用于显示所有 cpu 的中断向量表信息。ints cupidstart 显示 cpu id 大于等于 cupidstart 上的中断向量表信息。ints cupidstart cupidend 则显示 cpu id 大于等于 cupidstart 且小于等于 cupidend 上的中断向量表信息。

返回值:

执行返回 0。

备注:

本虚拟机只有 1 个 cpu。

样例:

```
[root@sylixos_station:/root]# ints
interrupt vector show >>

  IRQ      NAME      ENTRY      CLEAR      PARAM      ENABLE  RND  PREEMPT      CPU 0
-----
   4 dm9000_isr    30002958        0 3062bb90   true                22
  14 tick_timer    30000a0c        0      0   true            48655
  15 uart2_isr    30006410        0 3062deb4  false                0
  17 dma0_isr    300012f8        0      0   true                0
  18 dma1_isr    300012f8        0      1   true                0
  19 dma2_isr    300012f8        0      2   true                0
  20 dma3_isr    300012f8        0      3   true                0
  23 uart1_isr    30006410        0 3062de30  false                0
  27 i2c_isr    3000167c        0 3062bcbc  false                0
  28 uart0_isr    30006410        0 3062ddac   true            565
  31 touchscr    30005fb8        0      0   true                0

interrupt nesting show >>

  CPU  MAX NESTING      IPI
-----
   0          1          0
```

配置:

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

函数接口:

ints 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellSysCmdInts (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.19. kill - 向指定的线程或进程发送信号，默认发送 SIGKILL 信号

格式:

```
kill tid/pid
```

```
kill -n signum tid/pid
```

说明:

该命令有 2 种用法，kill tid/pid，向进程或线程发送 SIGKILL(9)信号，kill -n signum tid/pid 向进程或线程发送数字 signum 所代表的信号。

返回值:

执行成功返回 0，失败返回 -1。

备注:

进程号是十进制，线程号是十六进制。

样例:

```
[root@sylixos_station:/root]# ts
thread show >>

  NAME          TID      PID  PRI  STAT  ERRNO      DELAY  PAGEFAILS  FPU  CPU
-----
t_idle0         4010000    0 255  RDY      0          0          0      0
t_itimer        4010001    0  20  SLP      0          1          0      0
t_isrdefer      4010002    0   0  SEM      0          0          0      0
t_except        4010003    0   0  SEM      0          0          0      0
t_log           4010004    0  60  MSGQ     0          0          0      0
t_power         4010005    0 254  SLP      0         86          0      0
t_hotplug       4010006    0 250  SEM     506         86          0      0
t_reclaim       4010008    0 253  MSGQ     0          0          0      0
t_netjob        4010009    0 110  SEM      0          0          0      0
t_netproto      401000a    0 110  MSGQ     506          8          0      0
t_telnetd       401000d    0 160  MSGQ      2          0          0      0
t_xinput        401000e    0 199  SEM     506         79          0      0
t_touch         4010010    0 160  SEM      0          0          0      0
t_tshell        4010011    0 150  RDY    1503          0          0      0

thread: 14
[root@sylixos_station:/root]# kill 0x4010010
[root@sylixos_station:/root]# ts
thread show >>

  NAME          TID      PID  PRI  STAT  ERRNO      DELAY  PAGEFAILS  FPU  CPU
-----
t_idle0         4010000    0 255  RDY      0          0          0      0
t_itimer        4010001    0  20  SLP      0          1          0      0
t_isrdefer      4010002    0   0  SEM      0          0          0      0
t_except        4010003    0   0  SEM      0          0          0      0
t_log           4010004    0  60  MSGQ     0          0          0      0
t_power         4010005    0 254  SLP      0         61          0      0
t_hotplug       4010006    0 250  SEM     506         61          0      0
t_reclaim       4010008    0 253  MSGQ     0          0          0      0
t_netjob        4010009    0 110  SEM      0          0          0      0
t_netproto      401000a    0 110  MSGQ     506          3          0      0
t_telnetd       401000d    0 160  MSGQ      2          0          0      0
t_xinput        401000e    0 199  SEM     506         54          0      0
t_touch         4010010    0 160  SEM      0          0          0      0
t_tshell        4010011    0 150  RDY    1503          0          0      0

thread: 13
[root@sylixos_station:/root]#
```

#### 配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_SIGNAL\_EN > 0 时, 允许系统使用信号, 该命令将会被包含。

#### 函数接口:

kill 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellSysCmdKill (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.20. login - 切换用户, 重新登录

#### 格式:

```
login
```

#### 说明:

该命令用于切换用户, 输入 login 回车会提示你输入你登录的用户名、密码, 验证通过则切换用户, 否则还是以原先的用户身份登录。

#### 返回值:

执行返回 0。

#### 备注:

要登录的用户必须使能，否则登录失败。

#### 样例:

```
[root@sylixos_station:/root]# login
login: liang
password:
[liang@sylixos_station:/root]$
```

#### 配置:

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

#### 函数接口:

login 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellSysCmdLogin (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.21. loglevel - 显示或设置当前内核日志（printk）打印等级

#### 格式:

loglevel

loglevel level

#### 说明:

该命令有 2 种用法，显示当前内核日志 printk 接口的打印等级。若后面带参数，则将 printk 的打印等级设置成相应的等级。

#### 返回值:

执行返回 0。

#### 备注:

无。

#### 样例:

```
[root@sylixos_station:/root]# loglevel
printk() log message current level is: 4
[root@sylixos_station:/root]# loglevel 5
[root@sylixos_station:/root]# loglevel
printk() log message current level is: 5
[root@sylixos_station:/root]#
```

#### 配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_LOG\_LIB\_EN > 0 时，允许系统提供日志管理库，该命令将会被包含。

#### 函数接口:

loglevel 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellSysCmdLogLevel (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.22. mems - 查看操作系统内核内存堆与系统内存堆内存使用情况

#### 格式:

mems  
mems rid

#### 说明:

该命令用来查看系统内核内存堆与系统内存堆使用情况。

#### 返回值:

执行返回 0。

#### 备注:

无。

#### 样例:

```
[root@sylixos_station:/root]# mems
heap show >>

      HEAP          TOTAL      USED      MAX USED  SEGMENT  USED
-----
kersys          13279KB      3359KB      3376KB    1085    25%
```

#### 配置:

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

#### 函数接口:

mems 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdMems (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.23. part - 显示指定的内存分区信息

#### 格式:

part partition handle

#### 说明:

该命令用来显示指定的内存分区信息

#### 返回值:

执行成功返回 0，失败返回-1。

#### 备注:

无

#### 样例:

运行以下程序

```
#include <SylixOS.h>

typedef struct my_element {
    INT    iValue;
} MY_ELEMENT;

#define ELEMENT_MAX    (8)
UINT8      _G_pucMyElementPool[sizeof(MY_ELEMENT) * ELEMENT_MAX];
LW_HANDLE   _G_hMyPartition;

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

```

_G_hMyPartition = Lw_Partition_Create("my_partition",
                                     _G_pucMyElementPool,
                                     ELEMENT_MAX,
                                     sizeof(MY_ELEMENT),
                                     LW_OPTION_DEFAULT,
                                     LW_NULL);

if (_G_hMyPartition == LW_HANDLE_INVALID) {
    fprintf(stderr, "create partition failed.\n");
    return (-1);
}

printf("%x\n", _G_hMyPartition);
while(1);
return (0);
}

```

```

[root@sylixos:/apps/partition]# ./partition
4001000b

```

得到内存分区句柄。

```

[root@sylixos:/apps/partition]# part 4001000b
partition show >>

partition name      : my_partition
partition block number :      8
partition free block  :      8
partition per block size :      4
[root@sylixos:/apps/partition]#

```

**配置：**

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

**函数接口：**

part 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdPart (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.24. pcis – 显示 PCI 总线的信息

**格式：**

pcis

**说明：**

该命令显示 PCI 总线的信息。

**返回值：**

执行返回 0。

**备注：**

PCI 总线的信息存放在 /proc/pci 文件中。

x86 环境测试。

**样例：**

```
[root@sylixos:/]# pcis
PCI info:
Bus 0 Slot 0 Function 0 VendorID 8086 DeviceID 1237:
  Class 6 [Bridge] Sub 0 [Host bridge] Prog-if 0 []
  IRQ Line-0 Pin-0
  Latency=0 Min Gnt=0 Max lat=0
  Base0 00000000 Base1 00000000 Base2 00000000
  Base3 00000000 Base4 00000000 Base5 00000000
  Rom 0 SubVendorID 1af4 SubSystemID 1100
Bus 0 Slot 1 Function 0 VendorID 8086 DeviceID 7000:
  Class 6 [Bridge] Sub 1 [ISA bridge] Prog-if 0 []
  IRQ Line-0 Pin-0
  Latency=0 Min Gnt=0 Max lat=0
  Base0 00000000 Base1 00000000 Base2 00000000
  Base3 00000000 Base4 00000000 Base5 00000000
  Rom 0 SubVendorID 1af4 SubSystemID 1100
Bus 0 Slot 1 Function 1 VendorID 8086 DeviceID 7010:
  Class 1 [Mass storage controller] Sub 1 [IDE interface] Prog-if 128 []
  IRQ Line-0 Pin-0
  Latency=0 Min Gnt=0 Max lat=0
  Base0 00000000 Base1 00000000 Base2 00000000
  Base3 00000000 Base4 0000c101 Base5 00000000
  Rom 0 SubVendorID 1af4 SubSystemID 1100
  Region 4: I/O ports at c100 [Size 16] [flags 00040101]
Bus 0 Slot 1 Function 3 VendorID 8086 DeviceID 7113:
  Class 6 [Bridge] Sub 128 [Bridge] Prog-if 0 []
  IRQ Line-9 Pin-1
  Latency=0 Min Gnt=0 Max lat=0
  Base0 00000000 Base1 00000000 Base2 00000000
  Base3 00000000 Base4 00000000 Base5 00000000
  Rom 0 SubVendorID 1af4 SubSystemID 1100
  Irq start 9 end 9 [flags 00000400]
Bus 0 Slot 2 Function 0 VendorID 1013 DeviceID b8:
  Class 3 [Display controller] Sub 0 [VGA compatible controller] Prog-if 0 [VGA controller]
  IRQ Line-0 Pin-0
  Latency=0 Min Gnt=0 Max lat=0
  Base0 fc000008 Base1 febd0000 Base2 00000000
  Base3 00000000 Base4 00000000 Base5 00000000
  Rom febc0000 SubVendorID 1af4 SubSystemID 1100
  Region 0: Memory at fc000000 [Size 32M] [flags 00042208] (32-bit pre)
  Region 1: Memory at febd0000 [Size 4K] [flags 00040200] (32-bit non-pre)
  Region 6: Expansion ROM at febc0000 [Size 64K] [flags 00046200] (32-bit pre)
Bus 0 Slot 3 Function 0 VendorID 10ec DeviceID 8029:
  Class 2 [Network controller] Sub 0 [Ethernet controller] Prog-if 0 []
  IRQ Line-11 Pin-1
  Latency=0 Min Gnt=0 Max lat=0
  Base0 0000c001 Base1 00000000 Base2 00000000
  Base3 00000000 Base4 00000000 Base5 00000000
  Rom feb80000 SubVendorID 1af4 SubSystemID 1100
  Region 0: I/O ports at c000 [Size 256] [flags 00040101]
  Region 6: Expansion ROM at feb80000 [Size 256K] [flags 00046200] (32-bit pre)
  Irq start b end b [flags 00000400]
[root@sylixos:/]#
```

#### 配置:

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

#### 函数接口:

pcis 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellSysCmdPcis (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.25. ps – 查看系统所有进程的信息

#### 格式:

```
ps
```

#### 说明:

该命令用来显示系统中所有进程的信息。

返回值:

执行返回 0。

备注:

无。

样例:

```
[root@sylixos_station:/root]# ps
```

NAME	FATHER	PID	GRP	MEMORY	UID	GID	USER
kernel	<orphan>	0	0	0KB	0	0	root
test1	<orphan>	1	1	232KB	0	0	root

```
total vprocess: 2
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_MODULELOADER\_EN>0，需要提供模块装载服务，该命令将会被包含。

函数接口:

ps 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellVProcShow (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.26. reboot – 重新启动计算机

格式:

reboot

reboot -n/-f

说明:

该命令有 2 种用法，reboot 后面不跟参数时以冷启动的方式重新启动计算机。当 reboot 后面跟的参数是-n/-f 时，立即重新启动计算机；如果跟的是其他参数，则提示参数错误。

返回值:

执行成功返回 0，错误返回-1。

备注:

无。

样例:

```
[root@sylixos_station:/root]# reboot
[reboot]Reboot...
kernel rebooting...
kernel rebooting down.
see you

[root@sylixos_station:/root]# reboot -r
argument error.
[root@sylixos_station:/root]# reboot -n
[reboot]Force reboot...
```

配置:

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供



tshell 命令，则包含该命令。

函数接口：

reboot 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdReboot (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.27. renice – 设置指定进程的优先级

格式：

```
renice priority pid
renice priority -p pid
renice priority -g pgrp
renice priority -u user
```

说明：

该命令有 4 种用法，

```
renice priority pid      调整 pid 进程的优先级；
renice priority -p pid   调整 pid 进程的优先级；
renice priority -g pgrp  调整进程组的优先级；
renice priority -u user  调整进程使用者的优先级。
```

返回值：

执行成功返回 0，失败返回-1。

备注：

无。

样例：

使用 ts 查看线程的优先级

```
test1      4010019      1 247 JOIN      0      0      17      0
pthread    401001a      1 247 RDY      0      0      0      0
pthread    401001b      1 247 RDY      0      0      0      0
```

修改进程的优先级

```
[root@sylixos_station:/root]# renice 2 1
[root@sylixos_station:/root]#
```

再次使用 ts 查看现场的优先级

```
test1      4010019      1 249 JOIN      0      0      17      0
pthread    401001a      1 249 RDY      0      0      0      0
pthread    401001b      1 249 RDY      0      0      0      0
```

配置：

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

函数接口：

renice 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdRenice (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.28. restart – 重新启动线程

格式：

```
restart tid argument
```

**说明:**

该命令用来重新启动线程。

**返回值:**

执行成功返回 0，失败返回-1。

**备注:**

线程 id 是十六进制的。

只能重启内核线程。

**样例:**

在 bsp 的 main 函数中创建线程

```
static void * task0(void *arg)
{
    printf("hello task\n");
    while(1);
    return NULL;
}
```

重新启动虚拟机，线程给线程id，重新启动线程

```
t_tshell      4010012      0 150 SEM      71      0      0      0
t_main        4010013      0 200 JOIN      71      0      0      0
t_tshell      4010014      0 150 RDY      1503    0      0      0
pthread       4010015      0 250 RDY      1503    0      0      0

thread: 20
[unknown@sylixos:/]# restart 4010015 0
[unknown@sylixos:/]# hello task
```

**配置:**

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_THREAD\_RESTART\_EN>0 时，允许任务重新启动，该命令将会被包含。

**函数接口:**

restart 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdRestart (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.29. sem – 显示信号信息

**格式:**

sem semaphore handle

**说明:**

该命令显示指定信号的信息。

**返回值:**

执行成功返回 0，失败返回-1。

**备注:**

无。

**样例:**

tp 查看信号句柄

```
[root@sylixos_station:/root]# tp
thread pending show >>

  NAME          TID      PID  STAT    DELAY      PEND EVENT      OWNER
-----
t_itimer        4010001    0  SLP          3          0:
t_isrdefer      4010002    0  SEM          0 10010002:job_sync
t_except        4010003    0  SEM          0 10010003:job_sync
t_log           4010004    0  MSGQ         0 1c010004:log_msg:R
t_power         4010005    0  SLP          97          0:
t_hotplug       4010006    0  SEM          97 10010011:job_sync
t_reclaim       4010008    0  MSGQ         0 1c01001b:res_reclaim:R
t_netjob        4010009    0  SEM          0 1001005a:job_sync
t_netproto      401000a    0  MSGQ         2 1c010060:lwip_msg:R
t_tftpd         401000b    0  MSGQ         0 1c01007e:lwip_msg:R
t_ftpd          401000c    0  MSGQ         0 1c010082:lwip_msg:R
t_telnetd       401000d    0  MSGQ         0 1c010085:lwip_msg:R

pending thread: 12
[root@sylixos_station:/root]#

[root@sylixos_station:/root]# sem 1001005a
Semaphore show >>

Semaphore Name      : job_sync
Semaphore Id        : 0x1001005a
Semaphore Type      : BINARY
Thread Queuing      : FIFO
Pended Threads      : 1
Semaphore Value      : EMPTY

[root@sylixos_station:/root]#
```

#### 配置:

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

#### 函数接口:

sem 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellSysCmdSem (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.30. shell - 使用 ttydevice 作为标准文件创建一个 shell

#### 格式:

```
shell tty device nologin
```

#### 说明:

用 tty 设备作为标准文件创建 shell。

#### 返回值:

执行返回 0。

#### 备注:

无。

#### 样例:

```
[root@sylixos:/root]# shell /dev/ttyS0
[root@sylixos:/root]# login:
```

#### 配置:

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

#### 函数接口：

shell 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdShell (INT iArgC, PCHAR ppcArgV[]);
```

### 2.2.31. shstack -显示或者设置 shell 任务堆栈大小

#### 格式：

```
shstack
```

```
shstack new stack size
```

#### 说明：

该命令有 2 种用法，无参数时显示默认的 shell 任务堆栈大小。当后面跟参数时设置之后启动的 shell 任务的堆栈大小。

#### 返回值：

执行返回 0。

#### 备注：

设置仅对之后启动的 shell 有效。

#### 样例：

```
[root@sylixos_station:/root]# shstack
default shell stack: 32768
[root@sylixos_station:/root]# shstack 40000
default shell stack: 40000
[root@sylixos_station:/root]# shstack
default shell stack: 40000
[root@sylixos_station:/root]#
```

#### 配置：

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

#### 函数接口：

shstack 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdShStack (INT iArgC, PCHAR ppcArgV[]);
```

### 2.2.32. shutdown – 关闭或重启系统

#### 格式：

```
shutdown
```

```
shutdown -r
```

```
shutdown -h
```

```
shutdown -f
```

#### 说明：

该命令有 4 种用法，其中 shutdown 与 shutdown -h 的功能是一样，都是关机重启系统；shutdown -r 是关机，冷启动操作系统；shutdown -r 是立即重新启动系统。

#### 返回值：

执行成功返回 0，识别返回-1。

备注：

无。

样例：

```
[root@sylixos_station:/root]# shutdown
[shutdown]Shutdown...
kernel rebooting...
kernel rebooting down.
```

```
[root@sylixos_station:/root]# shutdown -r
[shutdown]Shutdown & reboot...
kernel rebooting...
kernel rebooting down.
```

配置：

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

函数接口：

shutdown 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdShutdown (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.33. sigqueue – 发送信号给线程或进程

格式：

```
sigqueue -n signum tid/pid
```

```
sigqueue tid/pid
```

说明：

该命令有 2 种用法，sigqueue tid/pid，向进程或线程发送 SIGKILL (9) 信号；sigqueue -n signum tid/pid 向进程或线程发送数字 signum 所代表的信号。

返回值：

执行成功返回 0，失败返回-1。

备注：

进程号是十进制，线程号是十六进制。

样例：

查看指定进程中的线程信息

```
[root@sylixos_station:/apps/test1]# ts 3
thread show >>

  NAME      TID      PID  PRI  STAT  ERRNO      DELAY      PAGEFAILS  FPU  CPU
-----
test1      401001a    3  200  JOIN      0          0          17         0
pthread    401001b    3  245  RDY       0          0          0         0
pthread    401001c    3  243  RDY       0          0          0         0

thread: 3
[root@sylixos_station:/apps/test1]#
```

发送消息，并再次查看线程信息

```
[root@sylixos_station:/apps/test1]# sigqueue 3
[root@sylixos_station:/apps/test1]# ts 3
thread show >>

      NAME      TID    PID  PRI  STAT  ERRNO      DELAY  PAGEFAILS  FPU  CPU
-----
thread: 0
[root@sylixos_station:/apps/test1]#
```

#### 配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_SIGNAL\_EN > 0 时，允许系统使用信号，该命令将会被包含。

#### 函数接口:

sigqueue 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellSysCmdSigqueue (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.34. sleep - 当前线程睡眠指定的时间

#### 格式:

```
sleep n
sleep ns
sleep nm
sleep nh
sleep nd
```

#### 说明:

该命令有 5 种用法，其中 sleep n 和 sleep ns 都是睡眠 n 秒；sleep nm，睡眠 n 分钟；sleep nh，睡眠 n 小时，sleep nd 睡眠 n 天。

#### 返回值:

执行失败返回-1，执行成功且睡眠到了所指定的时间返回 0，否则返回剩余的秒数。

#### 备注:

无。

#### 样例:

```
[root@sylixos_station:/root]# sleep 2s
[root@sylixos_station:/root]#
```

#### 配置:

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

#### 函数接口:

sleep 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellSysCmdSleep (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.35. sprio – 设置指定线程的优先级

#### 格式:

```
sprio priority thread_id
```

#### 说明:

该命令用来设置指定线程的优先级。

**返回值：**

执行成功返回 0，识别返回-1。

**备注：**

thread\_id 是 16 进程。

**样例：**

使用 ts 命令查看现有进程的优先级

```
test1      401001a    2 200 JOIN    0      0      17      0
pthread    401001b    2 245 RDY     0      0      0      0
pthread    401001c    2 243 RDY     0      0      0      0
```

使用 sprio 改变线程的优先级

```
[root@sylixos_station:/root]# sprio 240 4010015
[root@sylixos_station:/root]#
```

再次使用 ts 命令查看是否改变了相应进程的优先级

```
test1      4010013    1 200 JOIN    0      0      17      0
pthread    4010014    1 245 RDY     0      0      0      0
pthread    4010015    1 240 RDY     0      0      0      0
```

**配置：**

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

**函数接口：**

sprio 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdSprio (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.36. ss - 查看系统中所有线程与中断系统堆栈使用情况

**格式：**

ss

**说明：**

该命令用来查看线程和中断堆栈的使用情况。

**返回值：**

执行返回 0。

**备注：**

无。

**样例：**

```
[root@sylixos_station:/root]# ss
thread stack usage show >>

      NAME      TID    PRI  STK  USE   STK  FREE  USED
-----
t_idle0        4010000 255      356   3740   8%
t_itimer        4010001 20      804   3292  19%
t_isrdefer      4010002 0       432   3664  10%
t_except        4010003 0       496   3600  12%
t_log           4010004 60      520   3576  12%
t_power         4010005 254     352   3744   8%
t_hotplug       4010006 250     432   7760   5%
t_reclaim       4010008 253     384  16000   2%
t_netjob        4010009 110     416   3680  10%
t_netproto      401000a 110    1040   3056  25%
t_tftpd         401000b 160    3136   5056  38%
t_ftpd          401000c 160    2080  10208  16%
t_telnetd       401000d 160    2072   4072  33%
t_tshell        401000f 150    4096  28672  12%

interrupt stack usage show >>

CPU  STK  USE   STK  FREE  USED
---
0      388   3708   9%
[root@sylixos_station:/root]#
```

配置:

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

函数接口:

cpuus 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellSysCmdSs (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.37. top – 查看 cpu 的使用率

格式:

```
top
top -n times -t wait_seconds
```

说明:

该命令有 2 种用法，其中 -n 后面跟的参数是检测 cpu 利用率的次数，-t 指明每次检查用的时间。cpuus 默认检测 1 次，检测的时间是 1s，top -n times -t wait\_seconds 可以指明检测 times 次，每次检测的时间是 wait\_seconds 秒。规定检测时间不得超过 10 秒。

返回值:

执行返回 0。

备注

无。

样例:



```
[root@sylixos_station:/root]# top
CPU usage checking, please wait...
CPU usage show (measurement accuracy 1.0%) >>

      NAME      TID    PRI    CPU    KERN
-----
t_tshell        401000f  150    1.0%    0.0%
t_telnetd       401000d  160    0.0%    0.0%
t_ftpd          401000c  160    0.0%    0.0%
t_tftpd         401000b  160    0.0%    0.0%
t_netproto      401000a  110    0.0%    0.0%
t_netjob        4010009  110    0.0%    0.0%
t_reclaim       4010008  253    0.0%    0.0%
t_hotplug       4010006  250    0.0%    0.0%
t_power         4010005  254    0.0%    0.0%
t_log           4010004   60    0.0%    0.0%
t_except        4010003   0    0.0%    0.0%
t_isrdefer      4010002   0    0.0%    0.0%
t_itimer        4010001  20    0.0%    0.0%
t_idle0         4010000  255   98.0%    0.0%

[root@sylixos_station:/root]#
```

配置:

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

函数接口:

top 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellSysCmdCpuus (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.38. tp – 查看系统中被阻塞的线程信息

格式:

tp

tp pid

说明:

该命令有 2 种用法，tp 不跟参数，会显示所有进程中被阻塞的线程信息；带参数会显示相应进程下被阻塞的线程信息。

返回值:

执行成功返回 0，失败返回 -1。

备注:

无。

样例:

```
[root@sylixos_station:/root]# tp 1
thread pending show >>

      NAME      TID    PID  STAT    DELAY    PEND EVENT    OWNER
-----
test1         4010013     1  JOIN         0  4010014:pthread
```

配置:

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供

tshell 命令，则包含该命令。

函数接口：

tp 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdTp (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.39. ts – 查看系统线程信息

格式：

ts

ts pid

说明：

该命令有 2 种用法，ts 不跟参数，会显示所有进程中的线程信息；带参数会显示相应进程下的线程信息。

返回值：

执行成功返回 0，失败返回 -1。

备注：

无。

样例：

```
[root@sylixos_station:/root]# ts
thread show >>

  NAME      TID    PID  PRI  STAT  ERRNO    DELAY  PAGEFAILS  FPU  CPU
-----
t_idle0    4010000    0 255  RDY      0         0         0         0
t_itimer   4010001    0  20  SLP      0         4         0         0
t_isrdefer 4010002    0  0  SEM      0         0         0         0
t_except   4010003    0  0  SEM      0         0         0         0
t_log      4010004    0  60  MSGQ     0         0         0         0
t_power    4010005    0 254  RDY      0         0         0         0
t_hotplug  4010006    0 250  RDY     506         0         0         0
t_reclaim  4010008    0 253  MSGQ     0         0         0         0
t_netjob   4010009    0 110  SEM      0         0         0         0
t_netproto 401000a    0 110  MSGQ     506         5         0         0
t_tftpd    401000b    0 160  MSGQ      2         0         0         0
t_ftpd     401000c    0 160  MSGQ      2         0         0         0
t_telnetd  401000d    0 160  MSGQ      2         0         0         0
t_tshell   401000f    0 150  RDY    1503         0         0         0
t_ptyserver 4010010    0 160  JOIN      0         0         0         0
t_ptyproc  4010011    0 150  RDY    1503         0         0         0
t_tshell   4010012    0 150  JOIN      71         0         0         0
test1      4010013    1 200  JOIN      0         0        17         0
pthread    4010014    1 245  RDY      0         0         0         0
pthread    4010015    1 243  RDY      0         0         0         0

thread: 20
[root@sylixos_station:/root]#
```

配置：

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

函数接口：

ts 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdTs (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.40. tty – 显示当前 shell 终端对应的 tty 文件

格式:

tty

说明:

该命令用来显示当前 shell 终端对应的 tty 文件。

返回值:

执行返回 0。

备注:

无。

样例:

```
[root@sylixos_station:/root]# tty
/dev/ttyS0
[root@sylixos_station:/root]#
```

配置:

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

函数接口:

tty 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellSysCmdTty (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.41. vardel - 删除一个指定的系统环境变量

格式:

vardel variable

说明:

该命令用来删除指定的系统环境变量。

返回值:

执行成功返回 0，失败返回非 0 值。

备注:

无。

样例:

查看有那些环境变量

```
[root@sylixos_station:/root]# vars
variable show >>
-----
VARIABLE      REF      VALUE
-----
MYPATH        /home/
TERMCAP       /etc/termcap
TERM          vt100
LUA_CPATH     ?.so;/usr/local/lib/lua/?.so;/usr/lib/lua/?.so;/lib/lua/?.so
```

删除环境变量并在此查看

```
[root@sylixos_station:/root]# vardel MYPATH
[root@sylixos_station:/root]# vars
variable show >>

```

VARIABLE	REF	VALUE
TERMCAP	/etc/termcap	
TERM	vt100	
LUA_CPATH	?.so;/usr/local/lib/lua/?.so;/usr/lib/lua/?.so;/lib/lua/?.so	

#### 配置:

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

#### 函数接口:

vardel 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdVardel (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.42. varload - 从指定参数的文件中提取装载环境变量表

#### 格式:

```
varload
```

```
varload profile
```

#### 说明:

该命令有 2 种用法，不带参数默认从/etc/profile 文件中提取；带参数则从指定的文件中提取环境变量。

#### 返回值:

执行成功返回 0，失败返回-1。

#### 备注:

无。

#### 样例:

查看当前的环境变量

```
[root@sylixos_station:/root]# vars
variable show >>

```

VARIABLE	REF	VALUE
TERMCAP	/etc/termcap	
TERM	vt100	
LUA_CPATH	?.so;/usr/local/lib/lua/?.so;/usr/lib/lua/?.so;/lib/lua/?.so	
LUA_PATH	?.lua;/usr/local/lib/lua/?.lua;/usr/lib/lua/?.lua;/lib/lua/?.lua	
DEBUG_CPU	-1	
PATH_LOCALE	/usr/share/locale	
LC_ALL		
LANG	C	
LD_LIBRARY_PATH	/usr/lib:/lib:/usr/local/lib	
PATH	/usr/bin:/bin:/usr/pkg/sbin:/sbin:/usr/local/bin	
NFS_CLIENT_PROTO	udp	
NFS_CLIENT_AUTH	AUTH_UNIX	
SYSLOGD_HOST	0.0.0.0:514	
FIO_FLOAT	1	
SO_MEM_PAGES	8192	
TSLIB_CALIBFILE	/etc/pointercal	
TSLIB_TSDEVICE	/dev/input/touch0	
MOUSE	/dev/input/mouse0:/dev/input/touch0	
KEYBOARD	/dev/input/keyboard0	
TZ	CST-8:00:00	
TMPPDIR	/tmp/	
LICENSE	SylixOS license: Commercial & GPL.	
VERSION	1.3.5 (5)	
SYSTEM	SylixOS kernel version: 1.3.5 (5) LongYuan(b)	

```
[root@sylixos_station:/root]#
```

创建一个 path 文件，在文件中写 MYPATH="/home/"，保存。

```
[root@sylixos_station:/root]# vi path
MYPATH="/home/"
~
```

装载命令

```
[root@sylixos_station:/root]# varload path
envionment variables load from path success.
```

再次查看环境变量

VARIABLE	REF	VALUE
MYPATH	/home/	
TERMCAP	/etc/termcap	

配置：

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

函数接口：

varload 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdVarload (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.43. vars - 显示当前的环境变量

格式：

vars

#### 说明:

该命令用来查看当前环境变量。

#### 返回值:

执行返回 0。

#### 备注:

无。

#### 样例:

```
[root@sylixos_station:/root]# vars
variable show >>
-----
VARIABLE      REF      VALUE
-----
TERMCAP        /etc/termcap
TERM           vt100
LUA_CPATH      ?.so;/usr/local/lib/lua/?.so;/usr/lib/lua/?.so;/lib/lua/?.so
LUA_PATH       ?.lua;/usr/local/lib/lua/?.lua;/usr/lib/lua/?.lua;/lib/lua/?.lua
DEBUG_CPU      -1
PATH_LOCALE    /usr/share/locale
LC_ALL
LANG           C
LD_LIBRARY_PATH /usr/lib:/lib:/usr/local/lib
PATH           /usr/bin:/bin:/usr/pkg/sbin:/sbin:/usr/local/bin
NFS_CLIENT_PROTO udp
NFS_CLIENT_AUTH AUTH_UNIX
SYSLOGD_HOST    0.0.0.0:514
FIO_FLOAT       1
SO_MEM_PAGES    8192
TSLIB_CALIBFILE /etc/pointercal
TSLIB_TSDEVICE  /dev/input/touch0
MOUSE           /dev/input/mouse0:/dev/input/touch0
KEYBOARD        /dev/input/keyboard0
TZ              CST-8:00:00
TMPDIR          /tmp/
LICENSE         SylixOS license: Commercial & GPL.
VERSION         1.3.5 (5)
SYSTEM          SylixOS kernel version: 1.3.5 (5) LongYuan(b)
```

#### 配置:

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

#### 函数接口:

vars 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellSysCmdVars (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.44. varsave - 保存当前的操作系统环境变量表

#### 格式:

```
varsave
varsave rofile
```

#### 说明:

该命令有 2 种用法，varsave 不带参数默认将现在的环境变量保存到/etc/profile 文件中；如果带参数则将环境变量保存到相应的文件中。

#### 返回值:

执行成功返回 0，失败返回-1。

#### 备注:

如果 profile 文件不存在，varsave 会先创建该文件，如果存在则将环境变量响应的值修

改。

样例：

```
[root@sylixos_station:/root]# varsave path
environment variables save to path success.
```

查看 path 文件

```
[root@sylixos_station:/root]# cat path
#sylixos environment variables profile.
TERMCAP="/etc/termcap"
TERM="vt100"
LUA_CPATH="?..so;/usr/local/lib/lua/?..so;/usr/lib/lua/?..so;/lib/lua/?..so"
LUA_PATH="?..lua;/usr/local/lib/lua/?..lua;/usr/lib/lua/?..lua;/lib/lua/?..lua"
DEBUG_CPU="-1"
PATH_LOCALE="/usr/share/locale"
LC_ALL=""
LANG="C"
LD_LIBRARY_PATH="/usr/lib:/lib:/usr/local/lib"
PATH="/usr/bin:/bin:/usr/pkg/sbin:/sbin:/usr/local/bin"
NFS_CLIENT_PROTO="udp"
NFS_CLIENT_AUTH="AUTH_UNIX"
SYSLOGD_HOST="0.0.0.0:514"
SO_MEM_PAGES="8192"
TSLIB_CALIBFILE="/etc/pointercal"
TSLIB_TSDEVICE="/dev/input/touch0"
MOUSE="/dev/input/mouse0:/dev/input/touch0"
KEYBOARD="/dev/input/keyboard0"
TZ="CST-8:00:00"
```

配置：

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

函数接口：

varsave 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdVarsave (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.45. virtuals - 显示 vmm 虚拟存储器信息

格式：

virtuals

说明：

该命令用来显示虚拟存储器信息。

返回值：

执行返回 0。

备注：

无。

样例：

```
[root@sylixos:/root]# virtals
vmm virtual area show >>
vmm virtual program from: 0x60000000, size: 0x80000000
vmm virtual ioremap from: 0xe0000000, size: 0x10000000
vmm virtual area usage as follow:

VIRTUAL      SIZE    WRITE  CACHE
-----
60006000      1000  true   true
60007000      7000  true   true
6000e000      1000  true   true
6000f000      3000  true   true
e0000000      1000  true   false
```

#### 配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_VMM\_EN>0 时, 对虚拟内存支持, 该命令将会被包含。

#### 函数接口:

virtals 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellSysCmdVirtuals (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.46. which - 检查参数指定的文件位置

#### 格式:

```
which programfile
```

#### 说明:

该命令在环境变量指定的路径下查找指定的文件所在的位置, 并显示文件所在的路径。

#### 返回值:

执行返回 0。

#### 备注:

无。

#### 样例:

```
[root@sylixos_station:/]# which myusr
/usr/bin/myusr
[root@sylixos_station:/]#
```

#### 配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_MODULELOADER\_EN>0, 需要提供模块装载服务, 该命令将会被包含。

#### 函数接口:

which 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellWhich (INT iArgC, PCHAR *ppcArgV);
```

## 2.2.47. who - 查看当前登录用户身份

#### 格式:

```
who
```

#### 说明:

该命令用来查看当前登录用户信息。



返回值:

执行返回 0。

备注:

无。

样例:

```
[root@sylixos_station:/root]# who
user:root terminal:/dev/ttyS0 uid:0 gid:0 euid:0 egid:0
```

配置:

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

函数接口:

who 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellSysCmdWho (INT iArgC, PCHAR ppcArgV[]);
```

## 2.2.48. zones - 查看操作系统物理页面分区使用情况

格式:

zones

说明:

该命令用来显示 vmm 物理存储器信息。

返回值:

执行返回 0。

备注:

无。

样例:

```
[root@sylixos_station:/root]# zones
vmm physical zone show >>

ZONE PHYSICAL  SIZE  PAGESIZE  PGD  FREEPAGE  DMA  USED
-----
  0 31800000   600000    1000 30bcc000    1536 true   0%
  1 31e00000  2200000    1000 30bcc000    8704 false  0%

ALL-Physical memory size: 64 MBytes (67108864 Bytes)
VMM-Physical memory size: 40 MBytes (41943040 Bytes)
VMM-Physical memory free: 40 MBytes (41943040 Bytes)
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_VMM\_EN>0 时，对虚拟内存支持，该命令将会被包含。

函数接口:

zones 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellSysCmdZones (INT iArgC, PCHAR ppcArgV[]);
```

## 3. 文件命令

### 3.1. 介绍

和文件相关的命令有：

- `cat` - 显示文件内容
- `cd` - 切换当前目录
- `ch` - 改变目录
- `chmod` - 改变文件或目录的访问权限
- `close` - 关闭一个文件
- `cmp` - 比较一个文件
- `cp` - 拷贝一个文件
- `df` - 查看指定目录的文件系统信息
- `dosfslabel` - 查看 `fat` 文件系统卷标
- `dsize` - 计算一个指定的目录包含的所有文件信息
- `fatugid` - 设置 `fat` 文件系统用户与组 `id`
- `fdentrys` - 列出操作系统所有正在操作的文件信息
- `fdisk` - 显示或制作一个磁盘分区表
- `files` - 列出操作系统所有正在操作的文件信息
- `gzip` - 压缩或解压缩一个文件
- `ll` - 显示指定目录下的文件详细信息，默认当前目录
- `ln` - 创建符号链接文件
- `logfileadd` - 向内核日志打印函数加入指定的内核文件描述符
- `logfileclear` - 从内核日志打印文件中清除指定的内核文件描述符
- `logfiles` - 显示内核日志打印文件列表
- `ls` - 显示指定目录下的文件名，默认当前目录
- `mkdir` - 创建目录
- `mkfifo` - 创建一个命名管道，只能在根文件系统设备下创建
- `mkfs` - 格式化指定的磁盘
- `mmaps` - 显示系统 `mmap` 信息
- `mount` - 挂载一个卷
- `msgq` - 显示消息队列的信息
- `mv` - 移到或重命名一个文件
- `open` - 打开一个文件
- `pwd` - 查看当前的工作目录
- `rm` - 删除一个文件
- `rmdir` - 删除一个文件夹
- `showmount` - 查看系统中所有已经挂载的卷
- `shfile` - 执行指定的 `shell` 脚本
- `sync` - 将所有系统缓存的信息写入到物理设备
- `tmpname` - 获得一个可以创建的临时文件名
- `touch` - 创建一个普通文件
- `umount` - 卸载一个卷
- `untar` - 解包或解压缩一个 `tar` 或 `tar.gz` 文件包

- vi - 启动 vi 编辑器
- yaffscmd - 显示、设置和擦除一个块
- zlib - 添加一个.gz 的压缩文件

## 3.2. 命令使用

和文件相关 shell 命令的详细介绍和使用方法，主要有使用格式，说明，备注，使用样例，可裁剪配置，以及响应的接口。

### 3.2.1. cat - 显示文件内容

格式：

cat filename

说明：

该命令用于显示文件中的内容，其中 filename 可以包含文件的路径。

返回值：

执行成功返回 0，失败返回非 0 值。

备注：

无。

样例：

```
[root@sylixos_station:/root]# cat /proc/kernel/affinity
      NAME      TID    PID  CPU
-----
t_idle0        4010000    0    0
t_itimer        4010001    0    *
t_isrdefer      4010002    0    *
t_except        4010003    0    *
t_log           4010004    0    *
t_power         4010005    0    *
t_hotplug       4010006    0    *
t_reclaim       4010008    0    *
t_netjob        4010009    0    *
t_netproto      401000a    0    *
t_tftpd         401000b    0    *
t_ftpd          401000c    0    *
t_telnetd       401000d    0    *
t_tshell        401000f    0    *

[root@sylixos_station:/root]#
```

配置：

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_MAX\_VOLUMES>0)，系统同时支持的卷的数量>0 时，该命令将会被包含。

函数接口：

cat 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellFsCmdCat (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.2. cd - 切换当前目录

格式：

cd path

**说明:**

该命令用于切换到指定的目录下。

**返回值:**

执行成功返回 0，失败返回非 0 值。

**备注:**

无。

**样例:**

```
[root@sylixos_station:/root]# cd /
[root@sylixos_station:/]# cd /root/
[root@sylixos_station:/root]#
```

**配置:**

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_MAX\_VOLUMES>0), 系统同时支持的卷的数量>0 时, 该命令将会被包含。

**函数接口:**

cd 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellFsCmdCd (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.3. ch – 改变目录

**格式:**

ch dir

**说明:**

该命令用于切换到指定的目录下。

**返回值:**

执行成功返回 0，失败返回非 0 值。

**备注:**

无。

**样例:**

```
[root@sylixos_station:/]# ch /lib/modules/
[root@sylixos_station:/lib/modules]#
```

**配置:**

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_MAX\_VOLUMES>0), 系统同时支持的卷的数量>0 时, 该命令将会被包含。

**函数接口:**

ch 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellFsCmdCh (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.4. chmod – 改变文件或目录的访问权限

**格式:**

chmod newmode filename

**说明:**

该命令用来改变文件或目录的访问权限。

**返回值:**

执行成功返回 0，失败返回-1。

备注：

无。

样例：

```
[root@sylixos_station:/root]# ll
-rw-rw-rw- root    root    Thu Nov 17 13:38:35 2016    16 B, a
-rw-r--r-- root    root    Thu Nov 17 13:57:37 2016    653 B, path
total items: 2
[root@sylixos_station:/root]# chmod 777 a
[root@sylixos_station:/root]# chmod 666 path
[root@sylixos_station:/root]# ll
-rwxrwxrwx root    root    Thu Nov 17 13:38:35 2016    16 B, a
-rw-rw-rw- root    root    Thu Nov 17 13:57:37 2016    653 B, path
total items: 2
[root@sylixos_station:/root]#
```

配置：

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_MAX\_VOLUMES>0)，系统同时支持的卷的数量>0 时，该命令将会被包含。

函数接口：

chmod 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellFsCmdChmod (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.5. close – 关闭一个文件

格式：

```
close fd
```

说明：

该命令用来关闭一个打开的文件。

返回值：

执行成功返回 0，失败返回-1。

备注：

无。

样例：

```
[root@sylixos_station:/root]# open a 2
open file return: 7 dev 3062d1d4 inode 303 size 0
[root@sylixos_station:/root]# close 7
[root@sylixos_station:/root]#
```

配置：

该命令属于系统提供的 tshell 命令。当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。

函数接口：

close 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdClose (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.6. cmp – 比较一个文件

格式：

```
cmp file one file two
```

说明：

该命令用来比较两个文件是否相同。

返回值:

执行成功返回 0，失败返回-1。

备注:

无。

样例:

```
[root@sylixos_station:/root]# ls
c          b          a
[root@sylixos_station:/root]# cmp a b
file same!
[root@sylixos_station:/root]# cmp a c
file not same!
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_MAX\_VOLUMES>0), 系统同时支持的卷的数量>0 时, 该命令将会被包含。

函数接口:

cmp 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellFsCmdCmp (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.7. cp – 拷贝一个文件

格式:

```
cp src file name dst file name
```

说明:

该命令用来将 src file 内容拷贝到 dst file 中。

返回值:

执行成功返回 0，失败返回-1。

备注:

如果 dst file 不存在会先创建在拷贝。如果 dst file 存在会询问是否覆盖 dst file，如果选择不覆盖则此次拷贝失败。

样例:

```
[root@sylixos_station:/root]# cp a c
copy complete. size:653(Bytes) time:0(s) speed:653(Bps)
[root@sylixos_station:/root]# cp c a
destination file is exist, overwrite? (Y/N)
N
[root@sylixos_station:/root]# cp c a
destination file is exist, overwrite? (Y/N)
Y
copy complete. size:653(Bytes) time:0(s) speed:653(Bps)
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_MAX\_VOLUMES>0), 系统同时支持的卷的数量>0 时, 该命令将会被包含。

函数接口:

cp 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellFsCmdCp (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.8. df - 查看指定目录的文件系统信息

格式:

df volume name

说明:

该命令查看指定目录下的文件系统信息。

返回值:

执行成功返回 0，失败返回-1。

备注:

无。

样例:

```
[root@sylixos_station:/root]# df /
  VOLUME      TOTAL    FREE   USED RO      FS TYPE
-----
/          3.09KB    0.00KB 100% n  ROOT FileSystem
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_MAX\_VOLUMES>0), 系统同时支持的卷的数量>0 时, 该命令将会被包含。

函数接口:

df 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellFsCmdDf (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.9. dosfslabel - 查看或设置文件系统卷标

格式:

dosfslabel vol

dosfslabel vol vol newlabel

说明:

该命令只有 2 种用法, 带参数 vol 用于查看相应卷标的系统

返回值:

执行成功返回 0, 失败返回非 0 值

备注:

无。

样例:

```
[root@sylixos:/media]# dosfslabel hdd0 hdd
[root@sylixos:/media]# dosfslabel hdd0
HDD
[root@sylixos:/media]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_MAX\_VOLUMES>0) 且 LW\_CFG\_FATFS\_EN > 0 时, 支持的卷的数量>0 并且允许小型 FAT 文件系统时, 该命令将会被包含。

函数接口:

dosfslabel 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellFsCmdDosfslabel (INT iArgC, PCHAR ppcArgV[])
```

### 3.2.10. dsize – 计算一个指定的目录包含的所有文件信息

格式：

```
dsize pathname
```

说明：

该命令用来计算指定目录包含文件的信息，包含文件数和文件的总大小。

返回值：

执行成功返回 0，失败返回非 0 值。

备注：

无。

样例：

```
[root@sylixos_station:/root]# dsize /
scanning...
total file 104 size 779250 bytes.
[root@sylixos_station:/root]#
```

配置：

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_MAX\_VOLUMES>0)，系统同时支持的卷的数量>0 时，该命令将会被包含。

函数接口：

dsize 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellFsCmdDsize (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.11. fatugid - 设置 fat 文件系统用户与组 id

格式：

```
fatugid uid gid
```

说明：

该命令用于设置 fat 文件系统用户和组 id 。

返回值：

执行成功返回 0，失败返回非 0 值。

备注：

无。

样例：

```
[root@sylixos_station:/root]# fatugid 0 0
[root@sylixos_station:/root]#
```

配置：

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_MAX\_VOLUMES>0)且 LW\_CFG\_FATFS\_EN > 0 时，支持的卷的数量>0 并且允许小型 FAT 文件系统时，该命令将会被包含。

函数接口：

fatugid 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellFatUGID (INT iArgC, PCHAR ppcArgV[]);
```



### 3.2.12. fdentrys – 列出操作系统所有正在操作的文件信息

格式:

fdentrys

说明:

该命令用于显示操作系统所有正在操作的文件信息。

返回值:

执行返回 0。

备注:

所有的文件信息包括进程打开的文件。

样例:

```
[root@sylixos_station:/root]# fdentrys
all file entry show >>
ref abn name                      real                      type    drv
1    /dev/socket                  /dev/socket             socket   31
1    /dev/socket                  /dev/socket             socket   31
1    /dev/socket                  /dev/socket             socket   31
1    /dev/ttyS0                   /dev/ttyS0              orig     17
[root@sylixos_station:/root]#
```

配置:

该命令属于系统提供的 tshell 命令。当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。

函数接口:

fdentrys 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellSysCmdFdentrys (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.13. fdisk – 显示或制作一个磁盘分区表

格式:

fdisk block I/O device

fdisk -f block I/O device

说明:

该命令只有 2 种用法，当后面不跟选项-f 是显示当前的分区表，跟选项-f 用于制作磁盘分区表。

返回值:

执行成功返回 0，失败返回非 0 值。

备注:

x86 环境测试。

制作磁盘分区表需要先确定该磁盘没有被挂载。使用 showmount 查看是否被挂载，如挂载使用 umount 卸载。

分区完毕需要重新启动并将分好的每个区进行初始化才可以使用。

样例:

```
[root@sylixos:/root]# fdisk /dev/blk/hdd-0
block device: /dev/blk/hdd-0 partition >>

PART ACT  SIZE(KB)  OFFSET(KB)  TYPE
-----
0 *      261120    1024 SylixOS True Power Safe Partition

total partition 1
[root@sylixos:/root]#
```

```
[root@sylixos:/root]# fdisk -f /dev/blk/hdd-0
block device /dev/blk/hdd-0 total size: 256 (MB)
please input how many partition(s) you want to make (1 ~ 4) : 2
please input how many bytes align (4K 8K ...) : 4096
please input the partition 0 size percentage(%) 0 means all left space : 20
is this partition active(y/n) : y
please input the file system type
1: FAT 2: TPSFS 3: LINUX 4: RESERVED
1
please input the partition 1 size percentage(%) 0 means all left space : 0
is this partition active(y/n) : n
please input the file system type
1: FAT 2: TPSFS 3: LINUX 4: RESERVED
2
making partition...
block device: /dev/blk/hdd-0 partition >>

PART ACT  SIZE(KB)  OFFSET(KB)  TYPE
-----
0 *      52224    1024 Win95 FAT32 Partition
1      208896    53248 SylixOS True Power Safe Partition

total partition 2
[root@sylixos:/root]#
```

#### 配置：

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_MAX\_VOLUMES>0)且 LW\_CFG\_OEMDISK\_EN> 0 时，支持的卷的数量>0 并且需要自动挂载与分区工具，该命令将会被包含。

#### 函数接口：

fdisk 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellFsCmdFdisk (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.14. files -列出操作系统所有正在操作的文件信息

#### 格式：

files

#### 说明：

该命令用来显示操作系统正在操作的文件信息。

#### 返回值：

执行返回 0。

#### 备注：

该命令会显示内核文件，而不会显示线程打开的文件。

#### 样例：

```
[root@sylixos_station:/dev/blk]# files
kernel filedes show (process filedes in /proc/${pid}/filedes) >>
fd abn name type drv
3 /dev/ttyS0 orig 17 GLB STD_IN GLB STD_OUT GLB STD_ERR
4 /dev/socket socket 31
5 /dev/socket socket 31
6 /dev/socket socket 31
[root@sylixos_station:/dev/blk]#
```

#### 配置:

该命令属于系统提供的 tshell 命令。当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。

#### 函数接口:

ch 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdFdentrys (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.15. gzip - 压缩或解压缩一个文件

#### 格式:

```
gzip [-c] [-d] [-f] [-h] [-r] [-1 to -9] [files...]
```

#### 说明:

该命令用于压缩或解压一个文件，其中参数有：

- c 将输出写到标准输出上，并保留原有文件；
- d 将压缩文件解压；
- f Huffman 编码和字符串匹配结合；
- h 仅仅使用 Huffman 压缩算法压缩文件；
- r 使用 RLE 压缩算法压缩文件；
- 1 to -9 指定压缩的等级。

#### 返回值:

执行成功返回 0，失败返回非 0 值。

#### 备注:

无

#### 样例:

```
[root@sylixos_station:/root]# ll
-rw-r--r-- root root Thu Nov 24 12:18:05 2016 1251 B, b
-rw-r--r-- root root Thu Nov 24 12:17:57 2016 1251 B, a
total items: 2
[root@sylixos_station:/root]# gzip -r -3 a
[root@sylixos_station:/root]# gzip -f -3 b
[root@sylixos_station:/root]# ll
-rw-rw-rw- root root Thu Nov 24 12:18:38 2016 449 B, b.gz
-rw-rw-rw- root root Thu Nov 24 12:18:29 2016 697 B, a.gz
total items: 2
[root@sylixos_station:/root]# gzip -d a
[root@sylixos_station:/root]#
```

#### 配置:

该命令属于系统提供的 tshell 命令。当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。

#### 函数接口:

gzip 命令是通过 c 语言函数实现的，函数原型是：

```
int minigzip_main(int argc, char *argv[]);
```

### 3.2.16. ll - 显示指定目录下的文件详细信息

格式：

ll

ll [path name]

说明：

该命令用于显示指定目录下文件的详细信息。当不跟参数时，默认会显示当前目录下的文件信息；当跟上参数路径时，会显示相应目录下的文件信息。

返回值：

执行成功返回 0，失败返回-1。

备注：

无。

样例：

```
[root@sylixos_station:/]# ll
lrwxr-xr-- root    root    Fri Nov 18 10:18:12 2016    tmp -> /yaffs2/n1/tmp
lrwxr-xr-- root    root    Fri Nov 18 10:18:12 2016    var -> /yaffs2/n1/var
lrwxr-xr-- root    root    Fri Nov 18 10:18:12 2016    root -> /yaffs2/n1/root
lrwxr-xr-- root    root    Fri Nov 18 10:18:12 2016    home -> /yaffs2/n1/home
lrwxr-xr-- root    root    Fri Nov 18 10:18:12 2016    apps -> /yaffs2/n1/apps
lrwxr-xr-- root    root    Fri Nov 18 10:18:12 2016    sbin -> /yaffs2/n1/sbin
lrwxr-xr-- root    root    Fri Nov 18 10:18:12 2016    bin -> /yaffs2/n1/bin
lrwxr-xr-- root    root    Fri Nov 18 10:18:12 2016    usr -> /yaffs2/n1/usr
lrwxr-xr-- root    root    Fri Nov 18 10:18:12 2016    lib -> /yaffs2/n1/lib
lrwxr-xr-- root    root    Fri Nov 18 10:18:12 2016    qt -> /yaffs2/n1/qt
lrwxr-xr-- root    root    Fri Nov 18 10:18:12 2016    ftk -> /yaffs2/n1/ftk
lrwxr-xr-- root    root    Fri Nov 18 10:18:12 2016    etc -> /yaffs2/n0/etc
lrwxr-xr-- root    root    Fri Nov 18 10:18:12 2016    boot -> /yaffs2/n0/boot
drwxr-xr-- root    root    Fri Nov 18 10:18:12 2016    usb/
drw-rw-rw- root    root    Fri Nov 18 10:18:12 2016    yaffs2/
drw-r--r-- root    root    Fri Nov 18 10:18:12 2016    proc/
drwxr-xr-- root    root    Fri Nov 18 10:18:12 2016    media/
drwxr-xr-- root    root    Fri Nov 18 10:18:12 2016    mnt/
drwxr-xr-- root    root    Fri Nov 18 10:18:12 2016    dev/

total items: 19
[root@sylixos_station:/]#
```

配置：

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_MAX\_VOLUMES>0)，系统同时支持的卷的数量>0 时，该命令将会被包含。

函数接口：

ll 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellFsCmdLI (INT iArgC, PCHAR ppcArgV[]);
```

备注：

### 3.2.17. ln - 创建符号链接文件

格式：

ln [-s | -f] actualpath sympath

说明：

该命令有用来创建符号连接文件，其中默认带参数-s。actualpath 为实际路径，symoath 为链接文件名。

返回值:

执行成功返回 0，失败返回非 0 值。

备注:

无。

样例:

```
[root@sylixos_station:/]# ln -s /yaffs2/n0/lost+found/ lost
[root@sylixos_station:/]# ll
lrw-rw-rw- root root Thu Nov 24 13:38:54 2016 lost -> /yaffs2/n0/lost+found/
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_MAX\_VOLUMES>0), 系统同时支持的卷的数量>0 时, 该命令将会被包含。

函数接口:

ln 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellFsCmdLn (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.18. logfileadd - 向内核日志打印函数加入指定的内核文件描述符

格式:

```
logfileadd file descriptor
```

说明:

该命令用于向内核日志函数加入指定的内核文件描述符。

返回值:

执行成功返回 0，失败返回非 0 值。

备注:

无。

样例:

```
[root@sylixos_station:/root]# logfileadd 7
[root@sylixos_station:/root]# logfiles
log fd(s) include :
    1      7
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_LOG\_LIB\_EN>0 时, 允许系统提供日志管理库, 该命令将会被包含。

函数接口:

logfileadd 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellSysCmdLogFileAdd (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.19. logfileclear - 从内核日志打印文件表中清除指定的内核文件描述符

格式:

```
logfileclear file descriptor
```

说明:

该命令用来删除内核日志打印文件表中指定的内核文件描述符。

返回值:

执行成功返回 0，失败返回-1。

备注:

无。

样例:

```
[root@sylixos_station:/root]# logfileclear 7
[root@sylixos_station:/root]# logfiles
log fd(s) include :
    1
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_LOG\_LIB\_EN>0 时，允许系统提供日志管理库，该命令将会被包含。

函数接口:

logfileclear 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellSysCmdLogFileClear (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.20. logfiles - 显示内核日志打印文件列表

格式:

logfiles

说明:

该命令用来显示内核日志打印文件列表。

返回值:

执行成功返回 0，失败返回非 0 值。

备注:

无。

样例:

```
[root@sylixos_station:/root]# logfiles
log fd(s) include :
    1
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_LOG\_LIB\_EN>0 时，允许系统提供日志管理库，该命令将会被包含。

函数接口:

logfiles 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellSysCmdLogFiles (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.21. ls – 显示指定目录下的文件名，默认当前目录

格式:

ls

ls path name

说明:

该命令用来显示指定目录下的文件名。不带参数默认打印当前当前目录下的文件名；带

路径则打印相应目录下的文件名列表。

**返回值：**

执行成功返回 0，失败返回-1。

**备注：**

无。

**样例：**

```
[root@sylixos_station:/root]# ls
hello      test
[root@sylixos_station:/root]# mkdir test2
[root@sylixos_station:/root]# ls
test2      hello      test
[root@sylixos_station:/root]#
```

**配置：**

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_MAX\_VOLUMES>0)，系统同时支持的卷的数量>0 时，该命令将会被包含。

**函数接口：**

ls 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellFsCmdLls(INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.22. mkdir - 创建目录

**格式：**

```
mkdir directory
```

**说明：**

该命令用来在相应目录下创建一个文件夹，默认当前路径。

**返回值：**

执行成功返回 0，失败返回非 0 值。

**备注：**

directory 参数可以带路径，将在指定的路径下创建文件夹。

**样例：**

```
[root@sylixos_station:/root]# mkdir dir
[root@sylixos_station:/root]# ls
dir
[root@sylixos_station:/root]#
```

**配置：**

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_MAX\_VOLUMES>0)，系统同时支持的卷的数量>0 时，该命令将会被包含。

**函数接口：**

mkdir 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellFsCmdMkdir (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.23. mkfifo - 创建一个命名管道，只能在根文件系统设备下创建

**格式：**

```
mkfifo fifo name
```

**说明：**

该命令在根目录下创建一个命名管道。

返回值:

执行成功返回 0，失败返回非 0 值。

备注:

无。

样例:

```
[root@sylixos_station:/]# mkfifo fif01
[root@sylixos_station:/]# ls
fif01      tmp      var      root     home
apps       sbin     bin      usr      lib
qt         ftk      etc      boot     usb
yaffs2     proc     media    mnt      dev
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_MAX\_VOLUMES>0), 系统同时支持的卷的数量>0 时, 该命令将会被包含。

函数接口:

mkfifo 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellFsCmdMkfifo (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.24. mkfs – 格式化指定的磁盘

格式:

mkfs media name

说明:

该命令用来格式化指定的磁盘。

返回值:

执行成功返回 0，失败返回非 0 值。

备注:

x86 环境测试。

样例:

```
[root@sylixos:/root]# mkfs /media/hdd0/
now format media, please wait...
disk format ok.
[root@sylixos:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_MAX\_VOLUMES>0), 系统同时支持的卷的数量>0 时, 该命令将会被包含。

函数接口:

mkfs 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellFsCmdMkfs (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.25. mmap – 显示系统 mmap 信息

格式:

mmap

说明:

该命令用来显示系统 mmap 信息, 其中信息有: 起始地址 (ADDR)、映射长度 (SIZE)、文件偏移量 (OFFSET)、页面属性, 可写 (WRITE)、映射标识, 共享 (SHARE)、文件描述符



(FD)。

返回值：

执行返回 0。

备注：

无。

样例：

```
[root@sylixos_station:/root]# mmaps
ADDR      SIZE      OFFSET    WRITE SHARE  PID      FD
-----
```

配置：

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_POSIX\_EN>0 和 LW\_CFG\_VMM\_EN>0 时，使能 posix 兼容库并且对对虚拟内存支持，该命令将会被包含。

函数接口：

mmaps 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellMmaps (INT iArgC, PCHAR *ppcArgV);
```

### 3.2.26. mount – 挂载一个卷

格式：

```
mount -t fstype -o option blk dev mount path
```

说明：

该命令用来挂载一个卷。

-t 指定设备的文件系统类型

nfs 网络文件系统

ramfs 基于内存的文件系统

romfs 简单的、紧凑的、只读文件系统

tpsfs SylixOS 针对大容量存储设备集成的文件管理系统

-o 指定读写模式

ro 只读模式挂载

rw 用读写模式挂载。

blk dev 设备名称

mount path 挂载路径

返回值：

执行成功返回 0，失败返回非 0 值。

备注：

x86 环境测试。

样例：

```
[root@sylixos:/root]# mount /dev/blk/hdd-0 /mnt/hdd1
[root@sylixos:/root]# showmount
AUTO-Mount point show >>
      VOLUME                      BLK NAME
-----
/media/hdd0                      /dev/blk/hdd-0:0

Mount point show >>
      VOLUME                      BLK NAME
-----
/mnt/hdd1                       /dev/blk/hdd-0
/tmp                             0
[root@sylixos:/root]#
```

#### 配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_MAX\_VOLUMES>0 且 LW\_CFG\_MOUNT\_EN>0 时, 系统同时支持的卷的数量>0 并且系统需要使用 mount 工具时, 该命令将会被包含。

#### 函数接口:

mount 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellFsCmdMount (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.27. msgq – 显示消息队列的信息

#### 格式:

msgq message queue handle

#### 说明:

该命令用来查看指定消息队列的消息。

#### 返回值:

执行成功返回 0, 失败返回-1。

#### 备注:

无。

#### 样例:

tp 可以查看消息队列句柄

```
[root@sylixos_station:/root]# msgq 1c01007e
MsgQueue show >>

MsgQueue Name      : lwip_msg
MsgQueue Id        : 0x1c01007e
MsgQueue Max Msgs  : 512
MsgQueue N Msgs    : 0
MsgQueue Max Msg Len: 4
Thread Queuing     : FIFO
Pended Threads     : 1

[root@sylixos_station:/root]#
```

#### 配置:

该命令属于系统提供的 tshell 命令。当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。

#### 函数接口:

msgq 命令是通过 c 语言函数实现的, 函数原型是:

```
tatic INT __tshellSysCmdMsgq (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.28. mv – 移到或重命名一个文件

格式:

```
mv SRC file name DST file name
```

说明:

该命令只有 1 种用法，如果两者在同一个目录下将 src file 重命名为 dst file，不在同一目录下则将 srcfile 移到到 dst 的目录下并命名为 dst。

返回值:

执行成功返回 0，失败返回非 0 值。

备注:

mv 会监测目标文件是否存在，存在给出提示是否覆盖，若覆盖则移动，否则不移到。

样例:

```
[root@sylixos_station:/root]# cp a /home/b
copy complete. size:1013(Bytes) time:0(s) speed:1013(Bps)
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_MAX\_VOLUMES>0)，系统同时支持的卷的数量>0 时，该命令将会被包含。

函数接口:

mv 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellFsCmdMv (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.29. open – 打开一个文件

格式:

```
open filename flag
```

```
open filename flag mode
```

说明:

该命令用来以不用权限打开或创建一个文件。

返回值:

执行成功返回 0，失败返回非 0 值。

备注:

flag 可选项为:

O\_RDONLY : 00000000

O\_WRONLY : 00000001

O\_RDWR : 00000002

O\_APPEND : 00000008

O\_SHLOCK : 00000010

O\_EXLOCK : 00000020

O\_ASYNC : 00000040

O\_CREAT : 00000200

O\_TRUNC : 00000400

O\_EXCL : 00000800

O\_SYNC : 00002000

```
O_NONBLOCK : 00004000
O_NOCTTY   : 00008000
O_CLOEXEC  : 00080000。
```

样例：

```
[root@sylixos_station:/root]# open a 8
open file return: 7 dev 3062d1d4 inode 10f size 3f5
[root@sylixos_station:/root]#
```

配置：

该命令属于系统提供的 tshell 命令。当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。

函数接口：

open 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdOpen (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.30. pwd – 查看当前的工作目录

格式：

```
pwd
```

说明：

该命令用来显示当前的工作目录。

返回值：

执行返回 0。

备注：

无。

样例：

```
[root@sylixos_station:/root]# pwd
/root
[root@sylixos_station:/root]#
```

配置：

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_MAX\_VOLUMES>0)，系统同时支持的卷的数量>0 时，该命令将会被包含。

函数接口：

cpus 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellFsCmdPwd (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.31. rm – 删除一个文件

格式：

```
rm file name
```

说明：

该命令用来删除一个文件。

返回值：

执行成功返回 0，失败返回非 0 值。

备注：

无。

样例:

```
[root@sylixos_station:/root]# ls
hello
[root@sylixos_station:/root]# rm hello
[root@sylixos_station:/root]# ls
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_MAX\_VOLUMES>0), 系统同时支持的卷的数量>0 时, 该命令将会被包含。

函数接口:

rm 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellFsCmdRm (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.32. rmdir – 删除一个文件夹

格式:

```
rmdir directory
```

说明:

该命令只有 1 种用法, 用来删除一个给定的文件夹。

返回值:

执行成功返回 0, 失败返回非 0 值。

备注:

只能删除空的文件夹, 当文件夹下有文件或目录时删除失败。

样例:

```
[root@sylixos_station:/root]# ls
hello      a
[root@sylixos_station:/root]# rmdir a
[root@sylixos_station:/root]# ls
hello
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_MAX\_VOLUMES>0), 系统同时支持的卷的数量>0 时, 该命令将会被包含。

函数接口:

rmdir 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellFsCmdMkdir (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.33. showmount – 查看系统中所有已经挂载的卷

格式:

```
showmount
```

说明:

该命令用来查看系统中所有挂载的卷。

返回值:

执行成功返回 0, 失败返回非 0 值。

备注:。

x86 环境测试。

样例:

```
[root@sylixos:/root]# showmount
AUTO-Mount point show >>
      VOLUME                                BLK NAME
-----
/media/hdd0                                /dev/blk/hdd-0:0

Mount point show >>
      VOLUME                                BLK NAME
-----
/tmp                                         0
[root@sylixos:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_MAX\_VOLUMES>0) 且 LW\_CFG\_MOUNT\_EN>0 时, 系统同时支持的卷的数量>0 并且系统需要使用 mount 工具时, 该命令将会被包含。

函数接口:

showmount 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellFsCmdShowmount (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.34. shfile – 执行指定的 shell 脚本

格式:

```
shfile shell file
```

说明:

该命令用来执行指定的 shell 脚本。

返回值:

执行成功返回 0, 失败返回非 0 值。

备注:

无。

样例:

创建一个 shell 脚本文件

```
[root@sylixos_station:/root]# vi test.sh
#!/bin/bash
echo "hello world"
```

使用 shfile 运行脚本

```
[root@sylixos_station:/root]# shfile test.sh
hello world
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_MAX\_VOLUMES>0), 系统同时支持的卷的数量>0 时, 该命令将会被包含。

函数接口:

shfile 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellFsCmdShfile (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.35. sync – 将所有系统缓存的信息写入到物理设备

格式:

sync

说明:

该命令将所有系统缓存的文件、设备、磁盘信息全部写入到相应的物理设备中。

返回值:

执行返回 0。

备注:

无。

样例:

```
[root@sylixos_station:/root]# sync
[root@sylixos_station:/root]#
```

配置:

该命令属于系统提供的 tshell 命令。当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。

函数接口:

sync 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellSysCmdSync (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.36. tmpname – 获得一个可以创建的临时文件名

格式:

tmpname

说明:

该命令用来获得一个可以创建的临时文件名。

返回值:

执行返回 0。

备注:

无。

样例:

```
[root@sylixos_station:/root]# tmpname
can mktmp as name: /tmp/tmp.0.HjNXRI
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_MAX\_VOLUMES>0), 系统同时支持的卷的数量>0 时, 该命令将会被包含。

函数接口:

tmpname 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellFsCmdTmpname (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.37. touch – 创建一个普通文件

格式:

touch file name

```
touch -amc file name
```

#### 说明:

该命令有 2 种用法，直接跟文件名表示创建一个普通的文件；参数可以跟 amc 中的一个或多个的组合，其中 a 表示改变文件的访问时间，m 表示改变修改时间，c 表示如果文件不存在，不创建文件，若不跟参数 c，跟上 a 或 m 时，如果文件不存在会先创建文件。

#### 返回值:

执行成功返回 0，失败返回非 0 值

#### 备注:

参数 a 和 m 功能暂时未实现。

#### 样例:

```
[root@sylixos_station:/tmp/tmp.0.mhA9dG]# ls
a
[root@sylixos_station:/tmp/tmp.0.mhA9dG]# touch temp
[root@sylixos_station:/tmp/tmp.0.mhA9dG]# ls
temp
a
[root@sylixos_station:/tmp/tmp.0.mhA9dG]#
```

#### 配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_MAX\_VOLUMES>0)，系统同时支持的卷的数量>0 时，该命令将会被包含。

#### 函数接口:

touch 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellFsCmdTouch (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.38. umount – 卸载一个卷

#### 格式:

```
umount mount path
```

#### 说明:

该命令用来卸载系统中已经存在的卷。

#### 返回值:

执行成功返回 0，失败返回非 0 值。

#### 备注:

x86 环境测试。

#### 样例:

```
[root@sylixos:/root]# umount /mnt/hdd1
[root@sylixos:/root]# showmount
AUTO-Mount point show >>
VOLUME                                BLK NAME
-----
/media/hdd0                            /dev/blk/hdd-0:0

Mount point show >>
VOLUME                                BLK NAME
-----
/tmp                                    0
[root@sylixos:/root]#
```

#### 配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_MAX\_VOLUMES>



0)且 LW\_CFG\_MOUNT\_EN>0 时，系统同时支持的卷的数量>0 并且系统需要使用 mount 工具时，该命令将会被包含。

**函数接口：**

umount 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellFsCmdUmount (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.39. untar - 解包或解压缩一个 tar 或 tar.gz 文件包

**格式：**

```
untar .tar or .tar.gz file destination directory
```

**说明：**

该命令用来解压一个 .tar.gz 或 .tar 文件.可以指定解压路径。

**返回值：**

执行成功返回 0，失败返回非 0 值。

**备注：**

无。

**样例：**

```
[root@sylixos_station:/home]# untar b.tar.gz
unpackage test.c size: 356 ...
unpackage a.out size: 4754 ...
unpackage total 2 files 0 directory.
[root@sylixos_station:/home]#
```

**配置：**

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_SHELL\_TAR\_EN>0 时，使能 tar 工具后，该命令将会被包含。

**函数接口：**

untar 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellFsCmdUntar (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.40. vi - 启动 vi 编辑器

**格式：**

```
vi [filename]
```

**说明：**

该命令用来打开 vi 编辑器,可以预先指定文件名，也可以编辑好文档时输入文件名。

**返回值：**

执行返回 0。

**备注：**

无。

**样例：**

```
[root@sylixos_station:/home]# vi a
```

**配置:**

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

**函数接口:**

vi 命令是通过 c 语言函数实现的，函数原型是：

```
int vi_main(int argc, char **argv);
```

### 3.2.41. yaffscmd - 显示、设置和擦除一个块

**格式:**

```
yaffscmd  volname  bad
yaffscmd  volname  info
yaffscmd  volname  markbad
yaffscmd  volname  erase
```

**说明:**

该命令有 4 种用法，其中

```
yaffscmd  volname  bad   查看卷中被标记的不良块
yaffscmd  volname  info  查看卷中所有的块
yaffscmd  volname  markbad 标记卷中的不良块
yaffscmd  volname  erase 擦除卷中的块。
```

**返回值:**

执行成功返回 0，失败返回非 0 值。

**备注:**

无。

**样例:**

```
[root@sylixos_station:/home]# yaffscmd n0 info
Device : "/n0"
startBlock..... 1
endBlock..... 128
totalBytesPerChunk. 2048
chunkGroupBits..... 0
chunkGroupSize..... 1
nErasedBlocks..... 126
nReservedBlocks.... 10
nCheckptResBlocks.. nil
blocksInCheckpoint. 0
nObjects..... 10
nTnodes..... 4
nFreeChunks..... 8181

[root@sylixos_station:/home]# yaffscmd n0 markbad 54
yaffs: marking block 84 bad
mark the block 0x54 is a bad ok.
[root@sylixos_station:/home]# yaffscmd n0 bad
block 0x1 is bad block.
block 0x2 is bad block.
block 0x24 is bad block.
block 0x54 is bad block.
[root@sylixos_station:/home]#
```

#### 配置:

该当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 (LW\_CFG\_MAX\_VOLUMES > 0) 且 (LW\_CFG\_YAFFS\_EN > 0)，系统同时支持多个卷并且允许 VAFFS 文件系统，该命令将会被包含。

#### 函数接口:

yaffscmd 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellYaffsCmd (INT iArgC, PCHAR ppcArgV[]);
```

### 3.2.42. zlib -添加一个.gz 的压缩文件

#### 格式:

```
zlib test.zlib
```

#### 说明:

该命令用来添加一个.gz 压缩文件，zlib 的测试接口。

#### 返回值:

执行成功返回 0，失败返回非 0 值

#### 备注:

无。

#### 样例:

```
[root@sylixos_station:/root]# zlib test.gz
zlib version 1.2.8 = 0x1280, compile flags = 0x55
uncompress(): hello, hello!
gzread(): hello, hello!
gzgets() after gzseek: hello!
inflate(): hello, hello!
large_inflate(): OK
after inflateSync(): hello, hello!
inflate with dictionary: hello, hello!
[root@sylixos_station:/root]# ls
test.gz
[root@sylixos_station:/root]#
```

#### 配置:

该命令属于系统提供的 tshell 命令。当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。

#### 函数接口:

zlib 命令是通过 c 语言函数实现的，函数原型是：

```
int zlib_main(int argc, char *argv[]);
```

## 4. 用户命令

### 4.1. 介绍

- gadd – 增加一个新的用户组
- gdel – 删除一个用户组
- group – 显示用户组的信息
- pmod – 修改用户密码
- uadd – 添加用户
- udel – 删除用户
- umod – 设置用户的模式
- user - 显示用户信息或为用户生成密码

### 4.2. 命令使用

#### 4.2.1. gadd – 增加一个新的用户组

##### 格式:

```
gadd group_name gid
```

##### 说明:

该命令用来增加一个用户组并为该组分配一个 gid。

##### 返回值:

执行成功返回 0，失败返回非 0 值。

##### 备注:

无。

##### 样例:

```
[root@sylixos_station:/root]# gadd test 4
[root@sylixos_station:/root]#
```

#### 配置:

当 `LW_CFG_SHELL_EN>0` 时，会允许操作系统提供 `tshell` 命令。当 (`LW_CFG_SHELL_USER_EN > 0`)，使能了 `shell` 用户管理工具，该命令将会被包含。

#### 函数接口:

`gadd` 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellUserCmdGadd (INT iArgC, PCHAR ppcArgV[]);
```

### 4.2.2. `gdel` – 删除一个用户组

#### 格式:

```
gdel group_nam
```

#### 说明:

该命令用来删除一个用户组。

#### 返回值:

执行成功返回 0，失败返回非 0 值。

#### 备注:

只能删除没有用户的空用户组，当用户组不为空时删除失败。

#### 样例:

```
[root@sylixos_station:/root]# gdel test
[root@sylixos_station:/root]#
```

#### 配置:

当 `LW_CFG_SHELL_EN>0` 时，会允许操作系统提供 `tshell` 命令。当 (`LW_CFG_SHELL_USER_EN > 0`)，使能了 `shell` 用户管理工具，该命令将会被包含。

#### 函数接口:

`gdel` 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellUserCmdGdel (INT iArgC, PCHAR ppcArgV[]);
```

### 4.2.3. `group` – 显示用户组的信息

#### 格式:

```
group
```

#### 说明:

该命令用来显示所有用户组的信息。

#### 返回值:

执行成功返回 0，失败返回非 0 值。

#### 备注:

无。

#### 样例:

```
[root@sylixos_station:/root]# group
```

GROUP	GID	USERs
root	0	root,
server	100	server,
user	200	user,
apps	300	user,
anonymous	400	anonymous,

```
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 (LW\_CFG\_SHELL\_USER\_EN>0)，使能了 shell 用户管理工具，该命令将会被包含。

函数接口:

group 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellUserCmdGroup(INT iArgC, PCHAR ppcArgV[]);
```

#### 4.2.4. pmod - 修改用户密码

格式:

```
pmod name old_password new_password
```

说明:

该命令用来修改用户的密码。

返回值:

执行成功返回 0，失败返回非 0 值。

备注:

以 root 用户登录，才可以修改密码。

样例:

```
[root@sylixos_station:/root]# pmod liang liang 1234
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 (LW\_CFG\_SHELL\_USER\_EN>0)，使能了 shell 用户管理工具，该命令将会被包含。

函数接口:

pmod 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellUserCmdPmod(INT iArgC, PCHAR ppcArgV[]);
```

#### 4.2.5. uadd - 添加用户

格式:

```
uadd name password enable[0/1] uid gid comment homedir
```

说明:

该命令用来添加一个用户。

返回值:

执行成功返回 0，失败返回-1。

备注:

用户 id 不能重复，添加用户时必须将其添加到一个存在的用户组。

样例：

```
[root@sylixos_station:/root]# uadd liang liang 1 2 0 liang /home/liang
[root@sylixos_station:/root]# user
```

USER	ENABLE	UID	GID
root	yes	0	0
hanhui	yes	1	0
anonymous	no	400	400
liang	yes	2	0

配置：

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 (LW\_CFG\_SHELL\_USER\_EN > 0)，使能了 shell 用户管理工具，该命令将会被包含。

函数接口：

uadd 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellUserCmdUadd (INT iArgC, PCHAR ppcArgV[]);
```

#### 4.2.6. udel – 删除用户

格式：

```
udel username
```

说明：

该命令用来删除一个用户。

返回值：

执行成功返回 0，失败返回非 0 值。

备注：

无。

样例：

```
[root@sylixos_station:/root]# udel test
[root@sylixos_station:/root]#
```

配置：

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 (LW\_CFG\_SHELL\_USER\_EN > 0)，使能了 shell 用户管理工具，该命令将会被包含。

函数接口：

udel 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellUserCmdUdel (INT iArgC, PCHAR ppcArgV[]);
```

#### 4.2.7. umod – 设置用户的模式

格式：

```
umod name enable[0 / 1] comment homedir
```

说明：

该命令用来设置用户的使能情况、用户描述和用户目录。

返回值：

执行成功返回 0，失败返回非 0 值。

备注:

无。

样例:

```
[root@sylixos_station:/root]# umod liang 0 testuser /home/liang/
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 (LW\_CFG\_SHELL\_USER\_EN > 0)，使能了 shell 用户管理工具，该命令将会被包含。

函数接口:

umod 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellUserCmdUmod(INT iArgC, PCHAR ppcArgV[]);
```

## 4.2.8. user - 显示用户信息或为用户生成密码

格式:

```
user
user genpass
```

说明:

该命令有 2 种用法，无参数时，显示所有用户的用户名、使能情况、用户 id 和所属的组 id；当跟上参数 genpass 时，为用户生成一个密码。

返回值:

执行成功返回 0，失败返回非 0 值。

备注:

无。

样例:

```
[root@sylixos_station:/root]# user
```

USER	ENABLE	UID	GID
root	yes	0	0
hanhui	yes	1	0
anonymous	no	400	400
liang	yes	2	0

```
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_SHELL\_USER\_EN > 0，使能了 shell 用户管理工具，该命令将会被包含。

函数接口:

user 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellUserCmdUser(INT iArgC, PCHAR ppcArgV[]);
```



## 5. 网络

### 5.1. 介绍

和网络相关的命令有：

- aodvs – 显示 aodvs 路由表
- arp – 添加、删除或查看 ARP 表
- ftpdpath – 查看或设置 ftp 服务器初始化路径
- ftpds – 显示 ftp 服务器信息
- hosttable - 查看、添加或删除主机地址映射
- ifonfig - 显示或配置网络配置信息
- ifdown – 禁用一个网络接口
- ifrouter – 设置默认路由接口
- ifup – 启用一个网络接口
- ipv6 – 设置或显示 ipv6
- nat – 启动、关闭或设置 NAT 虚拟网络地址服务
- natalias
- natmap
- nats – 查看当前 NAT 虚拟地址服务状态
- nbname – 显示或设置本机的 NetBIOS 的名字
- netstat – 查看网络状态
- npfattach - 在指定网络接口上使能网络数据包过滤器
- npfdetach - 在指定网络接口上禁能网络数据包过滤器
- npfruleadd – 添加一条网络数据包过滤器规则
- npfruledel – 删除一条网络数据包过滤器规则
- npfs – 查看网络数据包过滤器状态
- ping - Ping 命令
- ping6 – IPv6 Ping 命令
- route – 添加、删除、修改或查看系统路由表
- tftp - 使用 tftp 命令接收或者发送一个文件
- tftpdpath - 查看或设置 tftp 服务器本地路径
- vlan - 显示、设置和删除 net 接口
- vpnclose - 删除一个虚拟网络接口
- vpnopen - 创建一个虚拟网络接口

### 5.2. 命令使用

#### 5.2.1. aodvs – 显示 aodv 路由表

格式：

aodvs

说明：

该命令用来显示 aodvs 路由表。

返回值：

执行成功返回 0，失败返回-1。

备注:

无。

样例:

```
[root@sylixos_station:/root]# aodvs
aodv routing tables
Destination      Gateway          Mask            Flag            Hops            Interface
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_NET\_EN> 0), 允许提供网络功能时, 该命令将会被包含。

函数接口:

aodvs 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellAodvs (INT iArgC, PCHAR *ppcArgV);
```

## 5.2.2. arp – 添加、删除或查看 ARP 表

格式:

```
arp -a
```

```
arp -s inet_address physical_address
```

```
arp -d inet_address
```

说明:

该命令有 3 种用法, 其中 arp -a 用于查看 ARP 表; arp -s inet\_address physical\_address 用于增加一个地址到 ARP 表中; arp -d inet\_address 用于删除一个存在 ARP 表中的地址。

返回值:

执行成功返回 0, 失败返回非 0 值。

备注:

arp 表信息存放在 /proc/net/arp 文件下。

样例:

```
[root@sylixos_station:/root]# arp -a
FACE INET ADDRESS      PHYSICAL ADDRESS TYPE
[root@sylixos_station:/root]# arp -s 192.168.7.60 00:11:22:33:44:55
[root@sylixos_station:/root]# arp -a
FACE INET ADDRESS      PHYSICAL ADDRESS TYPE
en1 192.168.7.60       00:11:22:33:44:55 static
[root@sylixos_station:/root]# arp -d 192.168.7.60
[root@sylixos_station:/root]# arp -a
FACE INET ADDRESS      PHYSICAL ADDRESS TYPE
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_NET\_EN> 0, 允许提供网络功能时, 该命令将会被包含。

函数接口:

arp 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellArp (INT iArgC, PCHAR *ppcArgV);
```

### 5.2.3. ftpdpath – 查看或设置 ftp 服务器初始化路径

格式:

```
ftpdpath
ftpdpath new path
```

说明:

该命令有 2 种用法, 无参数时, 显示 ftp 服务器的初始化路径; 当跟上参数 genpass 时, 设置 ftp 的路径。

返回值:

执行成功返回 0, 失败返回非 0 值。

备注:

无。

样例:

```
[root@sylixos_station:/root]# ftpdpath
ftpd path: /proc/
[root@sylixos_station:/root]# ftpdpath /
[root@sylixos_station:/root]# ftpdpath
ftpd path: /
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当(LW\_CFG\_NET\_EN > 0) 和 (LW\_CFG\_NET\_FTPD\_EN > 0), 允许提供网络功能且使能了 ftp 服务器, 该命令将会被包含。

函数接口:

ftpdpath 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellNetFtpdPath (INT iArgC, PCHAR ppcArgV[]);
```

### 5.2.4. ftpds – 显示 ftp 服务器信息

格式:

```
ftpds
```

说明:

该命令用来显示 ftp 服务器的信息。

返回值:

执行返回 0。

备注:

无。

样例:

```
[root@sylixos_station:/root]# ftpds
ftpd show >>
ftpd path: /

REMOTE                TIME                ALIVE (s)
-----
total ftp session: 0
[root@sylixos_station:/root]#
```

#### 配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当(LW\_CFG\_NET\_EN > 0) 和 (LW\_CFG\_NET\_FTPD\_EN > 0), 允许提供网络功能且使能了 ftp 服务器, 该命令将会被包含。

#### 函数接口:

ftpdpath 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellNetFtpdShow (INT iArgC, PCHAR ppcArgV[]);
```

### 5.2.5. hosttable - 查看、添加或删除主机地址映射

#### 格式:

```
hosttable
hosttable -s host addr
hosttable -d host
```

#### 说明:

该命令有 3 种用法, 无参数是显示主机的地址映射关系; 参数-s 表示添加地址映射关系, 后面跟主机名和地址; 参数-d 用于删除地址映射关系, 后面跟主机名。

#### 返回值:

执行成功返回 0, 失败返回非 0 值。

#### 备注:

无。

#### 样例:

```
[root@sylixos_station:/root]# hosttable -s localhost 127.0.0.1
[root@sylixos_station:/root]# hosttable

  addr                HOST
-----
127.0.0.1            localhost
[root@sylixos_station:/root]# hosttable -d localhost
[root@sylixos_station:/root]# hosttable

  addr                HOST
-----
[root@sylixos_station:/root]#
```

#### 配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_NET\_EN>0), 允许提供网络功能时, 该命令将会被包含。

#### 函数接口:

hosttable 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellHostTable (INT iArgC, PCHAR ppcArgV[]);
```

### 5.2.6. ifconfig - 显示或配置网络配置信息

#### 格式:

```
ifconfig
ifconfig netifname
ifconfig netifname inet address
ifconfig netifname netmask address
```

```
ifconfig netifname gateway address
```

```
ifconfig dns 0 address
```

#### 说明:

该命令有 5 种用法，无参数是显示所有的网络配置的信息；后面跟网口的名字显示指定的网口的配置信息；inet 用于配置 ip 地址，netmask 用于配置子网掩码，gateway 配置默认网关。dns 0 用于配置首选 dns 服务器，dns 1 用于配置备用的 dns 服务器。

#### 返回值:

执行返回 0。

#### 备注:

无。

#### 样例:

```
[root@sylixos_station:/root]# ifconfig en1 inet 192.168.7.32
[root@sylixos_station:/root]# ifconfig en1 netmask 255.255.255.0
[root@sylixos_station:/root]# ifconfig en1 gateway 192.168.7.1
[root@sylixos_station:/root]# ifconfig dns 192.168.1.254
[root@sylixos_station:/root]# ifconfig en1
en1      enable: true linkup: true MTU: 1500 multicast: false
metric: 1 type: Ethernet-Cap HWaddr: 08:08:3E:26:0A:5A
DHCP: Disable(Off) speed: 100(Mbps)
inet addr: 192.168.7.32 netmask: 255.255.255.0
gateway: 192.168.7.1 broadcast: 192.168.7.255
inet6 addr: FE80::A08:3EFF:FE26:A5A Scope:link <valid>
RX ucast packets:628 nucast packets:0 dropped:0
TX ucast packets:9 nucast packets:0 dropped:0
RX bytes:61752 TX bytes:594
[root@sylixos_station:/root]#
```

#### 配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_NET\_EN>0)，允许提供网络功能时，该命令将会被包含。

#### 函数接口:

ifconfig 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellifconfig(INT iArgC, PCHAR *ppcArgV);
```

### 5.2.7. ifdown – 禁用一个网络接口

#### 格式:

```
ifdown netifname
```

#### 说明:

该命令用来禁用指定的网络端口。

#### 返回值:

执行成功返回 0，失败返回非 0 值。

#### 备注:

无。

#### 样例:

```
[root@sylixos_station:/root]# ifdown en1
net interface "en1" set down.
[root@sylixos_station:/root]#
```

#### 配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_NET\_EN> 0), 允许提供网络功能时, 该命令将会被包含。

#### 函数接口:

ifdown 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellIfDown (INT iArgC, PCHAR *ppcArgV);
```

### 5.2.8. ifrouter – 设置默认路由接口

#### 格式:

```
ifconfig netifname
```

#### 说明:

该命令用来设置默认的路由接口。

#### 返回值:

执行返回 0。

#### 备注:

该命令已经被 route 替换。

#### 样例:

无。

#### 配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_NET\_EN> 0), 允许提供网络功能时, 该命令将会被包含。

#### 函数接口:

ifrouter 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellIfRouter (INT iArgC, PCHAR *ppcArgV);
```

### 5.2.9. ifup – 启用一个网络接口

#### 格式:

```
ifup netifname
```

```
ifup netifname -dhcp
```

```
ifup netifname -nodhcp
```

#### 说明:

该命令有 3 种用法, 其中只跟网络接口, 用于启用一个网络; 后面跟参数-dhcp, 指明使用 dhcp 客服得到网络地址 (自动获取 ip 地址); 后面跟参数-nodhcp, 表面不使用 dhcp 获得 ip 地址。

#### 返回值:

执行成功返回 0, 失败返回非 0 值。

#### 备注:

- 1.ifup netifname 启用网络接口, 关于是否使用 dhcp 依赖于上次是否使用了 dhcp。
- 2.dhcp 的使用需要配置 LWIP\_DHCP> 0, 是否允许 dhcp 协议。

#### 样例:

```
[root@sylixos_station:/root]# ifup en1
DHCP client starting...
DHCP client start.
net interface "en1" set up.
[root@sylixos_station:/root]# ifup en1 -nodhcp
net interface "en1" set up.
[root@sylixos_station:/root]#
```

#### 配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_NET\_EN> 0), 允许提供网络功能时, 该命令将会被包含。

#### 函数接口:

ifup 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellIfUp (INT iArgC, PCHAR *ppcArgV);
```

## 5.2.10. ipv6 – 设置或显示 ipv6

#### 格式:

```
ipv6 address ifname address%prefixlen
```

```
ipv6 noaddress ifname address%prefixlen
```

#### 说明:

该命令有 2 种用法。后面跟参数 address, 给 ifname 设置 ipv6 的地址; 后面跟参数 noaddress, 删除接口中此个 ipv6 地址。

#### 返回值:

执行返回 0。

#### 备注:

无。

#### 样例:

```
[root@sylixos_station:/root]# ipv6 address en1 80::d473:8fd4:249c:ba%14
[root@sylixos_station:/root]# ifconfig en1
en1      enable: true linkup: true MTU: 1500 multicast: false
metric: 1 type: Ethernet-Cap HWaddr: 08:08:3E:26:0A:5A
DHCP: Disable(On) speed: 100(Mbps)
inet addr: 0.0.0.0 netmask: 0.0.0.0
gateway: 0.0.0.0 broadcast: 255.255.255.255
inet6 addr: FE80::A08:3EFF:FE26:A5A Scope:link <valid>
inet6 addr: 21::1:30E1:F9BC:3013:4E68 Scope:unknown <valid>
inet6 addr: 80::D473:8FD4:249C:BA Scope:unknown <valid>
RX ucast packets:703 nucast packets:0 dropped:0
TX ucast packets:92 nucast packets:0 dropped:0
RX bytes:69102 TX bytes:14600

[root@sylixos_station:/root]#

[root@sylixos_station:/root]# ipv6 noaddress en1 80::d473:8fd4:249c:ba%14
[root@sylixos_station:/root]#
```

#### 配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_NET\_EN> 0), 允许提供网络功能时, 该命令将会被包含。

#### 函数接口:

ipv6 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellpv6(INT iArgC, PCHAR *ppcArgV);
```

### 5.2.11. nat – 启动、关闭或设置 NAT 虚拟网络地址服务

格式:

```
nat stop
nat LAN netif WAN netif
```

说明:

该命令有 2 种用法，后面跟参数 stop 时，关闭 NAT 虚拟网络地址服务；后面跟参数 LAN netif WAN netif 用来启动 NAT 虚拟网络地址服务。

返回值:

执行成功返回 0，失败返回非 0 值。

备注:

无。

样例:

```
[root@sylixos_station:/root]# nat stop
NAT network stoped.
[root@sylixos_station:/root]# nat lo0 en1
NAT network started, [LAN: lo0] [WAN: en1]
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_NET\_EN> 0 且 LW\_CFG\_NET\_NAT\_EN>0，允许提供网络功能并且使用了 nat 服务，该命令将会被包含。

函数接口:

nat 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellNat (INT iArgC, PCHAR ppcArgV[]);
```

### 5.2.12. natalias – 添加或删除 NAT 别名

格式:

```
natalias add alias LAN start LAN end
natalias del alias
```

说明:

该命令有两种用法。跟参数 add 时用来给 LAN start 到 LAN end 网段添加别名；跟参数 del 用来删除一个 NAT 别名。

返回值:

执行成功返回 0，失败返回非 0 值。

备注:

无。

样例:

```
[root@sylixos_station:/root]# natalias add 192.168.7.2 192.168.7.2 192.168.7.9
[root@sylixos_station:/root]# natalias del 192.168.7.2
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_NET\_EN> 0 且 LW\_CFG\_NET\_NAT\_EN>0，允许提供网络功能并且使用了 nat 服务，该命令将会被包含。

函数接口:



natalias 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellNatAlias (INT iArgC, PCHAR ppcArgV[]);
```

### 5.2.13. natmap – 添加或删除 NAT 映射

格式：

```
natmap add WAN port LAN port LAN IP protocol
```

```
natmap del WAN port LAN port LAN IP protocol
```

说明：

该命令有两中用法，跟参数 add 用来添加一个 NAT 映射；跟参数 del 用来删除一个 NAT 映射。协议有 tcp 和 udp 两种。

返回值：

执行成功返回 0，失败返回非 0 值。

备注：

无。

样例：

```
[root@sylixos_station:/root]# natmap add 80 80 192.168.7.65 tcp
[root@sylixos_station:/root]# nats
NAT networking alias setting >>

  ALIAS      LOCAL START  LOCAL END
-----
NAT networking direct map setting >>

  ASS PORT  LOCAL PORT  LOCAL IP  PROTO
-----
      80      80 192.168.7.65  TCP

NAT networking summary >>
  LAN: lo0 WAN: en1
  Total Ass-node: 2048
  Used Ass-node: 0
[root@sylixos_station:/root]# natmap add 80 80 192.168.7.65 tcp
[root@sylixos_station:/root]#
```

配置：

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_NET\_EN> 0 且 LW\_CFG\_NET\_NAT\_EN>0，允许提供网络功能并且使用了 nat 服务，该命令将会被包含。

函数接口：

natmap 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellNatMap (INT iArgC, PCHAR ppcArgV[]);
```

### 5.2.14. nats – 查看当前 NAT 虚拟地址服务状态

格式：

```
nats
```

说明：

该命令查看当前 NAT 虚拟地址服务的状态。

返回值：

执行成功返回 0，失败返回-1。

备注：

nats 的状态在/proc/net/nat/info 文件中。

样例：

```
[root@sylixos_station:/root]# nats
NAT networking alias setting >>

      ALIAS          LOCAL START      LOCAL END
-----
NAT networking direct map setting >>

      ASS PORT  LOCAL PORT      LOCAL IP      PROTO
-----
NAT networking summary >>
LAN: lo0 WAN: en1
Total Ass-node: 2048
Used Ass-node: 0
[root@sylixos_station:/root]#
```

配置：

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_NET\_EN> 0 且 LW\_CFG\_NET\_NAT\_EN>0，允许提供网络功能并且使用了 nat 服务，该命令将会被包含。

函数接口：

nats 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellNatShow (INT iArgC, PCHAR ppcArgV[]);
```

## 5.2.15. nbname – 显示或设置本机的 NetBIOS 的名字

格式：

```
nbname
nbname hostname
```

说明：

该命令有 2 种用法，不跟参数用于显示本机的 NetBIOS 的名字；跟参数设置 NetBIOIS 的名字。

返回值：

执行成功返回 0，失败返回-1。

备注：

无。

样例：

```
[root@sylixos_station:/root]# nbname
local host NetBIOS name is: SYLIXOS
[root@sylixos_station:/root]# nbname user
[root@sylixos_station:/root]# nbname
local host NetBIOS name is: USER
[root@sylixos_station:/root]#
```

配置：

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_NET\_EN> 0 且 LW\_CFG\_NET\_NETBIOS\_EN>0，允许提供网络功能并且使能简易 netbios 名字服务，该命令

将会被包含。

函数接口：

nbname 命令是通过 c 语言函数实现的，函数原型是：

```
static int __inetBiosNameSet (int iArgC, char *pcArgV[]);
```

## 5.2.16. netstat – 查看网络状态

格式：

```
netstat
netstat -wtux -A -i
netstat -hrigsapl
```

说明：

该命令有 3 种用法，无参数是显示全部的套接字信息；带参数时显示相应的信息。

-h, --help	显示帮助信息
-r, --route	显示路由表信息
-i, --interface	显示网络接口信息
-g, --groups	显示组播表情况
-s, --statistics	显示统计信息
-a, --all	显示全部的套接字信息
-p, --packet	显示数据包套接字信息
-l, --listening	显示接收到的服务器套接字信息
-w, --raw	显示原始套接字信息
-t, --tcp	显示 tcp 套接字信息
-u, --udp	显示 udp 套接字信息
-x, --unix	显示 unix 套接字信息
-A <net type>	列出该网络类型连线中的相关地址。有 inet、 inet6、 unix
数字 1、2、3	网络类型可以用数字表示，1 表示 unix、2 表示 inet、3 表示 inet6

返回值：

执行成功返回 0，失败返回-1。

备注：

无。

样例：

```
[root@sylixos_station:/root]# netstat
--UNIX--:
TYPE          FLAG STATUS  SHUTD      NREAD MAX_BUFFER PATH
--PACKET--:
TYPE          FLAG PROTOCOL INDEX MMAP MMAP_SIZE TOTAL      DROP
--TCP LISTEN--:
LOCAL          REMOTE          STATUS  RETRANS RCV_WND SND_WND
*:21           *:              listen   0        0        0
*:23           *:              listen   0        0        0
--UDP--:
LOCAL          REMOTE          UDPLITE
*:69           *:0             no
*:137          *:0             no
*:161          *:0             no
```

#### 配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_NET\_EN> 0, 允许提供网络功能时, 该命令将会被包含。

#### 函数接口:

netstat 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellNetstat (INT iArgC, PCHAR *ppcArgV);
```

### 5.2.17. npfattach - 在指定网络接口上使能网络数据包过滤器

#### 格式:

```
npfattach etifname
```

#### 说明:

该命令将指定的网络接口上的网络数据包过滤器使能。

#### 返回值:

执行成功返回 0, 失败返回非 0 值。

#### 备注:

无。

#### 样例:

```
[root@sylixos_station:/root]# npfattach en1
attached.
[root@sylixos_station:/root]# npfs
NETIF ATTACH SEQNUM RULE ALLOW MAC          IPi          IPe          PORTs  PORTe
en1   YES      0 MAC  NO   12:12:25:12:45:65 N/A          N/A          N/A    N/A
en1   YES      1 IP   NO   N/A          192.168.7.65 192.168.7.96 N/A    N/A
drop:0 allow:1
```

#### 配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_NET\_EN> 0 且 LW\_CFG\_NET\_NPF\_EN>0, 允许提供网络功能并且使能 NPF 服务, 该命令将会被包含。

#### 函数接口:

npfattach 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellNetNpfAttach (INT iArgC, PCHAR *ppcArgV);
```

### 5.2.18. npfdetach - 在指定网络接口上禁能网络数据包过滤器

格式:

```
npfdetach netifname
```

说明:

该命令有 1 种用法，将指定网络接口上的网络数据包过滤器禁用。

返回值:

执行成功返回 0，失败返回非 0 值。

备注:

无。

样例:

```
[root@sylixos_station:/root]# npfdetach en1
detached.
[root@sylixos_station:/root]# npfs
NETIF ATTACH SEQNUM RULE ALLOW MAC          IPs          IPe          PORTs  PORTe
en1    NO          0 MAC    NO    12:12:25:12:45:65 N/A          N/A          N/A    N/A
en1    NO          1 IP     NO    N/A          192.168.7.65 192.168.7.96 N/A    N/A
drop:0  allow:1
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_NET\_EN> 0 且 LW\_CFG\_NET\_NPF\_EN >0，允许提供网络功能并且使能 NPF 服务，该命令将会被包含。

函数接口:

npfdetach 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellNetNpfDetach (INT iArgC, PCHAR *ppcArgV);
```

### 5.2.19. npfruleadd – 添加一条网络数据包滤波器规则

格式:

```
npfruleadd netifname mac  ??:??:??:??:??:??
npfruleadd netifname ip   ????.????.????.???  ????.????.????.???
npfruleadd netifname udp  ????.????.????.???  ????.????.????.???  iports  iporte
npfruleadd netifname tcp  ????.????.????.???  ????.????.????.???  iports  iporte
```

说明:

该命令用来添加一条网络数据包滤波器规则。其中规则类型可以有 mac、ip、udp、tcp。添加 mac 规则时参数是禁止通行的 mac 地址数组；添加 ip 规则时，参数是禁止通行的 ip 起始与结束地址；当添加 udp 或 tcp 规则时，参数是禁止通行的 ip 起始与结束地址、禁止通行的本地起始与结束端口号。

返回值:

执行成功返回 0，失败返回非 0 值。

备注:

无。

样例:

```
[root@sylixos_station:/root]# npfruleadd en1 mac 12:12:25:12:45:65
rule add ok
[root@sylixos_station:/root]# npfruleadd en1 ip 192.168.7.65 192.168.7.96
rule add ok
[root@sylixos_station:/root]# npfs
NETIF ATTACH SEQNUM RULE ALLOW MAC          IPs          IPe          PORTs  PORTe
en1    NO          0 MAC    NO    12:12:25:12:45:65 N/A          N/A          N/A    N/A
en1    NO          1 IP     NO    N/A          192.168.7.65 192.168.7.96 N/A    N/A

drop:0  allow:1
[root@sylixos_station:/root]#
```

#### 配置：

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_NET\_EN> 0 且 LW\_CFG\_NET\_NPF\_EN >0，允许提供网络功能并且使能 NPF 服务，该命令将会被包含。

#### 函数接口：

nbname 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellNetNpfRuleAdd (INT iArgC, PCHAR *ppcArgV);
```

### 5.2.20. npfruledel – 删除一条网络数据包过滤器规则

#### 格式：

```
npfruledel netifname rule sequence num
```

#### 说明：

该命令用来删除一条网络数据包过滤器规则。

#### 返回值：

执行成功返回 0，失败返回非 0 值。

#### 备注：

无。

#### 样例：

```
[root@sylixos_station:/root]# npfruledel en1 0
delete.
[root@sylixos_station:/root]# npfs
NETIF ATTACH SEQNUM RULE ALLOW MAC          IPs          IPe          PORTs  PORTe
en1    NO          0 IP     NO    N/A          192.168.7.65 192.168.7.96 N/A    N/A

drop:0  allow:1
[root@sylixos_station:/root]#
```

#### 配置：

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_NET\_EN> 0 且 LW\_CFG\_NET\_NPF\_EN >0，允许提供网络功能并且使能 NPF 服务，该命令将会被包含。

#### 函数接口：

npfruledel 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellNetNpfRuleDel (INT iArgC, PCHAR *ppcArgV);
```

### 5.2.21. npfs – 查看网络数据包过滤器状态

#### 格式：

```
nbfs
```

#### 说明：

该命令用来查看网络数据包过滤器的状态。

返回值:

执行成功返回 0，失败返回-1。

备注:

网络数据包过滤器的状态保存在/proc/net/netfilter 下。

样例:

```
[root@sylixos_station:/root]# npfs
NETIF ATTACH SEQNUM RULE ALLOW MAC          IPs          IPe          PORTs  PORTe
p57    NO          0 IP    NO    N/A          192.168.7.2   192.168.7.29 N/A     N/A
ma208  NO          0 MAC   NO    32:56:21:23:23:32 N/A          N/A          N/A     N/A
drop:0  allow:0
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_NET\_EN> 0 且 LW\_CFG\_NET\_NPF\_EN>0，允许提供网络功能并且使能 NPF 服务，该命令将会被包含。

函数接口:

nbfs 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellNetNpfShow (INT iArgC, PCHAR *ppcArgV);
```

## 5.2.22. ping - Ping 命令

格式:

```
ping ip/hostname [-l datalen] [-n times] [-i ttl] [-w timeout]
```

说明:

向指定的 ip 发送数据，测试网络的通断。使用-l 参数指明发送数据包的大小；使用-n 参数指明发送数据包的大小；-i 参数设置 ttl 的值，最大为 255；-w 指定超时时间。其中默认的数据包大小为 32byte，发送次数为 4，TTL 为 255，超时时间为 3000。

返回值:

执行成功返回 0，失败返回非 0 值。

备注:

无。

样例:

```
[root@sylixos_station:/root]# ping 192.168.7.30
Pinging 192.168.7.30

Reply from 192.168.7.30: bytes=32 time=0ms TTL=255
Reply from 192.168.7.30: bytes=32 time=0ms TTL=255
Reply from 192.168.7.30: bytes=32 time=0ms TTL=255
Reply from 192.168.7.30: bytes=32 time=0ms TTL=255

Ping statistics for 192.168.7.30:
    Packets: Send = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_NET\_EN> 0 且 LW\_CFG\_NET\_PING\_EN>0，允许提供网络功能并且需要 ping 工具时，该命令将会被包含。

函数接口：

ping 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellPing (INT iArgC, PCHAR *ppcArgV);
```

### 5.2.23. ping6 – IPv6 Ping 命令

格式：

```
ping6 ip(v6)/hostname [-l datalen] [-n times] [-w timeout] [-I interface]
```

说明：

向指定的 ip (ipv6) 发送数据，测试网络的通断。使用 -l 参数指明发送数据包的大小；使用 -n 参数指明发送数据包的大小；-I 设置网络接口名；-w 指定超时时间。其中默认的数据包大小为 32byte，发送次数为 4，超时时间为 3000。

返回值：

执行成功返回 0，失败返回-1。

备注：

无。

样例：

```
[root@sylixos_station:/root]# ping6 FE80::A08:3EFF:FE26:A5A
Pinging FE80::A08:3EFF:FE26:A5A

Reply from FE80::A08:3EFF:FE26:A5A: bytes=32 time=0ms hoplim=255
Reply from FE80::A08:3EFF:FE26:A5A: bytes=32 time=0ms hoplim=255
Reply from FE80::A08:3EFF:FE26:A5A: bytes=32 time=0ms hoplim=255
Reply from FE80::A08:3EFF:FE26:A5A: bytes=32 time=0ms hoplim=255

Ping statistics for FE80::A08:3EFF:FE26:A5A:
    Packets: Send = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

[root@sylixos_station:/root]#
```

配置：

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_NET\_EN> 0 且 LW\_CFG\_NET\_PING\_EN>0，允许提供网络功能并且需要 ping 工具时，该命令将会被包含。

函数接口：

ping6 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellPing6 (INT iArgC, PCHAR *ppcArgV);
```

### 5.2.24. route – 添加、删除、修改或查看系统路由表

格式：

- 1、route
- 2、route add|change -host | -net ipaddr gateway if/dev name



3、route add/change default if|dev name

4、route del ipaddr

#### 说明:

该命令只有 4 种用法，格式 1 无参数用于显示路由表。格式 2 用于修改或增加一个路由表信息；格式 3 用于修改或增加默认的路由信息；格式 4 用于删除一条路由信息。

选项:

add	用来增加一个路由信息
change	改变现有的路由信息
del	删除现有的路由信息
-host	目标地址是一个主机
-net	目标地址是一个网络
if	为可以访问的目标接口指定接口索引
dev	指定目标接口索引为输入的 name
default	操作默认的路由

参数:

ipaddr	ip 地址
gateway	掩码
name	接口索引的名字

查看路由表输出信息:

**Destination:** 网络目的地址，列出了路由器连接的所有的网段。

**Gateway:** 网关，一旦路由器确定它要把这个数据包转发到哪一个目的网络，路由器就要查看网关列表。网关表告诉路由器这个数据包应该转发到哪一个 IP 地址才能达到目的网络。

**Mask:** 网络掩码，提供这个网段本身的子网掩码，而不是连接到这个网段的网卡的子网掩码。这基本上能够让路由器确定目的网络的地址类。

**Flag:** 路由标志，标记当前网络节点的状态。

U Up 表示此路由当前为启动状态

H Host, 表示此网关为一主机

G Gateway, 表示此网关为一路由器

R Reinstate Route, 使用动态路由重新初始化的路由

D Dynamically,此路由是动态性地写入

M Modified, 此路由是由路由守护程序或导向器动态修改

! 表示此路由当前为关闭状态

**Interface:** 接口索引，告诉路由器哪一个网卡连接到了合适的目的网络。

#### 返回值:

执行成功返回 0，失败返回非 0 值。

#### 备注:

无。

#### 样例:

```
[root@sylixos_station:/root]# route add -host 192.168.7.98 255.255.255.0 dev en1
route 192.168.7.98 add successful.
[root@sylixos_station:/root]# route
kernel routing tables
Destination      Gateway          Mask             Flag             Interface
192.168.7.98      *                255.255.255.0    UH               en1

build-in routing tables
Destination      Gateway          Mask             Flag             Interface
192.168.7.0      *                255.255.255.0    U               en1
192.168.7.30     *                255.255.255.0    UH              en1
127.0.0.0        *                255.0.0.0        U               lo0
127.0.0.1        *                255.0.0.0        UH              lo0
default          192.168.7.1     255.255.255.0    UG               en1
[root@sylixos_station:/root]#
```

```
[root@sylixos_station:/root]# route del 192.168.7.98
route 192.168.7.98 delete successful.
```

#### 配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_NET\_EN> 0, 允许提供网络功能时, 该命令将会被包含。

#### 函数接口:

route 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellRoute (INT iArgC, PCHAR ppcArgV[]);
```

### 5.2.25. tftp - 使用 tftp 命令接收或者发送一个文件

#### 格式:

```
tftp -i Host get | put Source Destination
```

#### 说明:

该命令用来发送或接受一个文件。

#### 返回值:

执行成功返回 0, 失败返回-1。

#### 备注:

无。

#### 样例:

```
[root@sylixos_station:/root]# tftp -i localhost put /apps/helloWord/helloWord hello
sending file...
file transfer completed.
[root@sylixos_station:/root]# ls /tmp/
hello
[root@sylixos_station:/root]#
```

#### 配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_NET\_EN> 0 且 LW\_CFG\_NET\_TFTP\_EN >0, 允许提供网络功能并且使能了 tftp 服务, 该命令将会被包含。

#### 函数接口:

tftp 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellTftp (INT iArgC, PCHAR *ppcArgV);
```

## 5.2.26. tftpdpath - 查看或设置 tftp 服务器本地路径

格式:

```
tftpdpath
tftpdpath    newpath
```

说明:

该命令有 2 种用法，不跟参数用于查看当前的 tftp 服务器的本地路径；加 newpath 设置 tftp 服务器的本地路径。

返回值:

执行成功返回 0，失败返回-1。

备注:

无。

样例:

```
[root@sylixos_station:/root]# tftpdpath
tftpd path: /tmp
[root@sylixos_station:/root]# tftpdpath /
[root@sylixos_station:/root]# tftpdpath
tftpd path: /
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_NET\_EN> 0 且 LW\_CFG\_NET\_NETBIOS\_EN>0，允许提供网络功能并且使能简易 netbios 名字服务，该命令将会被包含。

函数接口:

tftpdpath 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellNetTftpdPath (INT iArgC, PCHAR *ppcArgV);
```

## 5.2.27. vlan - 显示、设置和删除 net 接口的 VLAN ID

格式:

```
vlan
vlan set netifanme
vlan clear netifaname
```

说明:

该命令有 3 种用法，无参数时显示 net 接口；set 选项设置 VLAN ID；clear 清除响应接口的 VLAN ID。

返回值:

执行成功返回 0，失败返回-1。

备注:

无。

样例:

```
[root@sylixos_station:/etc]# vlan set en1 1
[root@sylixos_station:/etc]# vlan

INDEX  VLAN ID
-----
      1      1
[root@sylixos_station:/etc]# vlan clear en1
[root@sylixos_station:/etc]# vlan

INDEX  VLAN ID
-----
[root@sylixos_station:/etc]#
```

#### 配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_NET\_EN> 0 且 LW\_CFG\_NET\_VLAN\_EN >0, 允许提供网络功能并且使能 VLAN 工具, 该命令将会被包含。

#### 函数接口:

vlan 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellVlan (INT iArgC, PCHAR *ppcArgV);
```

### 5.2.28. vnpnclose - 删除一个虚拟网络接口\*\*

#### 格式:

```
vnpnclose netifname
```

#### 说明:

该命令用于删除一个指定的虚拟网络接口。

#### 返回值:

执行成功返回 0, 失败返回非 0 值。

#### 备注:

无。

#### 样例:

无。

#### 配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_NET\_EN> 0 且 LW\_CFG\_NET\_VPN\_EN >0, 允许提供网络功能并且使能 VNP 服务, 该命令将会被包含。

#### 函数接口:

vnpnclose 命令是通过 c 语言函数实现的, 函数原型是:

```
INT __tshellVpnClose (INT iArgC, PCHAR *ppcArgV)
```

### 5.2.29. vnpnopen - 创建一个虚拟网络接口 \*\*

#### 格式:

```
vnpnopen configuration file
```

#### 说明:

该命令用来创建一个虚拟的网络接口。

#### 返回值:

执行成功返回 0, 失败返回非 0 值。

#### 备注:

配置文件应含义的信息有:

- ca 证书文件名 (.pem or .crt)、
- 私有证书文件名 (.pem or .crt)、
- 私有密钥文件 (.pem or .crt)、
- 私有密钥文件解压密码、
- 服务器 ip、
- VPN 虚拟网卡地址、
- VNP 虚拟网卡掩码、
- VPN 虚拟网卡网关、
- VNP 服务器端口、
- SSL 通信超时时间、
- SSL 认证选项、
- 6 个字节的虚拟网卡 MAC 地址。

#### 样例:

无。

#### 配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_NET\_EN> 0 且 LW\_CFG\_NET\_VPN\_EN>0, 允许提供网络功能并且使能 VNP 服务, 该命令将会被包含。

#### 函数接口:

vpnopen 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellVpnOpen (INT iArgC, PCHAR *ppcArgV);
```

## 6. 时间

### 6.1. 介绍

- date – 显示或设置系统当前时间
- hwclock – 显示或同步操作系统与硬件 RTC 时钟
- times – 显示 utc 或 local 时间
- tzsync – 与环境变量 TZ 的时区同步

### 6.2. 命令使用

#### 6.2.1. date – 显示或设置系统当前时间

##### 格式:

date

date -s time | date

##### 说明:

该命令有 2 种用法, 无参数是显示当前系统时间; 参数-s 用于设置时间或日期, 但每次只能设置其中的一个。

##### 返回值:

执行成功返回 0, 失败返回-1。

##### 备注:

无。

样例：

```
[root@sylixos_station:/root]# date
Mon Nov 28 09:11:36 2016
[root@sylixos_station:/root]# date -s 20161129
Tue Nov 29 09:11:41 2016
[root@sylixos_station:/root]# date -s 10:12:00
Tue Nov 29 10:12:00 2016
[root@sylixos_station:/root]#
```

配置：

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

函数接口：

date 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdDate (INT iArgC, PCHAR *ppcArgV);
```

## 6.2.2. hwclock – 显示或同步操作系统与硬件 RTC 时钟

格式：

```
hwclock --show
hwclock --hctosys
hwclock --systohc
```

说明：

该命令有 3 种用法，跟参数—show 显示硬件 RTC 时钟；跟参数—hctosys 同步 RTC 时钟到操作系统时钟；跟参数—systohc 同步操作系统的时钟到 RTC 时钟。

返回值：

执行成功返回 0，失败返回非 0 值。

备注：

无。

样例：

```
[root@sylixos_station:/root]# hwclock --show
Tue Nov 22 15:54:59 2016
[root@sylixos_station:/root]# hwclock --hctosys
[root@sylixos_station:/root]#
```

配置：

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_RTC\_EN > 0 从 0.9.7 版后开始支持此命令，该命令将会被包含。

函数接口：

hwclock 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdHwclock (INT iArgC, PCHAR *ppcArgV);
```

### 6.2.3. times – 显示 utc 或 local 时间

格式:

```
times
times -utc
```

说明:

该命令有 2 种用法，不跟参数显示本地时间，跟参数-utc 显示格林威治时间。

返回值:

执行返回 0。

备注:

无。

样例:

```
[root@sylixos_station:/root]# times
Tue Nov 22 15:56:31 2016
[root@sylixos_station:/root]# times -utc
UTC : Tue Nov 22 07:56:37 2016
[root@sylixos_station:/root]#
```

配置:

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

函数接口:

times 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellSysCmdTimes (INT iArgC, PCHAR *ppcArgV);
```

### 6.2.4. tzsync – 与环境变量 TZ 的时区同步

格式:

```
tzsync
```

说明:

该命令将时间与环境变量 TZ 的时区同步。

返回值:

执行返回 0。

备注:

如果环境变量 TZ 的时区改变，则相应的时间也随之改变。

样例:

改变环境变量 TZ

```
[root@sylixos_station:/etc]# vi temp
TZ=CST-8:00:00
```

装载环境变量，同步前后，查看时间

```
[root@sylixos_station:/etc]# varload temp
environment variables load from temp success.
[root@sylixos_station:/etc]# times
Tue Nov 22 17:09:02 2016
[root@sylixos_station:/etc]# tzsync
[root@sylixos_station:/etc]# times
Tue Nov 22 16:09:11 2016
[root@sylixos_station:/etc]#
```

#### 配置:

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

#### 函数接口:

tzsync 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellSysCmdTzsync (INT iArgC, PCHAR *ppcArgV);
```

## 7. 动态装载

### 7.1. 介绍

- debug – 调试一个进程
- dlconfig - 配置动态链接器工作参数
- leakchk – 内存泄漏检查
- leakchkstart – 启动内存泄漏跟踪器
- leakchkstop – 关闭内存泄漏跟踪器
- lsmod - 查看系统装载的所有内核模块信息
- modulegcov – 生成内核模块代码文件 (\*.gcda)
- modulereg – 注册一个模块
- modules - 查看系统装载的所有内核模块与进程动态链接库信息
- modulestat - 查看一个内核模块或动态链接库文件信息
- moduleunreg – 卸载一个模块

### 7.2. 命令使用

#### 7.2.1. debug – 调试一个进程

##### 格式:

```
debug connect options program arguments
debug --attach connect options program arguments
```

##### 说明:

该命令用于调试一个进程。后面的参数依次是连接方式、工程名、参数列表。其中连接方式可以有网络、串口、终端、attach

##### 返回值:

执行成功返回 0，失败返回-1。

##### 备注:

无。



样例:

使用网络调试

```
[root@sylixos_station:/apps/test1]# debug localhost:1234 ./test1
[GDB]Waiting for connect...
```

使用串口调试

```
[root@sylixos_station:/apps/test1]# debug /dev/ttyS1 ./test1
[GDB]Serial device: /dev/ttyS1 115200,n,8,1
```

使用终端调试

```
[root@sylixos_station:/apps/test1]# debug terminal ./test1
[GDB]Serial device: terminal 115200,n,8,1
```

使用 attrch 方式调试

```
[root@sylixos_station:/apps/test1]# debug --attach localhost:1234 ./test1
```

配置:

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

函数接口:

debug 命令是通过 c 语言函数实现的，函数原型是:

```
static INT gdbMain (INT argc, CHAR **argv);
```

## 7.2.2. dlconfig - 配置动态链接器工作参数

格式:

```
dlconfig share en | dis
dlconfig refresh *
dlconfig refresh
```

说明:

该命令有 2 种用法，跟选项 share，使能或禁能动态链接器；跟选项 refresh 清除分享数据信息，其中跟\*表示只清除系统的，否则清除全部的。

返回值:

执行返回 0。

备注:

无。

样例:

```
[root@sylixos_station:/apps/test1]# dlconfig refresh *
[root@sylixos_station:/apps/test1]# dlconfig refresh
[root@sylixos_station:/apps/test1]# dlconfig share dis
[root@sylixos_station:/apps/test1]# dlconfig share en
[root@sylixos_station:/apps/test1]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_MODULELOADER\_EN>0，需要提供模块装载服务，该命令将会被包含。

函数接口:

dlconfig 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellDIConfig (INT iArgC, PCHAR *ppcArgV);
```

### 7.2.3. leakchk – 内存泄漏检查

格式:

```
leakchk
```

说明:

该命令用来打印内存泄漏追踪信息。

返回值:

执行成功返回 0，失败返回-1。

备注:

无。

样例:

```
[root@sylixos_station:/lib/modules]# leakchk
-----
HEAP      THREAD      TIME      ADDR      SIZE      PURPOSE
-----
kersys    t_tshell    Mon Nov 28 09:58:50 2016 30e13f88    32 __tshellHistorySave
total unfree segment: 1 size: 32
[root@sylixos_station:/lib/modules]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_SHELL\_HEAP\_TRACE\_EN>0，使能 shell heap 跟踪工具，该命令将会被包含。

函数接口:

leakchk 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellHeapCmdLeakChk (INT iArgC, PCHAR ppcArgV[]);
```

### 7.2.4. leakchkstart – 启动内存泄漏跟踪器

格式:

```
leakchkstart max save node number pid
```

说明:

该命令用来启动内存泄漏跟踪器，后面的参数是最大跟踪节点数、进程 id。

返回值:

执行成功返回 0，失败返回非 0 值。

备注:

最大跟踪节点数最小为 1024。

进程 id 默认为 0。

样例:

```
[root@sylixos_station:/lib/modules]# leakchkstart 1024 0
leakcheck start checking...
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_SHELL\_HEAP\_TRACE\_EN>0，使能 shell heap 跟踪工具，该命令将会被包含。

函数接口:

leakchkstart 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellHeapCmdLeakChkStart (INT iArgC, PCHAR *ppcArgV)
```

## 7.2.5. leakchkstop – 关闭内存泄漏跟踪器

格式:

leakchkstop

说明:

该命令用来关闭内存泄漏跟踪器。

返回值:

执行成功返回 0，失败返回非 0 值。

备注:

无。

样例:

```
[root@sylixos_station:/root]# leakchkstop
-----
HEAP      THREAD      TIME      ADDR      SIZE      PURPOSE
-----
kersys    t_tshell    Mon Nov 28 10:03:30 2016 30e22e50    32    __tshellHistorySave
kersys    t_tshell    Mon Nov 28 10:02:58 2016 30bc8600    48    API_TShellOptInd
kersys    t_tshell    Mon Nov 28 10:02:58 2016 30bc85c8    32    __tshellHistorySave
kersys    t_tshell    Mon Nov 28 10:02:07 2016 30bc85a0    16    __tshellReadlineInit
kersys    t_tshell    Mon Nov 28 10:02:07 2016 30bc8578    16    API_ThreadCleanupPush
kersys    t_tshell    Mon Nov 28 10:02:07 2016 30bc8350    528    __tshellShowPrompt
kersys    t_tshell    Mon Nov 28 10:02:07 2016 30e22c18    544    __rngCreate
kersys    t_tshell    Mon Nov 28 10:02:07 2016 30e229e0    544    __rngCreate
kersys    t_tshell    Mon Nov 28 10:02:07 2016 30e227b8    528    lib_malloc
kersys    t_tshell    Mon Nov 28 10:02:07 2016 30e22590    528    __IosEnvCreate
kersys    t_tshell    Mon Nov 28 10:02:07 2016 30e22568    16    API_ThreadCleanupPush
kersys    t_except    Mon Nov 28 10:02:07 2016 30e22230    800    __selTaskCreateHook
-----
total unfree segment: 12 size: 3632
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_SHELL\_HEAP\_TRACE\_EN>0，使能 shell heap 跟踪工具，该命令将会被包含。

函数接口:

leakchkstop 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellHeapCmdLeakChkStop (INT iArgC, PCHAR *ppcArgV)
```

## 7.2.6. lsmod - 查看系统装载的所有内核模块信息

格式:

lsmod

说明:

该命令有 1 种用法，用来显示系统装置的所有内核模块的信息。

返回值:

执行返回 0。

备注:

无。

样例:

```
[root@sylixos_station:/root]# lsmod

NAME                                HANDLE  TYPE  GLB  BASE  SIZE  SYMCNT
-----
VPROCESS: kernel                    pid:    0  TOTAL MEMORY: 0

total modules: 0
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_MODULELOADER\_EN>0，需要提供模块装载服务，该命令将会被包含。

函数接口:

lsmod 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellLsmod (INT iArgC, PCHAR *ppcArgV);
```

## 7.2.7. modulegcov – 生成内核模块代码文件 (\*.gcda)

格式:

```
modulegcov kernel module handle
```

说明:

该命令用来生成内核模块代码文件。

返回值:

执行成功返回 0，失败返回非 0 值。

备注:

无。

样例:

```
[root@sylixos_station:/root]# modules

NAME                                HANDLE  TYPE  GLB  BASE  SIZE  SYMCNT
-----
VPROCESS: kernel                    pid:    0  TOTAL MEMORY: 8192
+ file.ko                          30e14be8  KERNEL YES 60005000      940      1

total modules: 1
[root@sylixos_station:/root]# modulegcov 30e14be8
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_MODULELOADER\_EN>0 且 LW\_CFG\_MODULELOADER\_GCOV\_EN>0，需要提供模块装载服务并且允许内核模块代码覆盖率分析接口，该命令将会被包含。

函数接口:

modulegcov 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellModuleGcov (INT iArgC, PCHAR *ppcArgV);
```

## 7.2.8. modulereg – 注册一个模块

格式:

```
modulereg kernel module file *.ko
```

说明:

该命令有 1 种用法，用来注册一个模块文件。

返回值：

执行成功返回 0，失败返回非 0 值。

备注：

无。

样例：

```
[root@sylixos_station:/root]# modulereg /lib/modules/file.ko
module /lib/modules/file.ko register ok, handle: 0x30e15370
[root@sylixos_station:/root]#
```

配置：

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_MODULELOADER\_EN>0，需要提供模块装载服务，该命令将会被包含。

函数接口：

modulereg 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellModuleReg (INT iArgC, PCHAR *ppcArgV);
```

## 7.2.9. modules - 查看系统装载的所有内核模块与进程动态链接库信息

息

格式：

modules

modules module name

说明：

该命令有 1 种用法，无参数是显示所有的内核模块与进程动态链接库信息；带参数查看指定的内核模块与进程动态链接库信息。

返回值：

执行返回 0。

备注：

无。

样例：

```
[root@sylixos_station:/root]# modules

-----
NAME                HANDLE    TYPE  GLB   BASE      SIZE    SYMCNT
-----
VPROCESS: kernel    pid:    0 TOTAL MEMORY: 16384
+ interruptK.ko     30e13f98 KERNEL YES 60006000    28c      2
+ file.ko           30e14be8 KERNEL YES 60005000    940      1
total modules: 2
[root@sylixos_station:/root]#
```

配置：

该命令属于系统提供的 tshell 命令，当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令，则包含该命令。

函数接口：

modules 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellModuleShow (INT iArgC, PCHAR *ppcArgV);
```

## 7.2.10. modulestat - 查看一个内核模块或动态链接库文件信息

格式:

modulestat program file

说明:

该命令用来查看一个内核模块或动态链接库文件的信息。

返回值:

执行成功返回 0，失败返回非 0 值。

备注:

无

样例:

```
[root@sylixos_station:/root]# modulestat /lib/modules/file.ko
File Type: ELF
Machine:   ARM family
Type:      ET_REL
Entry:     0
Section Headers:
TYPE      ADDRESS      OFFSET    SIZE      FLAGS
NULL      00000000             0          0
PROGBITS  00000000             34         7c4    [ALLOC] [EXEC]
REL       00000000             dc4        178
PROGBITS  00000000             7f8        78    [ALLOC]
PROGBITS  00000000             870         a    [ALLOC] [WRITE]
NOBITS    00000000             87c        94    [ALLOC] [WRITE]
PROGBITS  00000000             87c        79
NONE      00000000             8f5        2f
STRTAB    00000000             924        51
SYMTAB    00000000             b30       1a0
STRTAB    00000000             cd0        f4
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_MODULELOADER\_EN>0，需要提供模块装载服务，该命令将会被包含。

函数接口:

modulestat 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellModulestat (INT iArgC, PCHAR *ppcArgV);
```

## 7.2.11. moduleunreg – 卸载一个模块

格式:

moduleunreg kernel module handle

说明:

该命令有 1 种用法，用来卸载一个模块。

返回值:

执行成功返回 0，失败返回非 0 值。

备注:

无。

样例:

```
[root@sylixos_station:/root]# modules
```

NAME	HANDLE	TYPE	GLB	BASE	SIZE	SYMCNT
-----						
VPROCESS: kernel	pid: 0	TOTAL MEMORY: 16384				
+ interruptK.ko	30e13f98	KERNEL	YES	60006000	28c	2
+ file.ko	30e14be8	KERNEL	YES	60005000	940	1
-----						
total modules: 2						

```
[root@sylixos_station:/root]# moduleunreg 30e13f98
hello_module exit!
module /lib/modules/interruptK.ko unregister ok.
[root@sylixos_station:/root]#
```

配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_MODULELOADER\_EN>0，需要提供模块装载服务，该命令将会被包含。

函数接口:

moduleunreg 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellModuleUnreg (INT iArgC, PCHAR *ppcArgV);
```

## 8. 其他

### 8.1. 介绍

- args – 显示输入的全部参数，并且以空格为分隔符
- crypt – 对数据进行加密
- perfrefresh – 更新性能信息统计
- perfs – 显示性能统计的信息
- perfstart – 启动性能分析工具
- perfstop – 停止性能分析工具
- xmodemr - 使用 xmodem 协议接收一个文件
- xmodems - 使用 xmodem 协议发送一个文件

### 8.2. 命令使用

#### 8.2.1. args – 显示输入的全部参数，并且以空格为分隔符

格式:

```
args [any argument...]
```

说明:

该命令将输入的参数以一定的格式显示出来。

返回值:

执行返回 0。

备注:

无。

样例:

```
[root@sylixos_station:/root]# args
arg 1 is args
[root@sylixos_station:/root]# args SylixOS system is gool
arg 1 is args
arg 2 is SylixOS
arg 3 is system
arg 4 is is
arg 5 is gool
[root@sylixos_station:/root]#
```

#### 配置:

该命令属于系统提供的 tshell 命令。当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。

#### 函数接口:

args 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdArgs (INT iArgC, PCHAR ppcArgV[]);
```

### 8.2.2. crypt – 对数据进行加密

#### 格式:

```
crypt key salt
```

#### 说明:

该命令将输入的参数以键值对的格式显示出来。其中 key 是要加密的明文，salt 指定用来加密的密钥。

#### 返回值:

执行成功返回 0，失败返回-1。

#### 备注:

无。

#### 样例:

```
[root@sylixos_station:/etc]# crypt root root
roK20XGbWEsSM
[root@sylixos_station:/etc]# crypt root 2root
2rUSJaaFNwmv6
[root@sylixos_station:/etc]# crypt root 1root
1r9MzDzsjs4uM
[root@sylixos_station:/etc]#
```

#### 配置:

当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。当 LW\_CFG\_SHELL\_PASS\_CRYPT\_EN>0，用户密码相关的支持，该命令将会被包含。

#### 函数接口:

crypt 命令是通过 c 语言函数实现的，函数原型是：

```
static INT __tshellSysCmdCrypt (INT iArgC, PCHAR ppcArgV[]);
```

### 8.2.3. perfrefresh – 更新性能信息统计 \*\*

#### 格式:

```
perfrefresh
```

#### 说明:



该命令更新信息统计。

返回值:

返回值是 0。

备注:

无。

样例:

无。

配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_SYSPERF\_EN>0 且 LW\_CFG\_SHELL\_PERF\_TRACE\_EN >0, 允许系统性能分析并且使能了系统性能分析工具, 该命令将会被包含。

函数接口:

perfrefresh 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellPerfCmdPerfRefresh (INT iArgC, PCHAR ppcArgV[]);
```

#### 8.2.4. perfs – 显示性能统计的信息 \*\*

格式:

perfs

说明:

该命令用来显示性能统计的信息, 参数为管道缓存大小 (128~4096)、

返回值:

执行返回 0。

备注:

无。

样例:

无。

配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_SYSPERF\_EN>0 且 LW\_CFG\_SHELL\_PERF\_TRACE\_EN >0, 允许系统性能分析并且使能了系统性能分析工具, 该命令将会被包含。

函数接口:

perfs 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellPerfCmdPerfShow (INT iArgC, PCHAR ppcArgV[]);
```

#### 8.2.5. perfstart – 启动性能分析工具 \*\*

格式:

perfstart pipe buffer len performance save node refresh period

说明:

该命令用来显示性能统计的信息, 参数为管道缓存大小 (128~4096)、性能保存的节点 (10~30)、更新周期 (大于等于 1s)。

返回值:

执行成功返回 0, 失败返回-1。

#### 备注:

pipe buffer len : 默认 1024  
performance save node : 默认 20  
refresh period : 默认 10000

#### 样例:

无。

#### 配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_SYSPERF\_EN>0 且 LW\_CFG\_SHELL\_PERF\_TRACE\_EN >0, 允许系统性能分析并且使能了系统性能分析工具, 该命令将会被包含。

#### 函数接口:

perfstart 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellPerfCmdPerfStart (INT iArgC, PCHAR ppcArgV[]);
```

### 8.2.6. perfstop – 停止性能分析工具 \*\*

#### 格式:

perfstop

#### 说明:

该命令将输入的参数以键值对的格式显示出来。

#### 返回值:

执行成功返回 0, 失败返回-1。

#### 备注:

无。

#### 样例:

无。

#### 配置:

当 LW\_CFG\_SHELL\_EN>0 时, 会允许操作系统提供 tshell 命令。当 LW\_CFG\_SYSPERF\_EN>0 且 LW\_CFG\_SHELL\_PERF\_TRACE\_EN >0, 允许系统性能分析并且使能了系统性能分析工具, 该命令将会被包含。

#### 函数接口:

args 命令是通过 c 语言函数实现的, 函数原型是:

```
static INT __tshellSysCmdArgs (INT iArgC, PCHAR ppcArgV[]);
```

### 8.2.7. xmodemr - 使用 xmodem 协议接收一个文件

#### 格式:

xmodemr file path

#### 说明:

该命令使用 xmodem 协议从远程接受一个文件。

#### 返回值:

执行成功返回 0, 失败返回非 0 值。

#### 备注:

无。

样例:

```
[root@sylixos_station:/apps/hello]# xmodems hello
```

配置:

该命令属于系统提供的 tshell 命令。当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。

函数接口:

xmodemr 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellFsCmdXmodemr (INT iArgC, PCHAR ppcArgV[]);
```

## 8.2.8. xmodems - 使用 xmodem 协议发送一个文件

格式:

```
xmodems file path
```

说明:

该命令用来使用 xmodem 协议发送给远程一个文件。

返回值:

执行成功返回 0，失败返回非 0 值。

备注:

无。

样例:

```
[root@sylixos_station:/root]# xmodems /apps/helloWord/helloWord
```

配置:

该命令属于系统提供的 tshell 命令。当 LW\_CFG\_SHELL\_EN>0 时，会允许操作系统提供 tshell 命令。

函数接口:

xmodems 命令是通过 c 语言函数实现的，函数原型是:

```
static INT __tshellFsCmdXmodems (INT iArgC, PCHAR ppcArgV[]);
```