**Git-Openh264-Conformance Test**

Contents

[**1.** **总概** 2](#_Toc376805008)

[**2.** **测试目录说明** 2](#_Toc376805009)

[**3.** **测试脚本文件使用说明** 4](#_Toc376805010)

[**4.** **脚本语法** 5](#_Toc376805011)

[**5.** **其他** 6](#_Toc376805012)

1. **总概**

* **一致性测试**

一致性测试就是比较两个版本的encoder 在相同的case下输出码流是否一致。在脚本中的一致性比较，是用修改更新的codec(下面用**codec\_target**表示)，同之前经过测试通过的codec(下面用**codec\_benchmark**表示)，比较两者的码流在配置文件中的所有case下的码流是否一致。

* **功能说明**

1. 过配置文件，来修改测试所使用的case

2. 过配置文件，来修改测试所使用的测试序列

3. 过配置文件，修改用于比较的codec的github地址，或者svn 地址以及版本

（该脚本采用git 管理，可以切换不同的branch来选用git 或者svn）

4.自动下载指定的codec,并自动build

5. 自动生成配置文件对应的case文件（如XX.\_case.csv;

6.自动测试所有序列的所有case,并自动生成测试报告，

7. 可结合CI系统，集成为自动化的测试环境

1. **测试目录说明**

* **测试脚本目录结构**

**--TestScript**

-- run\_GenerateCase.sh --run\_OneSequenceAllCase.sh

**bats**

**cfg**

**exe**

--run\_CodecUpdate.sh

-- run\_Main.sh

-- run\_PrepareTestSpace.sh

-- run\_TestAllSequence.sh

--case.cfg

Codec\_target

Codec\_benchmrk

* **脚本+测试数据目录结构**

**-- Source-openh264-git**

run\_CodecUpdate.sh语句SourDir="../Source-openh264-git"指定，可修，为codec\_target的源文件）

**--TestScript**

-- run\_GenerateCase.sh --run\_OneSequenceAllCase.sh

**bats**

**cfg**

**exe**

--run\_CodecUpdate.sh

-- run\_Main.sh

-- run\_PrepareTestSpace.sh

-- run\_TestAllSequence.sh

--case.cfg

Codec\_target

Codec\_benchmrk

*由run\_Main.sh的函数runCheckoutAllTestPassStatus（）指定*

AllTestReport.csv

Issue

testlog

**result**

***由run\_PrepareTestSpace.sh以及case.cfg配置文件共同配置生成***

**Xxx01.yuv**

**AllTestData**

**TestSetCIF**

**Xxx02.yuv**

**TestSet720P**

**result**

**...**

**issue**

1. **测试脚本文件使用说明**

* **Codec的自动下载和build**

执行 ./run\_CodecUpdate.sh $GitAddr

如 ./run\_CodecUpdate.sh <https://github.com/cisco/openh264>

脚本文件工作流程：

1. 从$GitAddr 下载codec代码，自动build,
2. 删除./exe/codec\_target所有文件
3. 将build成功的 .exe .a 等文件拷贝到 ./exe/codec\_target 目录下
4. 将codec中的配置文件 ../testbin/\*.cfg 拷贝到./exe/codec\_target目录下
5. 为统一两个codec的配置文件，用./exe/codec\_target下最新的\*.cfg文件覆盖./exe/codec\_benchmark下的配置文件

* **测试数据空间的创建**

执行 ./ **run\_PrepareTestSpace.sh** $ConfigureFile

如 . /**run\_PrepareTestSpace.sh**  ./cfg/case.cfg

创建空间即为第二部分提及的文件目录AllTestData

脚本文件工作流程：

1. 根据配置文件，解析测试序列集包含的测试序列信息
2. 为每一个序列集(TestSet)创建一个目录
3. 对每一个YUV序列，在其对应的序列集目录下建一个YUV目录
4. 将./exe ./cfg 以及./bats目录下的每一个文件拷到每一个YUV目录下
5. 在每一个YUV目录，调用run\_GenerateCase.sh，为其生成对应的case文件

* **Case文件的生成**

执行 ./run\_GenerateCase.sh $ConfigureFile $OutputFile $TestSetName

如./run\_GenerateCase.sh ./cfg/case.cfg **\**

TestSet\_CIF\_Cisco\_392X192.yuv\_AllCase.csv \

CIF

将配置文件的配置信息转化为一个参数排列组合而成的二维case矩阵，以.csv数据文件作为输出。

脚本文件工作流程：

1. 根据YUV所在目录，解析当前YUVName，TestSetName等信息

如 ./AllTestData/TestSetCIF/Cisco\_392X192.yuv 目录信息，可解析出上述信息

1. 根据配置文件，解析编码参数的对应参数集
2. 排列组合，生成输出文件

* **单个序列所有case的测试**

执行 ./ run\_OneSequenceAllCase.sh （无参数）

对当前YUV目录下的case文件调用，对每一个case进行一次编码比较。caseFile=${TestSetIndex}\_${TestSequenceName}.csv

脚本文件工作流程：

1. 根据YUV所在目录，解析当前YUVName，TestSetName，PicW,PicH等信息,同时对YUV的文件位置进行确认，是否存在测试序列
2. 解析每一行case,将其与编码命令行参数意义匹配；
3. 对codec\_target 和codec\_benchmak分别编码比较同一case下的输出码流是否一致
4. 对所有case的比较结果进行统计记录并将结果放到../result 目录下

* **./run\_Main.sh**

执行 ./run\_Main.sh $ConfigureFile

如 ./run\_Main.sh ./case.sh

执行出run\_CodecUpdate.sh 外的所有功能

1. 创建AllTestData 测试目录；
2. 对所有的YUV进行所有case 的测试
3. 对所有的测试结果进行总结，并将结果放到./result 目录下，生成AllTestReport.csv文件
4. **脚本语法**
5. **If [[ ]] 正则表达式**
6. **Echo $string | awk ' ' 文本信息提取**
7. **一维数组及伪二维数组使用技巧**
8. **变量的变量**
9. **其他**