



# **Tina Linux Tinatest 测试 使用指南**

**版本号: 1.0  
发布日期: 2020.08.06**

## 版本历史

版本号	日期	制/修订人	内容描述
1.0	2020.08.06	AWA1526	初始版本



# 目 录

<b>1 概述</b>	<b>1</b>
1.1 编写目的 . . . . .	1
1.2 适用范围 . . . . .	1
1.3 相关人员 . . . . .	1
<b>2 TinaTest 简介</b>	<b>2</b>
2.1 简介 . . . . .	2
2.2 使能 TinaTest 软件包 . . . . .	2
2.3 TinaTest 目录结构 . . . . .	2
2.3.1 测试说明文档 . . . . .	2
<b>3 TinaTest 的使用</b>	<b>4</b>
3.1 配置树简述 . . . . .	4
3.2 TinaTest 命令说明 . . . . .	5
3.2.1 命令说明 . . . . .	5
3.2.2 示例 . . . . .	5
3.2.2.1 显示配置文件 . . . . .	5
3.2.2.2 执行单个测试用例 . . . . .	6
3.2.2.3 执行一类测试用例 . . . . .	7
3.2.2.4 执行多类测试用例 . . . . .	9
<b>4 TinaTest 配置说明</b>	<b>10</b>
4.1 修改配置项 . . . . .	10
4.1.1 直接修改配置文件 . . . . .	10
4.1.1.1 配置格式简述 . . . . .	10
4.1.1.2 配置项配置值类型 . . . . .	11
4.1.1.3 示例 . . . . .	11
4.1.2 通过 menuconfig 修改配置项 . . . . .	11
4.1.2.1 示例 . . . . .	12
4.2 配置项说明 . . . . .	12
4.2.1 配置项分类 . . . . .	12
4.2.2 配置项 . . . . .	13
4.2.2.1 任务相关 . . . . .	13
4.2.2.2 局部信息 . . . . .	13
4.2.2.3 局部限制 . . . . .	13
4.2.2.4 全局信息 . . . . .	14
<b>5 测试用例</b>	<b>15</b>
5.1 测试用例分类 . . . . .	15
5.1.1 一级分类 . . . . .	15
5.1.2 多级分类 . . . . .	15
5.2 添加测试用例 . . . . .	16

5.2.1 添加测试用例源码（可选） . . . . .	16
5.2.2 基于 C/C++ 的 API . . . . .	17
5.2.2.1 与 json 相关的 API . . . . .	17
5.2.2.2 交互 API . . . . .	19
5.2.2.3 其余 API . . . . .	20
5.2.3 基于 shell 的 API . . . . .	21
5.2.3.1 与 json 相关的 API . . . . .	21
5.2.3.2 交互 API . . . . .	22
5.2.4 测试用例属性文件 (private.conf) 及快速注册/注销测试用例 . . . . .	23
5.2.4.1 add_testcase.sh 使用说明 . . . . .	24
5.2.4.2 del testcase.sh 使用说明 . . . . .	25
5.2.5 测试用例源码自编译 . . . . .	26
5.2.5.1 源码中有 Makefile . . . . .	26
5.2.5.2 源码中无 Makefile . . . . .	26



# 1 概述

## 1.1 编写目的

本文档主要介绍全志科技 Tina Linux SDK 的测试平台 TinaTest，包括 TinaTest 的使用，TinaTest 的差异性配置，TinaTest 测试用例分类，TinaTest 如何添加测试用例等。

## 1.2 适用范围

Tina Linux SDK V2.5 之后的版本。

## 1.3 相关人员

Tina Linux 的开发、维护、测试人员。

## 2 TinaTest 简介

### 2.1 简介

TinaTest 是全志科技股份有限公司开发的一套用于 Tina Linux SDK 的测试平台。TinaTest 集成了 Tina SDK 的几乎所有的测试用例，集成多个输出插件以适应各种测试环境，并对外提供简单统一且允许高度自定义的配置方法。

### 2.2 使能 TinaTest 软件包

```
$ make menuconfig
TestTools --->
<*> tinatest..... Test Platform For TinaSDK --->
```

### 2.3 TinaTest 目录结构

TinaTest 的目录位于：

```
tina/package/testtools/tinatest
```

目录结构如下：

```
.
├── config      : TinaTest的kconfig相关文件
├── doc         : TinaTest框架及其测试用例的说明文档
├── src         : TinaTest的源码
├── testcase    : 测试用例的集合
└── tools        : 常用的工具集合，包括快速注册/注销测试用例等
```

#### 2.3.1 测试说明文档

TinaTest 的测试用例说明文档目录位于：

```
tinatest/doc/testcase
```

其目录结构如下：

- .
- |--- base      : TinaTest的功能测试说明文档
- |--- stress    : TinaTest的压力测试说明文档
- |--- spec     : TinaTest的性能测试说明文档

说明文档首先按功能分类，分为功能测试、性能测试、压力测试。在功能之下再按模块进行分类，例如 CPU、存储、显示、电源等等。通过查看对应功能和模块的文档，能够详细了解如何利用 tinatest 进行相应的测试。



## 3 TinaTest 的使用

### 3.1 配置树简述

TinaTest 的配置文件以 json 的格式解析，在设备端的路径为：

```
/etc/tinatest.json
```

在编译 PC 中会根据配置动态生成配置文件，临时文件保存在：

```
tina/out/<方案名>/staging_dir/target/rootfs/etc/tinatest.json
```

配置文件以树状结构排版，例如：

```
"/" : {
    "stress" : {
        "reboot" : {
            "enable" : true,
            "command" : "echo \"===== Going to reboot =====\"; reboot -f",
            "run_times" : 1000,
            "may_reboot" : true
        }
    }
}
```

上例对应的测试用例路径为：

```
/stress/reboot
```

而 enable,command,run\_times,may\_reboot 等为测试用例/stress/reboot 的属性配置项。

更多配置项以及含义参考下文：TinaTest 配置说明。

测试用例路径：测试用例在配置文件的配置树的路径，例如上述的/stress/reboot。

## 3.2 TinaTest 命令说明

### 3.2.1 命令说明

```
tinatest [选项]... [测试用例路径]...
或 tt [选项]... [测试用例路径]...
```

选项：

-p :以树状结构显示配置文件

### 3.2.2 示例

示例中默认只开启了 serial 的输出插件，测试结果只通过命令行终端输出。若在其他测试环境中，可以使能其他输出插件（例如 markdown,dragonmat 等）。

#### 3.2.2.1 显示配置文件

命令 1：

```
root@TinaLinux:/# tinatest -p
```

显示：

```
. (/)
|-- sys
|   |-- global
|   |   |-- info
|   |   |   |-- outlog
|   |   |   |   |-- serial = TRUE
|   |   |   |   |-- markdown
|   |   |   |   |   |-- outdir = "/mnt/UDISK/md"
|   |   |-- limit
|   |   |   |-- run_cnt_up_to = 3
|   |   |   |-- tinatest_run_time = ["0","0","0","356"]
|-- local
|   |-- info
|   |   |-- date = TRUE
|   |   |-- resource = TRUE
|   |-- limit
|   |   |-- run_times = 1
|   |   |-- testcase_run_once_time = ["0","0","0","356"]
|   |   |-- testcase_run_time = ["0","0","0","356"]
|   |   |-- timeout_with_pass = TRUE
|-- demo
|   |-- demo-c
|   |   |-- enable = TRUE
|   |   |-- command = "demo-c"
|   |   |-- date = TRUE
```

```
| | | -- resource = TRUE  
| | | -- run_times = 1  
| | | -- run_alone = TRUE  
| | | -- testcase_run_once_time = ["10"]  
| | | -- testcase_run_time = ["10"]  
| | | -- timeout_with_pass = TRUE  
| | | -- exit_once_failed = TRUE  
| -- demo-sh  
| | | -- enable = TRUE  
| | | -- command = "demo-sh.sh"  
| | | -- date = TRUE  
| | | -- resource = TRUE  
| | | -- run_times = 1  
| | | -- run_alone = TRUE  
| | | -- testcase_run_once_time = ["10"]  
| | | -- testcase_run_time = ["10"]  
| | | -- timeout_with_pass = TRUE  
| | | -- exit_once_failed = TRUE
```

命令 2：

```
root@TinaLinux:/# tinatest -p /demo/demo-sh
```

显示：

```
. (/demo/demo-sh)  
|-- enable = TRUE  
|-- command = "demo-sh.sh"  
|-- date = TRUE  
|-- resource = TRUE  
|-- run_times = 1  
|-- run_alone = TRUE  
|-- testcase_run_once_time = ["10"]  
|-- testcase_run_time = ["10"]  
|-- timeout_with_pass = TRUE  
|-- exit_once_failed = TRUE
```

### 3.2.2.2 执行单个测试用例

只需要在 TinaTest 后指定单个测试用例路径即可（测试用例路径解析见上文：配置树简述）。命令：

```
root@TinaLinux:/# tt /demo/demo-c
```

显示：

```
===== tasks list =====  
/demo/demo-c  
===== end =====  
----- /demo/demo-c -----  
* task path : /demo/demo-c  
* task command : demo-c
```

```
* run times(real/max) : 1/1
* run parallel : no
* run alone : yes
* may reboot : no
* run once time limit : 10s
* run time limit : 10s
* timeout with : pass
* exit once failed : yes
* real-time log: no
* begin date : Thu Jan  1 00:10:04 1970
* end date : Thu Jan  1 00:10:04 1970
* result :
*     num      pid      pgid      return      begin      end      note
*     0       1500     1499           0   00:10:04   00:10:04
* task resource :
*     user cpu time = 0.0
*     system cpu time = 0.0
*     maximum resident size = 448kB
*     page faults break times (without I/O) = 564
*     page faults break times (with I/O) = 0
*     input times = 0
*     output times = 0
*     wait resource actively times = 24
*     wait resource passively times = 9
* run log :
*****config value:
    /demo/demo-c/command = demo-c
system information:
    kernel version: 4.4.89
    target: koto-perf1
----- end -----
===== tasks result =====
/demo/demo-c - YES
===== end =====
```

### 3.2.2.3 执行一类测试用例

以/demo/demo-c 与/demo/demo-sh 为例，都归属于上级树节点：/demo，因此执行一类测试用例如下。

```
root@TinaLinux:/# tt /demo
```

显示：

```
===== tasks list =====
/demo/demo-c
/demo/demo-sh
===== end =====

----- /demo/demo-c -----
* task path : /demo/demo-c
* task command : demo-c
* run times(real/max) : 1/1
```

```
* run parallel : no
* run alone : yes
* may reboot : no
* run once time limit : 10s
* run time limit : 10s
* timeout with : pass
* exit once failed : yes
* real-time log: no
* begin date : Thu Jan  1 00:10:32 1970
* end date : Thu Jan  1 00:10:32 1970
* result :
*     num    pid    pgid    return      begin      end    note
*     0      1516    1515          0  00:10:32  00:10:32
* task resource :
*     user cpu time = 0.0
*     system cpu time = 0.0
*     maximum resident size = 448kB
*     page faults break times (without I/O) = 569
*     page faults break times (with I/O) = 0
*     input times = 0
*     output times = 0
*     wait resource actively times = 26
*     wait resource passively times = 10
* run log :
*****
config value:
    /demo/demo-c/command = demo-c
system information:
    kernel version: 4.4.89
    target: koto-perf1
----- end -----  
----- /demo/demo-sh -----
* task path : /demo/demo-sh
* task command : demo-sh.sh
* run times(real/max) : 1/1
* run parallel : no
* run alone : yes
* may reboot : no
* run once time limit : 10s
* run time limit : 10s
* timeout with : pass
* exit once failed : yes
* real-time log: no
* begin date : Thu Jan  1 00:10:33 1970
* end date : Thu Jan  1 00:10:33 1970
* result :
*     num    pid    pgid    return      begin      end    note
*     0      1529    1528          0  00:10:33  00:10:33
* task resource :
*     user cpu time = 0.0
*     system cpu time = 0.0
*     maximum resident size = 448kB
*     page faults break times (without I/O) = 692
*     page faults break times (with I/O) = 0
*     input times = 0
*     output times = 0
*     wait resource actively times = 25
*     wait resource passively times = 11
* run log :
```

```
*****
key: /demo/demo-sh/command
val: demo-sh.sh

target: koto-perf1
boot_media: emmc
----- end -----

===== tasks result =====
/demo/demo-c - YES
/demo/demo-sh - YES
===== end =====
```

### 3.2.2.4 执行多类测试用例

只需在 tinatest 命令后指定多个测试用例类或测试用例路径，例如：

```
root@TinaLinux:/# tt /demo /stress/reboot /base/production/keytester
```

将同时执行：

1. /demo 类下所有有效测试用例。
2. /stress 类下所有有效测试用例。
3. /base/production/keytester 测试用例。

## 4 TinaTest 配置说明

此章节主要介绍如何修改配置及配置树中的配置项含义，配置树的介绍见上文：配置树简述。

### 4.1 修改配置项

TinaTest 支持两种修改配置项的方式：

1. 直接修改设备端的 /etc/tinatest.json 文件中配置项的值。
2. 通过 menuconfig，以界面形式修改配置项的值。

以方式 1 直接在设备端修改，可直接使用，以方式 2 在 PC 界面间接修改，需要重新编译安装 TinaTest 软件包或烧录固件。

#### 4.1.1 直接修改配置文件

TinaTest 的配置文件以 json 格式解析，修改符合 json 语法且符合配置项类型即可。配置项类型参考下文：配置项介绍。

##### 4.1.1.1 配置格式简述

以/stree/reboot 测试用例为例，配置树如下：

```
"/" : {
    "stress" : {
        "reboot" : {
            "enable" : true,
            "command" : "echo \"===== Going to reboot =====\"; reboot -f",
            "run_times" : 1000,
            "may_reboot" : true
        },
        "reboot-for-note" : {
            "enable" : false
        }
    }
}
```

说明：

1. 配置项基本结构为：“”：。
2. 一对 “{}” 之间内容为下一级配置项。
3. 同一级配置项之间要以逗号间隔，同一级最后一个配置项不需逗号。例如，“run\_times” 下还有 “may\_reboot”，因此 “run\_times” 配置项的最后需要加上逗号。而 “may\_reboot” 作为/stress/reboot 节点下的最后一个配置项，不需要逗号。

#### 4.1.1.2 配置项配置值类型

TinaTest 的配置文件支持：整型、浮点数、布尔型、字符串、字符串数组，5 种类型。

配置值类型	示例
int	“run_times” : 1000
double	“percent” : 3.2
string	“command” : “echo ”=Going to reboot=”; reboot -f”
string-array	“stdin” : [ “input1”, “input2”, “input3” ]
bool	“enable” : false

#### 4.1.1.3 示例

取消测试用例/stress/reboot 的使能，则/stress/reboot 节点下的配置项 “enable” 改为 false，例如：

```
"/" : {
    "stress" : {
        "reboot" : {
            "enable" : false,
            "command" : "echo \"===== Going to reboot =====\"; reboot -f",
            "run_times" : 1000,
            "may_reboot" : true
        },
        "reboot-for-note" : {
            "enable" : false
        }
    }
}
```

#### 4.1.2 通过 menuconfig 修改配置项

进入 TinaTest 的 menuconfig 配置界面：

```
$ make menuconfig
TestTools --->
  <*> tinatest..... Test Platform For TinaSDK --->
```

显示界面如下：

菜单项	说明
System Config	TinaTest 的系统配置，包括配置项默认值，输出插件选择，系统信息采集插件选择等
demo	测试用例 demo
base	基本功能相关的测试用例，其菜单下的子菜单 production 为量产测试用例
spec	性能相关的测试用例
stress	压力老化测试相关的测试用例

#### 4.1.2.1 示例

以测试用例 /stress/reboot 为例，修改执行测试为 555 次，操作流程如下：

1. 选中 Stress 并进入 Stress 的子菜单。
2. 选中 reboot 并进入 reboot 的子菜单。
3. 修改 run\_times 值为 555。

保存并退出 menuconfig，重新编译安装 TinaTest 软件包或编译烧写固件即可。

## 4.2 配置项说明

### 4.2.1 配置项分类

类别名	含义
局部	可在/sys 中定义全局默认值，也可在测试用例节点中为测试用例定制其他值。
全局	只能在/sys 中修改，对所有测试用例有效。
信息	与采集信息相关的配置项。
限制	会修改测试用例的执行行为，例如最多同时执行测试用例次数等。

类别名	配置节点路径
任务相关	/stress/reboot
局部信息	/sys/local/info
局部限制	/sys/local/limit

类别名	配置节点路径
全局信息	/sys/global/info
全局限制	/sys/global/limit

## 4.2.2 配置项

### 4.2.2.1 任务相关

配置项	类型	说明
command	string	调用测试用例的 shell 命令
enable	bool	是否使能
stdin	string-array	字符串数组的每个元素作为对测试用例的一次输入
stdin	string	重定向文件为用例的标准输入 (优先级 stdin > fstdin)

### 4.2.2.2 局部信息

配置项	类型	说明
date	bool	配置是否记录用例开始执行日期和结束日期 (时间)
resource	bool	配置是否记录用例使用资源情况
real_time_log	bool	配置是否实时显示测试用例的 log(默认会收集 log，并在用例结束后一次性显示)

### 4.2.2.3 局部限制

配置项	类型	说明
run_times	int	测试用例循环执行次数 (小于 0 无效)
run_alone	bool	用例单独执行
run_parallel	bool	并行执行 (同时执行 run_times 个用例)
may_reboot	bool	设备在测试用例中可能会重启。 配置是否在发生重启时将测试数据保存到 flash。 保存的数据会在开机过程中加载，避免设备重启导致测试的数据丢失。
testcase_run_once_time	string	单次执行用例执行时长限制，格式：秒分时天。 例如：0 0 0 2 (2 天)
testcase_run_time	string	该用例执行总时长限制，格式：秒分时天。
timeout_with	bool	当测试用例超时，是否判断为执行失败， 格式：pass/failed

配置项	类型	说明
exit_once_failed	bool	当测试用例有一次执行失败就退出

#### 4.2.2.4 全局信息

插件	子配置项	类型	说明
outlog_serial	无	bool	在命令行终端显示测试信息
outlog_markdown	outdir	string	设备端以 markdown 格式保存测试信息的路径
outlog_dragonmat	wait_till_connected exit_when_end	bool bool	等待 PC 和小机端连接后，再执行 当测试完成时退出
collectd_interval_sec	无	int	信息采集间隔
collectd_rrdtool	outdir	string	采集结果以 rrd 数据格式输出的目录
collectd_csv	outdir	string	采集结果以 csv 数据格式输出的目录
collectd_cpu	report_to_percentage	bool	以百分比形式记录采集数据
collectd_memory	report_to_absolute report_to_percentage	bool bool	以绝对值形式记录采集数据 以百分比形式记录采集数据
collectd_df	report_to_absolute report_to_percentage select_or_ignore device mountpoint fstype	bool bool string string string string	以绝对值形式记录采集数据 以百分比形式记录采集数据 忽略匹配项 (ignore)or 选择匹配项 (select) df - 匹配的设备名, "ALL" 表示所有 df - 匹配的挂载点, "ALL" 表示所有 df - 匹配的文件系统, "ALL" 表示所有
collectd_disk	select_or_ignore disk_regular_expression	string	忽略匹配项 (ignore)or 选择匹配项 (select)
collectd_filecount	directory include_hidden include_subdir name size mtime	string bool bool string string string	统计的文件夹 是否包含隐藏文件 是否遍历子目录 匹配文件名 (支持通配符) (参考 find 命令) 匹配文件大小 (参考 find 命令) 匹配文件修改时间 (参考 find 命令)
collectd_ping	host send_interval_sec timeout max_ttl	string string string string	ping - ping 的 ip ping - pint 一次的时间间隔 (<=0 无效) ping - ping 一次的超时时间 ping - ping 一次的 ttl 上限 (0-255)

## 5 测试用例

### 5.1 测试用例分类

#### 5.1.1 一级分类

分类	说明	节点路径
base	基本功能测试，其中/base/production 为量产测试用例	/base
demo	测试用例编写的示例，用于指导编写测试用例	/demo
spec	性能相关的测试	/spec
stress	压力老化测试	/stress

配置文件 tinatest.json 和 Menuconfig 配置界面都按分类归类，配置界面参考：tinatest 配置 1 级界面，配置文件结构类似：

```
"/" : {  
    "base" : {  
        ...  
    },  
    "demo" : {  
        ...  
    },  
    "spec" : {  
        ...  
    },  
    "stress" : {  
        ...  
    }  
}
```

#### 5.1.2 多级分类

允许添加多级分类，例如量产测试用例/base/production/udisktester 的一级分类是 base，二级分类是 production，配置文件结构类似：

```
"/" : {  
    "base" : {  
        "production" : {  
            ...  
        }  
    }  
}
```

```
        }
    },
    "demo" : {
        ...
    }
    "spec" : {
        ...
    }
    "stress" : {
        ...
    }
}
```

## 5.2 添加测试用例

测试用例的源码路径为：

```
tina/package/testtools/tinatest/testcase
```

内部目录结构与测试用例路径（见 3.1 配置树简述）呈现一一对应关系，目录结构如下：

```
.
├── base
│   └── production
│       ├── headphonetest
│       ├── hosttester
│       ├── keytester
│       ├── mictester
│       ├── satatest
│       └── udisktester
└── demo
    ├── demo-c
    └── demo-sh
└── spec
└── stress
```

TinaTest 支持 C/C++ 和 shell 格式的测试用例，支持 C/C++ 源码的自动编译，同时提供快速注册/注销测试用例的脚本工具。添加测试用例有 3 个步骤：

1. 添加源码（可选）
2. 添加测试用例属性文件
3. 调用快速注册脚本工具注册

### 5.2.1 添加测试用例源码（可选）

1. 测试用例源码支持 C/C++ 和 shell 脚本格式。

2.TinaTest 以测试用例的返回值作为测试结果的判定，当返回值为 0，表示测试通过，反之，表示测试不通过。

3. 当测试仅仅是一句命令无须其他源码时，则可不添加源码。以/stress/reboot 为例。

测试只需一行 shell 命令：

```
root@TinaLinux:/# echo "===== reboot now =====" && reboot -f
```

此时不需要添加源码，只需要在测试用例属性文件（见测试用例属性文件（private.conf）及快速注册/注销测试用例）中的设置：

```
command = "echo \\"===== Going to reboot =====\\\"; reboot -f"
```

4. 对复杂的测试用例，必须添加测试源码时，测试用例源码的路径必须与测试用例路径匹配。以测试用例路径 /base/production/udisktester 为例，其源码必须保存在对应路径文件夹中：

```
tina/package/dragontools/tinatest/testcase/base/production/udisktester
```

5.TinaTest 为测试用例提供了部分 API，用于获取配置文件信息和系统信息。获取配置文件 tinatest.json 的配置值：一般在定义了私有配置项时，获取私有配置项的值来控制测试用例的执行流程（见测试用例属性文件（private.conf）及快速注册/注销测试用例）。获取系统信息：获取内核版本，方案，启动介质等。

## 5.2.2 基于 C/C++ 的 API

### 5.2.2.1 与 json 相关的 API

mjson\_fetch:

---

函数原型 struct mjson\_value mjson\_fetch(const char \*keypath);

参数说明 keypath: 测试用例路径

返回说明 失败则 mjson\_value.type == mjson\_type\_error

功能描述 获取任意类型配置项的值

注意事项 务必检查 mjson\_value.type, 当为 mjson\_type\_error 时，此结构体无效，使用会导致段错误

---

mjson\_fetch\_int:

---

函数原型 int mjson\_fetch\_int(const char \*keypath);

参数说明 keypath: 测试用例路径

返回说明 成功返回相应的值，失败返回-1

---

函数原型 int mjson\_fetch\_int(const char \*keypath);

---

功能描述 获取 int 型配置项的值

---

mjson\_fetch\_boolean:

---

函数原型 int mjson\_fetch\_boolean(const char \*keypath);

---

参数说明 keypath: 测试用例路径

返回说明 成功返回 true/false(1/0), 失败返回-1

功能描述 获取 bool 型配置项的值

---

mjson\_fetch\_double:

---

函数原型 double mjson\_fetch\_double(const char \*keypath);

---

参数说明 keypath: 测试用例路径

返回说明 成功返回相应的值, 失败返回-1

功能描述 获取 double 型配置项的值

---

mjson\_fetch\_string:

---

函数原型 char \* mjson\_fetch\_string(const char \*keypath);

---

参数说明 keypath: 测试用例路径

返回说明 成功返回相应的字符串指针, 失败返回 NULL

功能描述 获取字符串型配置项的值

注意事项 务必检查返回, 否则引发段错误。指针内存不需要(不能)释放

---

mjson\_fetch\_array:

---

函数原型 char \*\*mjson\_fetch\_array(const char  
\*keypath);

---

参数说明 keypath: 测试用例路径

返回说明 成功返回相应的二维字符串指针, 失败返回 NULL

功能描述 获取字符串数组型配置项的值

注意事项 务必检查返回, 否则引发段错误, 第一个字符串为有效字符串个数(不算第一个字符串)。字符串示例:  
array={"3", "one", "two", "three"}。可用  
atoi(array[0]) 函数获取有效字符串个数, 指针内存  
不需要(不能)释放

---

### 5.2.2.2 交互 API

使用前提：测试用例需包含头文件： #include “interact.h”

task:

---

函数原型 int task(const char *ask*, char *reply*, int *len*);

---

参数说明 *ask*: 该测试用例向用户提出的问题。

eg1: *ask* = “Please enter the WiFi password:”

*reply*: 指向一块内存空间，用于存放用户输入的回答

*len*: *reply* 指向的内存空间的大小

返回说明 0: 成功 -1: 发送失败

功能描述 该测试用例向用户提出问题 (*ask*)，并获取用户的回答 (*reply*)

交互形式 串口：打印提示信息

DragonMAT: 弹出对话框，用户输入数据后，点击提交

---

ttips:

---

函数原型 int ttips(const char \**tips*)

---

参数说明 *tips*: 该测试用例向用户提示的信息

返回说明 0: 成功 -1: 发送失败

功能描述 该测试用例向用户提示信息

交互形式 串口：打印提示信息

DragonMAT: DragonMAT 界面中，该测试用例对应区域显示 *tips* 字符串

---

ttrue:

---

函数原型 int ttrue(const char \**tips*);

---

参数说明 *tips*: 该测试用例向用户提出的问题

eg1: *tips* = “Could you see this picture?”

eg2: *tips* = “Could you sound the music clearly?”

返回说明 1: 用户选择“是” 0: 用户选择“否” -1: 发送失败

功能描述 该测试用例向用户提问，用户根据问题，选择（是／否）

交互形式 串口：打印提示信息

DragonMAT: 该测试用例对应区域显示 *tips* 字符串，并出现是和否按钮，让用户点击选择

---

tupfile:

---

函数原型 int tupfile(const char *filepath*, const char *tips*);

参数说明 *filepath*: 设备端要上传的文件路径

*tips*: 该测试用例向用户提示的信息

返回说明 0: 成功 -1: 失败

功能描述 将设备端的文件上传到 PC 端。PC 端文件保存目录: Dragonmat\_XXX/result\_dir/0/

交互形式 串口: 暂不支持该接口

DragonMAT: 显示信息“文件上传: *filename*”,

并上传到 PC 端 Dragonmat\_XXX/result\_dir/0/目录

---

tshowing:

---

函数原型 int tshowing(const char *filepath*, const char *tips*);

参数说明 *filepath*: 设备端要上传的图片路径

*tips*: 该测试用例向用户提出的问题

返回说明 0: 用户选择“是” 1: 用户选择“否” -1: 发送失败

功能描述 设备端上传图片到 PC 端, PC 端显示图片并询问用户 *tips* 问题(比如: 图片是否清晰正确?)

交互形式 串口: 暂不支持该接口

DragonMAT:

1. 显示信息“文件上传: *filename*”, 上传图片到 PC 端 Dragonmat\_XXX/result\_dir/0/目录
  2. 自动显示图片
  3. 询问用户, 且出现是和否按钮, 用户点击选择
- 

### 5.2.2.3 其余 API

get\_kernel\_version:

---

函数原型 char \*get\_kernel\_version(void);

参数说明 无

返回说明 成功返回有效字符串, 失败返回 NULL

功能描述 获取内核版本字符串

注意事项 务必检查返回是否为 NULL, 指针内存由测试用例释放, 否则造成内存泄露

---

get\_target:

---

函数原型 char \*get\_target(void);

参数说明 无

返回说明 成功返回有效字符串, 失败返回 NULL

功能描述 获取方案代号。

---

函数原型 char \*get\_target(void);

---

注意事项 务必检查返回是否为 NULL，指针内存由测试用例释放，否则造成内存泄露

---

get\_boot\_media:

---

函数原型 char \*get\_boot\_media(void);

参数说明 无

返回说明 成功返回有效字符串，失败返回 NULL

功能描述 获取引导系统的存储设备

---

注意事项 务必检查返回是否为 NULL，指针内存由测试用例释放，否则造成内存泄露

---

### 5.2.3 基于 shell 的 API

#### 5.2.3.1 与 json 相关的 API

mjson\_fetch:

---

函数原型 mjson\_fetch keypath1 [keypath2 ...]

参数说明 keypath: 测试用例路径配置项

返回说明 成功返回有效字符串，失败无任何显示

功能描述 获取配置项的值，例如：mjson\_fetch /demo/demo-c/run\_times

---

get\_kernal\_version:

---

函数原型 get\_kernel\_version

参数说明 无

返回说明 成功返回有效字符串，失败无任何显示

功能描述 获取内核版本，例如 3.10.65

---

get\_target:

---

函数原型 get\_target

参数说明 无

返回说明 成功返回有效字符串，失败无任何显示

功能描述 获取方案代号，例如 azalea-m2ultra

---

get\_boot\_media:

---

函数原型 get\_boot\_media

---

参数说明 无

返回说明 成功返回有效字符串，失败无任何显示

功能描述 获取引导系统的存储设备，emmc/sdcard/nand/nor-flash，其中 sdcard 代表卡启动

---

### 5.2.3.2 交互 API

使用方法与 C 接口类似。其交互形式与 3.2 中 C 接口交互形式相同。

task:

---

命令 task “as”

---

参数说明 ask: 该测试用例向用户提出的问题。

eg: task Please enter the WiFi password:

返回说明 0: 成功 1: 失败

功能描述 该测试用例向用户提出问题 (ask)，并打印出用户的回答。

---

ttips:

---

命令 ttips “tips”

---

参数说明 tips: 该测试用例向用户提示的信息

返回说明 0: 成功 1: 失败

功能描述 该测试用例向用户提示信息。

---

ttrue:

---

命令 ttrue “tips”

---

参数说明 tips: 该测试用例向用户提出的问题

返回说明 0: 用户选择“是” 1: 用户选择“否”

功能描述 该测试用例向用户提问，用户根据问题，选择（是／否）

---

tupfile:

---

命令 tupfile “filepath” “tips”

---

参数说明 filepath: 设备端要上传的文件路径

tips: 该测试用例向用户提示的信息

返回说明 0: 成功 1: 失败

---

命令        `tupfile "filepath" "tips"`

---

功能描述    将设备端的文件上传到 PC 端。PC 端文件保存目录：`Dragonmat_XXX/result_dir/0/`

---

tshowing:

---

命令        `tshowimg "filepath" "tips"`

---

参数说明    `filepath`: 设备端要上传的图片路径  
                `tips`: 该测试用例向用户提示的信息

返回说明    0: 用户选择“是”    1: 用户选择“否”

功能描述    设备端上传图片到 PC 端，PC 端显示图片并询问用户 tips 问题  
(比如：图片是否清晰正确？)

---

## 5.2.4 测试用例属性文件 (`private.conf`) 及快速注册/注销测试用例

`private.conf` 为测试用例的属性配置文件，每一个测试用例都应有一个对应的 `private.conf`，放在测试用例源码目录，`private.conf` 的编写规则见图 5-1 `add_testcase.sh` 使用说明快速注册/注销测试用例的脚本工具位于：

```
tina/package/dragontools/tinatest/tools
```

快速注册测试用例只需要执行：

```
./add_testcase.sh <测试用例属性配置文件 (private.conf路径) >
```

快速注销测试用例只需要执行：

```
./del_testcase.sh <测试用例属性配置文件 (private.conf路径) >
```

### 5.2.4.1 add\_testcase.sh 使用说明

```
[GMPY@11:50 tinatest]$./tools/add_testcase.sh
使用说明:
    add testcase.sh <配置文件1> [配置文件2] ...

配置文件:
    记录新用例的路径以及默认配置值,一行一条键值对,格式为:
        <配置项>[:类型] = <配置值>
    [配置项]: 包含PATH/ENABLE/INFO/LIMIT/DEPENDS和测试用例的所有配置项(例如:command,run_times,run_alone等)
    其中:
        PATH: 测试用例在配置树中的绝对路径(字符串)
        ENABLE: 默认是否使能此用例(bool)
        INFO: 默认是否使能所有的 局部信息 配置项(bool)
        LIMIT: 默认是否使能所有的 局部限制 配置项(bool)
        DEPENDS: 测试用例依赖的第三方应用包(string),多个包之间以逗号间隔
            格式: "<依赖的软件包1>[,<依赖的软件包2>,...]"
            例如 /stress/rw/rw-auto 依赖 rwcheck 软件包, 则 DEPENDS="rwcheck"
    [大写字母为特定配置项][无指定类型的小写字母为用例属性项][指定类型的小写字母为私有配置项]
    [类型]: (私有配置项才需要)为mjson支持的数据类型,包括:int,double,true/false/string/array
    [配置值]: 支持字符串/字符串数组/整数/浮点数/bool(见示例)
    其中:
        1) 字符串必须用双引号括起来
        2) 字符串数组以字符串形式表示,字符串之间以空格间隔
        3) 字符串内若双引号\转义字符等,需要有双重转义, 例如: command = "echo \\\\"test\\\\\"" 表示echo "test"
        4) 每一行开头不能有空格/Tab等
```

示例如下:

```
|PATH = /demo/demo-c
|ENABLE = false
|INFO = true
|command = "demo-c"
|run_times = 10
|run_alone = false
|workers:int = 2
|words:string = "test"
|right:bool = true
|str_array:array = "one two three"
[GMPY@11:53 tinatest]$
```

图 5-1: add testcase 显示界面

例如, 在添加了/demo/demo-c 的源码 demo-c.c 后, 编写的 private.conf 如下:

```
PATH = "/demo/demo-c"
INFO = false
LIMIT = false
command = "demo-c"
run_times = 3
```

表示:

1. 默认关闭局部信息相关的配置项
2. 默认关闭局部限制相关的配置项
3. 默认执行测试用例的shell命令是: demo-c
4. 默认执行次数是3次

执行注册测试用例工具:

```
$ ./tools/add testcase.sh testcase/demo/demo-c/private.conf
```

在正确执行了 add testcase.sh 后, 主要在 kconfig 中修改和添加对应的 Config.in 文件。

```
tinatest/config/
├── demo
│   ├── Config.in
│   └── demo-c
└── Config.in
```

注册测试用例后，即可通过界面命令进行使能，配置等：

```
$ make menuconfig
```

### 5.2.4.2 del\_TestCase.sh 使用说明

```
[GMPY@11:53 tinatest]$ ./tools/del_TestCase.sh
使用说明:
  del_TestCase.sh <配置文件1> [配置文件2] ...

配置文件:
  记录新用例的路径以及默认配置值,一行一条键值对,格式为:
    <配置项>[:类型] = <配置值>
  [配置项]: 包含PATH/ENABLE/INFO/LIMIT/DEPENDS和测试用例的所有配置项(例如:command,run_times,run_alone等)
  其中:
    PATH: 测试用例在配置树中的绝对路径(字符串)
    ENABLE: 默认是否使能此用例(bool)
    INFO: 默认是否使能所有的 局部信息 配置项(bool)
    LIMIT: 默认是否使能所有的 局部限制 配置项(bool)
    DEPENDS: 测试用例依赖的第三方应用包(string),多个包之间以逗号间隔
      格式: "<依赖的软件包1>[,<依赖的软件包2>...]"
      例如 /stress/rw/rw-auto 依赖 rwcheck 软件包, 则 DEPENDS="rwcheck"
    [大写字母为特定配置项][无指定类型的小写字母为用例属性项][指定类型的小写字母为私有配置项]
  [类型]: (私有配置项才需要)为json支持的数据类型,包括:int,double,true/false/string/array
  [配置值]: 支持字符串/字符串数组/整数/浮点数/bool(见示例)
  其中:
    1) 字符串必须用双引号括起来
    2) 字符串数组以字符串形式表示,字符串之间以空格间隔
    3) 字符串内若双引号\转义字符等,需要有双重转义,例如: command = "echo \\\"test\\\""
       表示echo "test"
    4) 每一行开头不能有空格/Tab等

示例如下:
  |PATH = /demo/demo-c
  |ENABLE = false
  |INFO = true
  |command = "demo-c"
  |run_times = 10
  |run_alone = false
  |workers:int = 2
  |words:string = "test"
  |right:boolean = true
  |str_array:array = "one two three"
All Done!
[GMPY@11:54 tinatest]$
```

图 5-2: del testcase 显示界面

**特殊使用场景一：**在执行 add\_TestCase.sh 注册测试用例后，若要修改测试用例路径（即移动测试用例源码文件夹），请务必在移动前执行 del\_TestCase.sh 注销，并在移动后重新执行 add\_TestCase.sh 注册 add\_TestCase.sh 与 del\_TestCase.sh 的解析与测试用例所在的源码目录路径是有绑定关系的，因此移动 private.conf 前必须先 del，移动后再 add。

## 5.2.5 测试用例源码自编译

在注册测试用例后，只要在 make menuconfig 中使能了相应的测试用例，则会在编译 TinaTest 的过程中自动编译测试用例源码，编译逻辑如下：

### 5.2.5.1 源码中有 Makefile

若测试用例源码路径中有 Makefile，则调用源码的 Makefile 进行编译。源码 Makefile 中有效的变量有：

1. 方案相关：ARCH
2. 编译器相关：AR、CC、C++
3. 编译标志相关：CFLAGS、LDFLAGS
4. 安装路径相关：PREFIX（安装可执行文件的路径前缀）、PREFIX\_DATA（安装数据文件的路径前缀）

### 5.2.5.2 源码中无 Makefile

若测试用例源码路径中无 Makefile，则使能自动编译。

1. Shell脚本则直接安装。
2. 单个C/C++源码文件则编译成对应可执行文件，并自动安装。  
例如：demo.c.c 则编译成 demo-c，自动安装到相对小机目录的/usr/bin中。

## 著作权声明

版权所有 © 2021 珠海全志科技股份有限公司。保留一切权利。

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

## 商标声明

  **全志科技**  (不完全列举) 均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标，产品名称，和服务名称，均由其各自所有人拥有。

## 免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。