A20 Dragonboard 测试用例编写说明书

V1.1

2014-06-03

Revision History

Version	Date	Author	Changes compared to previous issue
V1.1	2014-06-03		Create

目录

1. 前言

1.1. 简介

DragonBoard 是一个基于 Linux BSP，集成了 DirectFB 的图形化板卡测试系统。该系统旨在检测板卡能否在特定的环境中正常工作。

DragonBoard 系统默认启动两个进程，分别是 core 进程和 launcher 进程，后者是前者的子进程。测试任务由模块测试程序完成，由 launcher 进程负责加载，该进程会根据 test_config.fex 文件的配置创建（fork）新的进程运行测试用例。目前测试用例支持 C、shell 脚本编写。

1.2. 目的

本文档主要介绍如何编写一个 DragonBoard 系统的模块测试用例。撰写的比较匆忙，欢迎大家指正文中的错误。

1.3. 名词解释

1.4. 参考文档

2. 快速入门

目前 DragonBoard 系统只需要在 lichee 的根目录运行./build.sh –p sun7i_dragonboard && ./build.sh pack 就可以编出一个可以调试的固件。下面的章节会介绍如何编写一个测试用例。

2.1. 目录结构

DragonBoard 放在 buildroot/target 目录下，其目录结构大致如下：

```
|-- output/             # 输出目录
    |-- bin/            # 测试用例（程序）输出目录
|-- rootfs/             # 根目录
    |-- dragonboard/
        |-- bin/        # 测试用例（程序）rootfs 的输出目录
|-- src/                # 源码目录
    |-- core/           # 系统核心模块
    |-- include/        # 系统公共头文件
    |-- lib/            # 系统公共库
    |-- testcases/      # 测试用例源码目录
        |-- example/    # 示例
            |-- example.c
            |-- Makefile
        |-- Makefile
    |-- view/           # UI
    |-- Makefile        # 顶层 Makefile
    |-- rule.mk         # 编译变量，include by Makefile
|-- sysroot/            # 交叉编译环境依赖目录
|-- build.sh            # 生成 rootfs.ext4
|-- README.txt
```

2.2. C 测试用例

1. 创建模块测试用例的目录

```
$ cd buildroot/target/dragonboard
$ mkdir src/testcases/example
$ cd src/testcases/example
```

2. 编写测试用例源码 example.c

```
参照 4.2 节。
```

3. 编写 Makefile

```
参照 4.2 节。
```

4. 让系统编译你的模块测试用例

在 testcases/Makefile 中添加

```
all:
    make -C example
```

2.3. Shell 测试用例

Shell 测试用例就是用 shell 脚本完成模块的测试任务。launcher 进程会向脚本传递 4 个参数，分别是：$0 是脚本的名称，$1 是 DragonBoard 系统版本号，目前没有使用，$2 是配置脚本的共享内存 id，$3 是当前测试用例的 id，返回结果的时候会使用。下面是一个简单的示例（memtester）：

```
#!/bin/sh

memtester 128K 1 > /dev/null

echo "$3 $?" >> /tmp/cmd_pipe
```

为了编译的时候能够将脚本拷贝到 output/bin 目录下，我们还需要一个 Makefile，如：

```
# define sources root directory before everything
SRC_ROOT := ../..

# include rule.mk
include $(SRC_ROOT)/rule.mk

.PHONY: all
all:
    cp memtester.sh $(BINDIR)/
```

然后在 testcases/Makefile 添加：

```
all:
    make -C memtester
```

更加详细的说明参照 4.3 节。

2.4. 编译

1. 在 lichee 根目录下编译 DragonBoard 的内核时，即执行：

```
$ ./build.sh -p sun7i_dragonboard
```

会调用 dragonboard 目录下的 build.sh 脚本译 DragonBoard 系统，并生成 rootfs.ext4。

2.5. 打包烧录

DragonBoard 系统的打包烧录流程和 Linux 相似，下面简要描述一下 A20 sugar-ref001 的打包流程：

```
$ cd $lichee
$ ./build.sh pack
```

```
Start packing for Lichee system

All valid chips:
0. Sun7i
Please select a chip:0
All valid platforms:
0. android
1. dragonboard
2. linux
Please select a platform:1
All valid boards:
0. Sugar-ref001
Please select a board:0
```

打包生成的固件放在 tools/pack 目录下，用 phoenixCard 制作启动卡。

3. 配置脚本

DragonBoard 系统使用一个配置脚本（test_config.fex）来完成系统和各个模块的配置，core 进程起来的时候会去解释这个配置脚本，并将其内容按照固定的格式放在一块共享内存里，方便各个测试用例的进程访问。在 C 语言中，这块共享内存的访问 id 通过 main 函数的参数 argv[2]传递到各个测试用例中。目前没有找到 shell 脚本访问这块共享内存的方法，后期将提供其他的解决方案。

test_config.fex 的格式类似 INI 文件格式，可以说是一种扩展。下面是一个测试模块配置示例：

```
[example]
display_name= "Example"
activated   = 1
program     = "example.sh"
category    = 0
run_type    = 1
```

- **display_name**

显示到界面的名称，字符串类型，如果需要显示双引号，可以使用以下语法：

```
display_name= string:"Example"
```

区别在于后者会显示"string:"后面所有的字符，包括空格和制表符。该项能够容纳 63 个英文字符，31 个中文字符。如果 display_name 为空，其它项无效。

- **activated**

0：不测试该模块；1：测试该模块。

- **program**

模块的测试程序，该项能够容纳 15 个英文字符。用于指定模块测试的入口。

- **category**

0：自动化测试模块；1：手动测试模块。

- **run_type**

0：等待当前模块的测试程序执行完毕再运行下一个模块的测试程序；1：不等待当前模块的测试程序执行完毕。一般为了提高整体的测试速度，对于耗时较长的测试程序建议填 0，反之填 1；注意，当，category = 1，即手动测试模块时，该项无效。

以上是一些基础的配置项，用户亦可根据需要添加自己的配置项。

4. 测试用例

4.1. 与 core 交互

DragonBoard 系统的 core 进程负责接收测试用例返回的结果，目前这是与 core 唯一的交互方式，通过命名管道（/tmp/cmd_pipe）实现。core 进程会循环地一行一行地读取该管道，因此要求测试用例返回结果时必须按行的方式写入。

吐槽一下，建议测试用例的程序名称以 tester 或者 tester.sh 结尾。

4.2. C 测试用例

以下是一个简单的 C 测试用例：

```
/*
 * \file        example.c
 * \brief       just an example.
 *
 * \version     1.0.0
 * \date        2012 年 05 月 31 日
 * \author      James Deng <csjamesdeng@allwinnertech.com>
 *
 * Copyright (c) 2012 Allwinner Technology. All Rights Reserved.
 *
 */

/* include system header */
#include <string.h>

/* include "dragonboard_inc.h" */
#include "dragonboard_inc.h"

/* C entry.
 *
 * \param argc the number of arguments.
 * \param argv the arguments.
 *
 * DO NOT CHANGES THE NAME OF PARAMETERS, otherwise your program will get
 * a compile error if you are using INIT_CMD_PIPE macro.
 */
int main(int argc, char *argv[])
```

```
{
    char binary[16];

    /* init cmd pipe after local variable declaration */
    INIT_CMD_PIPE();

    strcpy(binary, argv[0]);
    db_msg("%s: here is an example.\n", binary);

    /* send OK to core if test OK, otherwise send FAIL
     * by using SEND_CMD_ PIPE_FAIL().
     */
    SEND_CMD_PIPE_OK();

    return 0;
}
```

几个需要注意的地方：

1. 必须包含 dragonboard_inc.h，里面定义了与 core 交互的宏，建议用户直接使用；
2. main 函数的参数个数和名字固定，不建议更改；
3. main 函数的参数 argc = 4，表示 args 有 4 个值，args[0]是测试程序的名称，args[1]是 DragonBoard 系统版本号，目前没有使用，args[2]是配置脚本的共享内存 id，args[3]是当前测试用例的 id，返回结果的时候会使用；
4. 建议使用 SEND_CMD_PIPE_OK()和 SEND_CMD_ PIPE_FAIL()返回结果。

编译链接：

```
# define sources root directory before everything
SRC_ROOT := ../..

# change compiler and linker option before you include rule.mk
#
# link to libscript.a when you need to fetch configuration
# from test_script
#
#CFLAGS := $(CFLAGS) -g
LDFLAGS := -lscript

# include rule.mk
include $(SRC_ROOT)/rule.mk

# define objects collection variable
example_objs = example.o
```

```
# add your target(s) to all
.PHONY: all
all: example

# define you target, the target will be output to dragonboard/output/bin
# directory
example: $(example_objs)
    $(LINK_MSG)
    $(LINKX)

# change 'example_objs' to your objects collection variable
$(example_objs): %.o: %.c
    $(COMPILE_MSG)
    $(COMPILEX)
```

上面是一个 Makefile 的示例，用户可以根据里面的注释修改。如果想让您的测试用例随 DragonBoard 系统一同编译，那么需要在 testcases/Makefile 中添加以下语句：

```
all:
    make -C example
```

将 example 换成你的模块测试用例所在目录。

4.3. Shell 测试用例

1. shell 脚本的参数

```
$0：脚本的名称；
$1：DragonBoard 系统版本号，目前没有使用；
$2：配置脚本的共享内存 id，目前没有找到在 shell 脚本使用共享内存的方法；
$3：是当前测试用例的 id，返回结果的时候会使用；
```

2. 返回结果

测试用例通过向命名管道（/tmp/cmd_pipe）写入一行字符的方式通知当前模块的测试结果，可以通过 echo 重定向实现：

```
echo "$3 $?" >> /tmp/cmd_pipe
```

$?是脚本中上一个命令的返回值，注意一般返回 0 表示 OK，非 0 表示 Fail。请使用“>>”，而不是“>”。“>>”表示把内容追加到/tmp/cmd_pipe 末尾，这样就不会影响其他模块的结果；“>”表示把内容写入/tmp/cmd_pipe，可能清除其他模块的结果。

3. 测试过程的调试信息

不建议将测试过程的调试信息输出到串口，可以重定向到/dev/null，例如：

```
memtester 128K 1 > /dev/null
```

5. Declaration

This **A20 dragonboard 测试用例编写说明书** is the original work and copyrighted property of Allwinner Technology ("Allwinner"). Reproduction in whole or in part must obtain the written approval of Allwinner and give clear acknowledgement to the copyright owner.

The information furnished by Allwinner is believed to be accurate and reliable. Allwinner reserves the right to make changes in circuit design and/or specifications at any time without notice. Allwinner does not assume any responsibility and liability for its use. Nor for any infringements of patents or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Allwinner. This datasheet neither states nor implies warranty of any kind, including fitness for any particular application.