
# AOSP12 龙芯架构移植功能测试

| | |
|---|---|
| 文件名 | AOSP12 龙芯架构移植功能测试 |
| 版本 | |
| 最后更新日期 | 2024-10-20 |
| 作者 | 熵核科技 |

# 目录

1 移植内容 . . . . . 4
1.1 工具链 . . . . . 4
1.2 AOSP . . . . . 4
2 工具链测试 . . . . . 5
2.1 测试准备 . . . . . 5
2.2 Clang 单元测试 . . . . . 6
2.3 LLVM 单元测试 . . . . . 7
2.4 codegen 单元测试 . . . . . 7
2.5 LLD 单元测试 . . . . . 7
2.6 crt 单元测试 . . . . . 8
2.7 Builtins 单元测试 . . . . . 8
2.8 check-llvm-tools 单元测试 . . . . . 8
2.9 thinlto 单元测试 . . . . . 9
2.10 Clang_c 单元测试 . . . . . 9
2.11 Check_cxx 单元测试 . . . . . 9
2.12 Clang_cxx 单元测试 . . . . . 10
2.13 cxxabi 单元测试 . . . . . 10
2.14 check_clang_tools 单元测试 . . . . . 10
2.15 CFI 单元测试 . . . . . 11
3 AOSP 基础模块测试 . . . . . 11
3.1 bionic 测试 . . . . . 11
3.2 ART 测试 . . . . . 12
3.2.1 ART host gtest 测试 . . . . . 12
3.2.2 ART device gtest 测试 . . . . . 13

3.2.3 ART device java 测试 . . . . . 14
3.3 system 测试 . . . . . 15
3.3.1 system-core 测试 . . . . . 15
3.3.2 system-libbase 测试 . . . . . 16
3.3.3 system-unwind 测试 . . . . . 16
3.3.4 system-other 测试 . . . . . 17
3.4 framework 测试 . . . . . 17

# 1 移植内容

## 1.1 工具链

下图列出了工具链部分，熵核需要完成的工作内容：

| Android编译工具链集 | | | | |
|---|---|---|---|---|
| 1.1 | 整个环境以及代码设置 | AOSP12的整体代码以及repo环境设置 | 0.5 | 熵核（龙芯辅助） |
| 1.2 | clang(15.0.3,或者15.0.x版本) | 此部分内容有龙芯依据Clang基础代码将其编译为Android12需要的工具链，并通过测试（熵核给出编译、测试步骤） | 1.5 | 龙芯 |
| 1.3 | rustc（1.51.0） | 此部分内容有龙芯依据基础源代码,并编译为系统需要的prebuild（熵核给出编译、测试步骤） | -1 | 龙芯 |
| 1.4 | gcc (prebuilts/gcc/linux-x86/riscv64) | 用于部分S文件的汇编，且会用到其中部分o文件。版本10.x或者老一点的版本（此部分内容有龙芯提供二进制），且用于编译Kernel | 0 | 龙芯 |
| 1.5 | ndk (只需要4个二进制文件，但需要编译NDK才可以生成) | 移植NDK基础部分,以及gdbserver（prebuilts/misc/gdbserver/android-riscv64/） | 2.5 | 熵核 |
| 1.6 | bionic库 - 初始移植，这个库在编译Clang以及NDK时是必须的 | 移植Bionic的基础部分 | -3 | 龙芯 |
| 1.7 | runtime | mainline库以及APIs | 1.5 | 熵核 |
| 1.8 | uboot或则其他类似功能软件 | 此部分内容有龙芯提供 | 0 | 龙芯 |
| 1.9 | 龙芯的某一款硬件开发板为目标的编译 | 熟悉龙芯机器以及编译烧录环境 | 1 | 熵核 |
| 1.10 | 龙芯的某一款硬件开发板为目标的编译 | 上述龙芯负责的内容，熵核提供支持（1.1， 1.2，1.3，1.6） | 1.5 | 熵核 |

其中熵核主要负责编译出来 AOSP 环境下需要的 Clang 工具链以及 NDK，对用的 Clang 工具链的实际功能性补丁文件有龙芯提供的。本部分的测试请参考本文第二章。

## 1.2 AOSP

下图是合同中关于 AOSP 基础模块的移植内容：

| | | | | |
|---|---|---|---|---|
| **基础库** | | | | |
| 3.1 | bionic | 相当于Android自己的libc/libm，正常移植（在1.7中仅仅为了编译工具链） | -3 | 龙芯 |
| 3.2 | Art虚拟机 - 解释执行 | 解释方式执行Art虚拟机（必须的模块，速度慢，但是功能完整） | 5 | 熵核 |
| 3.4 | system: iorap, apex, nvram, logging, netd, bt, security, extras, connectivity, memory/libmemunreachable, libhwbinder, libbase, core | system模块(这里需要跟Linux内核合作调试) | 3 | 熵核 |
| 3.5 | framework: base, native,proto_logging等 | framework | 1.5 | 熵核 |
| 3.6 | 基础库测试 | 上述各个基础库模块的测试 | 2 | 熵核 |
| **外部模块库** | | | | |
| 4.1 | rust/crates/libc | Arch相关移植 | -0.5 | 龙芯 |
| 4.2 | scudo, perfetto, mdnsresponder, crosvm, grpc-grpc, minijail, vixl, boringssl, llvm, avb | 这些基础模块的基本支持 | 2.5 | 熵核 |
| 4.3 | 外部模块测试 | 上述各个外部库模块的测试 | 1 | 熵核 |
| **应用与测试** | | | | |
| 5.1 | 基础测试 | 基础单元测试：bionic, art, hw interface 等 | 1.5 | 熵核+龙芯 |
| 5.2 | 集成测试 | 简单集成应用测试：联系人，拨号，时钟等 | 1 | 熵核 |

本部分的测试请参考本文第三章。在此合同基础上，熵核做了很多不在上述列表中的模块移植，具体请参考 M12 验收文件。而外面模块的移植主要是可以编译，且对上述模块中其中必要的文件进行移植。

# 2 工具链测试

Clang 的测试分为 2 类：

1. 回归测试
2. 单元测试

回归测试与单元测试的代码分别位于 llvm/test 和 llvm/unittests。

## 2.1 测试准备

在进行单元测试前，需要先设置环境。由于最终打包的是 stage 2 的 CLANG，所以单元测试均在 stage 2 的编译目录中完成。由于 stage 2 中缺乏部分单元测试所需的库文件，测试前需先复制缺少的文件或建立符号链接。

以下操作以编译工作目录即 $ATOOLCHAIN_WS/clang-15.0.3 为初始目录，步骤如下：

```
cd $LA_WS/clang_la
```

```
# copy libxml2 from stage2-install to stage2
cp $CLANG_OUT/stage2-install/lib/libxml2* $CLANG_OUT/stage2/lib/

# copy libc++ from stage2-install to stage2
cp $CLANG_OUT/stage2-install/lib/libc++.* $CLANG_OUT/stage2/lib/

# copy libc++abi from stage2-install to stage2
cp $CLANG_OUT/stage2-install/lib/libc++abi.* $CLANG_OUT/stage2/lib/

# copy llvm-lit from stage1 to stage2
cp $CLANG_OUT/stage1/bin/llvm-lit $CLANG_OUT/stage2/bin/

# prepare crt files
cd prebuilts/gcc/linux-x86/host/x86_64-linux-glibc2.17-4.8/sysroot/usr/lib

# copy crtbegin.o from GCC 4.8.3 to sysroot
cp ../../../lib/gcc/x86_64-linux/4.8.3/crtbegin.o .

# copy crtend.o from GCC 4.8.3 to sysroot
cp ../../../lib/gcc/x86_64-linux/4.8.3/crtend.o .

# copy libgcc from GCC 4.8.3 to sysroot
cp ../../../lib/gcc/x86_64-linux/4.8.3/libgcc.a .

# copy libgcc_s from GCC lib64 to sysroot
cp ../../../x86_64-linux/lib64/libgcc_s.so.1 .

# done, go back to src working directory
cd -
```

所有测试之前，执行测试的终端中必须先执行一次：export LD_LIBRARY_PATH=$CLANG_OUT/stage2/lib:$LD_LIBRARY_PATH

## 2.2 Clang 单元测试

按以下步骤进行 Clang 单元测试：

```
# run following unit test in $OUT_DIR/stage2 directory
cd $CLANG_OUT/stage2

# run clang unit test with ninja
ninja check-clang
```

结果如下（Linux）：

```
Testing Time: 191.67s
  Skipped        :   33
  Unsupported    :   551
  Passed         : 30479
  Expectedly Failed:   26
```

## 2.3 LLVM 单元测试

按以下步骤进行 LLVM 单元测试：

```
# run llvm unit test with ninja
ninja check-llvm
```

结果如下（Linux）：

```
[0/1] Running the LLVM regression tests

Testing Time: 80.34s
  Skipped        :    11
  Unsupported    : 12530
  Passed         : 36592
  Expectedly Failed:    64
```

## 2.4 codegen 单元测试

按以下步骤进行 codegen 单元测试：

```
# run codegen unit test with ninja
$ ninja check-llvm-codegen
```

结果如下（Linux）：

```
[0/1] Running lit suite /data2/wendong/aclang_toolchain/clang-toolchain/clang-15.0.3/out/llvm-project/llvm/test/CodeGen
Testing Time: 34.61s
  Unsupported    :  9354
  Passed         : 10851
  Expectedly Failed:    20
```

## 2.5 LLD 单元测试

按以下步骤进行 LLD 单元测试：

```
# run lld unit test with ninja
ninja check-lld
```

结果如下（Linux）：

```
Testing Time: 9.46s
  Unsupported    :  440
  Passed         : 2251
  Expectedly Failed:    1
```

## 2.6 crt 单元测试

按以下步骤进行 CRT 单元测试：

```
# run lld unit test with ninja
ninja check-crt
```

结果如下（Linux）：

```
[0/1] Running the CRT tests

Testing Time: 0.20s
  Passed: 2
```

## 2.7 Builtins 单元测试

按以下步骤进行 Builtins 单元测试：

```
# run lld unit test with ninja
ninja check-builtins
```

结果如下（Linux）：

```
[0/1] Running the Builtins tests

Testing Time: 2.35s
  Unsupported     : 59
  Passed          : 154
  Expectedly Failed:   1
```

## 2.8 check-llvm-tools 单元测试

按以下步骤进行 check-llvm-tools 单元测试：

```
# run lld unit test with ninja
ninja check-llvm-tools
```

结果如下（Linux）：

```
[0/1] Running lit suite /data2/wendong/aclang_toolchain/clang-toolchain/clang-15.0.3/out/llvm-project/llvm/test/tools

Testing Time: 6.62s
  Unsupported     : 301
  Passed          : 3211
```

```
  Expectedly Failed:  5
```

## 2.9 thinlto 单元测试

按以下步骤进行 thinlto 单元测试：

```
# run lld unit test with ninja
ninja check-llvm-thinlto
```

结果如下（Linux）：

```
[0/1] Running lit suite /data2/wendong/aclang_toolchain/clang-toolchain/clang-15.0.3/out/llvm-project/llvm/test/ThinLTO

Testing Time: 4.00s
  Unsupported:  7
  Passed     : 132
```

## 2.10 Clang_c 单元测试

按以下步骤进行 Clang_c 单元测试：

```
# run lld unit test with ninja
ninja check-clang-c
```

结果如下（Linux）：

```
[0/1] Running lit suite /data2/wendong/aclang_toolchain/clang-toolchain/clang-15.0.3/out/llvm-project/clang/test/C

Testing Time: 0.24s
  Passed: 14
```

## 2.11 Check_cxx 单元测试

按以下步骤进行 Check_cxx 单元测试：

```
# run lld unit test with ninja
ninja check-cxx
```

结果如下（Linux）：

```
Testing Time: 505.39s
  Unsupported     :  304
  Passed          : 7282
  Expectedly Failed:  41
```

## 2.12 Clang_cxx 单元测试

按以下步骤进行 Clang_cxx 单元测试：

```
# run lld unit test with ninja
ninja check-clang-cxx
```

结果如下（Linux）：

```
Testing Time: 1.40s
  Passed         : 833
  Expectedly Failed:   1
```

## 2.13 cxxabi 单元测试

按以下步骤进行 Clang_cxx 单元测试：

```
# run lld unit test with ninja
ninja check-cxxabi
```

结果如下（Linux）：

```
Testing Time: 18.72s
  Unsupported: 15
  Passed     : 56
```

## 2.14 check_clang_tools 单元测试

按以下步骤进行 check_clang_tools 单元测试：

```
# run lld unit test with ninja
ninja check-clang-tools
```

结果如下（Linux）：

```
Testing Time: 42.61s
  Unsupported     :   7
  Passed          : 2506
  Expectedly Failed:   2
```

## 2.15 CFI 单元测试

按以下步骤进行 cfi 单元测试：

```
# run lld unit test with ninja
ninja check-cfi
```

结果如下（Linux）：

```
Failed Tests (4):
  cfi-devirt-lld-thinlto-x86_64 :: mfcall.cpp
  cfi-devirt-lld-x86_64 :: mfcall.cpp
  cfi-standalone-lld-thinlto-x86_64 :: mfcall.cpp
  cfi-standalone-lld-x86_64 :: mfcall.cpp


Testing Time: 1.82s
  Unsupported      : 208
  Passed           : 40
  Expectedly Failed:   4
  Failed           :   4
```

这个测试在原始的 Clang（龙芯版本）也同样存在

```
Failed Tests (4):
  cfi-devirt-lld-thinlto-x86_64 :: mfcall.cpp
  cfi-devirt-lld-x86_64 :: mfcall.cpp
  cfi-standalone-lld-thinlto-x86_64 :: mfcall.cpp
  cfi-standalone-lld-x86_64 :: mfcall.cpp


Testing Time: 2.24s
  Unsupported      : 208
  Passed           : 40
  Expectedly Failed:   4
  Failed           :   4
```

这个 FAILED 会影响 AOSP 的 Java 中 137 号测试。具体内容请参考 3.2.3 节的内容。

# 3 AOSP 基础模块测试

本章描述熵核移植的 AOSP 基础模块的测试。suo'you 所有的测试都在烧录 AOSP 固件后，通过 adb 命令进行的测试。

## 3.1 bionic 测试

从 1.2 节的内容可以看到，这部分移植有龙芯负责的，熵核帮忙做验证以及部分修订（从合同中可以看到，熵核字本模块的工作量应该为 0 的）。这个的详细说明请参考文件：” 02-bionic .md “

编译测试用例：

```
## 编译测试用例
$ . art/xc_tools/bionic_g_b.sh


## 测试
$ . art/xc_tools/bionic_g_r.sh
```

测试结果如下（请参考测试结果文件：02-bionic-20241020.txt）：

```
##  正确运行的测试用例如下，总共 6274 个：
[  PASSED  ] 3100 tests.
[  PASSED  ] 2924 tests.
[  PASSED  ] 34 tests.
[  PASSED  ] 132 tests.
[  PASSED  ] 8 tests.
[  PASSED  ] 10 tests.
[  PASSED  ] 9 tests.
[  PASSED  ] 1 test.
[  PASSED  ] 56 tests.


## 错误的测试用例如下（总共 4 个）：
[  FAILED  ] 4 tests, listed below:
[  FAILED  ] fenv.feenableexcept_fegetexcept
[  FAILED  ] sys_ptrace.watchpoint_stress
[  FAILED  ] sys_ptrace.watchpoint_imprecise
[  FAILED  ] sys_ptrace.hardware_breakpoint

14 SLOW TESTS
 4 FAILED TESTS
  YOU HAVE 2 DISABLED TESTS
```

上述错误测试 kernel 的移植有关，会影响部分情况下的调试，但是不影响正常情况下的使用。

## 3.2 ART 测试

art 移植是本次移植的最重要内容，但是由于本次移植仅仅覆盖了解释执行模式，其他模式对应的测试用例则会 fail。

### 3.2.1 ART host gtest 测试

编译测试用例：

```
## 编译运行测试用例
$ art/test.py --host -g --64
```

测试结果如下（请参考测试结果文件：03-art-gtest-host-20241020.txt）：

```
## 正确运行的测试用例如下，总共 3047 个：
[ PASSED  ] 19 tests.
[ PASSED  ] 1 test.
[ PASSED  ] 5 tests.
[  PASSED  ] 5 tests.
[  PASSED  ] 5 tests.
[  PASSED  ] 1 test.
[  PASSED  ] 218 tests.
[  PASSED  ] 2 tests.
[  PASSED  ] 8 tests.
[  PASSED  ] 6 tests.
[  PASSED  ] 6 tests.
[  PASSED  ] 618 tests.
[  PASSED  ] 53 tests.
[  PASSED  ] 106 tests.
[  PASSED  ] 894 tests.
[  PASSED  ] 37 tests.
[  PASSED  ] 64 tests.
[  PASSED  ] 17 tests.
[  PASSED  ] 3 tests.
[  PASSED  ] 33 tests.
[  PASSED  ] 26 tests.
[  PASSED  ] 17 tests.
[  PASSED  ] 699 tests.
[  PASSED  ] 21 tests.
[  PASSED  ] 183 tests.

## 错误的测试用例如下（总共 7 个）：
[  FAILED  ] 5 tests, listed below:
[  FAILED  ] ArmVIXLAssemblerTest.VixlJniHelpers
[  FAILED  ] ArmVIXLAssemblerTest.VixlLoadFromOffset
[  FAILED  ] ArmVIXLAssemblerTest.VixlStoreToOffset
[  FAILED  ] AssemblerX86_64Test.Movss
[  FAILED  ] AssemblerX86_64Test.Movsd

 5 FAILED TESTS

------
[  FAILED  ] 2 tests, listed below:
[  FAILED  ] DwarfTest.DebugFrame
[  FAILED  ] DwarfTest.x86_64_RegisterMapping

 2 FAILED TESTS
  YOU HAVE 1 DISABLED TEST
```

上述错误测试都是由于采用了 Clang15 从而造成对比的汇编格式有变化造成的（AOSP 原始附带的 Clang 版本是 12）。

### 3.2.2 ART device gtest 测试

编译测试用例：

```
## 编译运行测试用例
$ art/tools/run-gtests.sh -j4
```

测试结果如下（请参考测试结果文件：04-art-gtest-device-20241020.txt）：

| testname | total | passed | failed |
|---|---|---|---|

| | | | |
|---|---|---|---|
| art_cmdline_tests | 19 | 19 | 0 |
| art_compiler_tests | 831 | 701 | 130 |
| art_dex2oat_tests | 114 | 109 | 5 |
| art_dexanalyze_tests | 5 | 5 | 0 |
| art_dexdiag_tests | 4 | 4 | 0 |
| art_dexdump_tests | 5 | 5 | 0 |
| art_dexlayout_tests | 17 | 17 | 0 |
| art_dexlist_tests | 5 | 5 | 0 |
| art_dexoptanalyzer_tests | 17 | 17 | 0 |
| art_imgdiag_tests | 2 | 2 | 0 |
| art_libartbase_tests | 219 | 219 | 0 |
| art_libartpalette_tests | 2 | 2 | 0 |
| art_libdexfile_support_tests | 6 | 6 | 0 |
| art_libdexfile_tests | 106 | 106 | 0 |
| art_libprofile_tests | 64 | 64 | 0 |
| art_oatdump_tests | 21 | 12 | 9 |
| art_odrefresh_tests | 33 | 33 | 0 |
| art_profman_tests | 37 | 37 | 0 |
| art_runtime_compiler_tests | 26 | 25 | 1 |
| art_runtime_tests | 703 | 697 | 6 |
| art_sigchain_tests | 14 | 13 | 1 |

上述错误测试与本次移植的编译器、运行时不支持 JIT、AOT 有关。比如运行时错的 6 个测试用例：

```
## yun'xing'sh 运行时错误的测试用例
 [==========] 703 tests from 95 test suites ran. (446765 ms total)
[  PASSED  ] 697 tests.
[  FAILED  ] 6 tests, listed below:
[  FAILED  ] OatFileAssistantTest.RaceToGenerate
[  FAILED  ] OatFileAssistantTest.GetDexLocation
[  FAILED  ] OatFileAssistantTest.SystemFrameworkDir
[  FAILED  ] OatFileAssistantTest.LoadOatNoArt
[  FAILED  ] TwoRuntimesTest.FirstInvocation
[  FAILED  ] TwoRuntimesTest.SecondInvocation

 6 FAILED TESTS
  YOU HAVE 1 DISABLED TEST
```

这些都是在调用 AOT 代码或者从在汇编代码执行过程中才用到的功能函数。

### 3.2.3 ART device java 测试

编译测试用例：

```
## 编译测试用例
$ art/tools/buildbot-build.sh --target

## 同步文件到设备
$ . push.sh

## 测试
$ art/test.py -j 4 --target -r --64 --ndebug  --interpreter -v
```

测试结果如下（请参考测试结果文件：05-art-java-device-20241020.txt）：

```
## 正确运行的测试用例如下，总共 889 个：


## 错误的测试用例如下（总共 3 个）：
test-art-target-run-test-ndebug-prebuild-interpreter-no-relocate-ntrace-cms-checkjni-picimage-ndebuggable-no-jvmti-cdex-fast-137-cfi64
test-art-target-run-test-ndebug-prebuild-interpreter-no-relocate-ntrace-cms-checkjni-picimage-ndebuggable-no-jvmti-cdex-fast-988-method-trace64
test-art-target-run-test-ndebug-prebuild-interpreter-no-relocate-ntrace-cms-checkjni-picimage-ndebuggable-no-jvmti-cdex-fast-989-method-trace-throw64
```

上述错误测试与 unwind 功能有关，但是不影响正常情（unwind 模块有龙芯负责，熵核在初期帮忙进行必要代码移植 - 为了便于编译、初始阶段的功能调试）。另外，137 号测试用例与工具链有关，如 2.15 节描述。

这三个错误的测试用例不影响正常功能，只是会在调试 Art 功能的时候会用到。

## 3.3 system 测试

system 下的部分与 Arch 相关的代码移植是本次合同的一个重要部分。主要涉及到如下模块。

### 3.3.1 system-core 测试

编译测试用例：

```
## 编译测试用例
$ . art/xc_tools/system_core_g_b.sh

## 测试
$ . art/xc_tools/system_core_g_r.sh
```

测试结果如下（请参考测试结果文件：06-system-core-20241020.txt）：

```
## 正确运行的测试用例如下，总共 444 个：
[ PASSED ] 75 tests.
[ PASSED ] 75 tests.
[ PASSED ] 75 tests.
[ PASSED ] 8 tests.
[ PASSED ] 1 test.
[ PASSED ] 5 tests.
[ PASSED ] 12 tests.
[ PASSED ] 156 tests.
[ PASSED ] 1 test.
[ PASSED ] 4 tests.
[ PASSED ] 1 test.
[ PASSED ] 26 tests.
[ PASSED ] 3 tests.
[ PASSED ] 2 tests.
system-base
## 错误的测试用例如下（总共 1 个）：
```

```
[  FAILED  ] 1 test, listed below:
[  FAILED  ] SchedPolicy.set_sched_policy

 1 FAILED TEST
```

上述错误测试与当前使用的 Kernel 有关，函数本是直接调用了 Kernel 的实现(此模块无 Arch 相关代码)。

### 3.3.2 system-libbase 测试

编译测试用例：

```
## 编译测试用例
$ .  art/xc_tools/system_libbase_g_b.sh

## 测试
$ .  art/xc_tools/system_libbase_g_r.sh
```

测试结果如下（请参考测试结果文件：07-system-libbase-20241020.txt）：

```
##  正确运行的测试用例如下，总共 302 个：
[==========] 303 tests from 23 test suites ran. (3957 ms total)
[  PASSED  ] 302 tests.
[  SKIPPED ] 1 test, listed below:
[  SKIPPED ] properties.too_long
```

### 3.3.3 system-unwind 测试

unwinding 是有龙芯负责的模块，但是前期熵核为了尽快编译此模块用于 Art 移植调试，就依据 RISC-V 移植了部分代码，后续的修订有龙芯负责的。

编译测试用例：

```
## 编译测试用例
$ .  art/xc_tools/system_unwinding_g_b.sh

## 测试
$ .  art/xc_tools/system_unwinding_g_r.sh
```

测试结果如下（请参考测试结果文件：05-art-java-device-20241020.txt）：

```
##  正确运行的测试用例如下，总共 26 个，错误的测试用例 10 个：
[==========] 37 tests from 1 test suite ran. (21021 ms total)
[  PASSED  ] 26 tests.
[  TIMEOUT ] 1 test, listed below:
[  TIMEOUT ] BacktraceTest.ptrace_threads (stopped at 15001 ms)
[  FAILED  ] 10 tests, listed below:
[  FAILED  ] BacktraceTest.ptrace_trace
[  FAILED  ] BacktraceTest.ptrace_max_trace
[  FAILED  ] BacktraceTest.ptrace_ignore_frames
[  FAILED  ] BacktraceTest.check_unreadable_elf_remote
[  FAILED  ] BacktraceTest.unwind_through_unreadable_elf_remote
[  FAILED  ] BacktraceTest.remote_get_function_name_before_unwind
[  FAILED  ] BacktraceTest.unwind_disallow_device_map_remote
```

```
[  FAILED  ] BacktraceTest.unwind_remote_through_signal_using_handler
[  FAILED  ] BacktraceTest.unwind_remote_through_signal_using_action
[  FAILED  ] BacktraceTest.check_for_leak_remote

 1 TIMEOUT TEST
10 FAILED TESTS
```

上述错误测试在一定程度上会影响 Java 测试中 trace 测试功能，但是不影响正常使用。

### 3.3.4 system-other 测试

编译测试用例：

```
## 编译测试用例
$ . art/xc_tools/system_others_g_b.sh

## 测试
$ . art/xc_tools/system_others_g_r.sh
```

测试结果如下（请参考测试结果文件：05-art-java-device-20241020.txt）：

```
##  正确运行的测试用例如下，总共 228 个：
[  PASSED  ] 0 tests.
[  PASSED  ] 1 test.
[  PASSED  ] 56 tests.
[  PASSED  ] 66 tests.
[  PASSED  ] 1 test.
[  PASSED  ] 37 tests.
[  PASSED  ] 9 tests.
[  PASSED  ] 56 tests.
[  PASSED  ] 1 test.
[  PASSED  ] 1 test.
```

## 3.4 framework 测试

art 移植是本次移植的最重要内容，但是由于本次移植仅仅覆盖了解释执行模式，其他模式对应的测试用例则会 fail。

编译测试用例：

```
## 编译测试用例
$ . art/xc_tools/system_core_g_b.sh

## 测试
$ . art/xc_tools/system_core_g_r.sh
```

测试结果如下（请参考测试结果文件：02-bionic-20241020.txt）：

```
##  正确运行的测试用例如下，总共 203 个：
```

```
[  PASSED  ] 15 tests.
[  PASSED  ] 188 tests.

## 错误的测试用例如下（总共 3 个）：
[  FAILED  ] 3 tests, listed below:
[  FAILED  ] BackupHelpersTest.WriteTarFileWithSizeGreaterThan2GB
[  FAILED  ] PosixUtilsTest.AbsolutePathToBinary
[  FAILED  ] PosixUtilsTest.RelativePathToBinary
```

上述第一个错误与大小超过 2G 有关，在我们当时的配置下不支持；另外的两个错误是测试用例错误（在’ date --help ‘命令中寻找字符串 usage: date，且希望是打印的帮助信息的最开始，但是实际不是，如下图所示）。

```
[^_^aosp.la]$ adb shell
loongson_3a5000:/ # date --help
Toybox 0.8.4-android multicall binary: https://landley.net/toybox (see toybox --help)

usage: date [-u] [-I RES] [-r FILE] [-d DATE] [+DISPLAY_FORMAT] [-D SET_FORMAT] [SET]

Set/get the current date/time. With no SET shows the current date.

-d      Show DATE instead of current time (convert date format)
-D      +FORMAT for SET or -d (instead of MMDDhhmm[[CC]YY][.ss])
-I RES  ISO 8601 with RESolution d=date/h=hours/m=minutes/s=seconds/n=ns
-r      Use modification time of FILE instead of current date
-u      Use UTC instead of current timezone
```