Redemption Unit Tests

Jonathan Poelen

Wallix

17 septembre, 2019

Table of Contents

Table of Contents

Hello World

```cpp
#include "test_only/test_framework/redemption_unit_tests.hpp"

AUTO_TEST_CASE(TestRect)
{
    Rect r(10, 110, 10, 10);

    TEST(r.x == 10);
    TEST(r.y == 10);
}
```

tests/utils/test_rect.cpp(8): **error** in "TestRect": check r.y == 10 has failed (110 != 10)

***** 1 failure is detected in the test module "./tests/utils/test_rect.cpp"**

Objectifs

Avoir un maximum d'information sur l'erreur

Objectifs

Avoir un maximum d'information sur l'erreur

- Les lignes en cause

Objectifs

Avoir un maximum d'information sur l'erreur

- Les lignes en cause
- Les conditions d'erreur

Objectifs

Avoir un maximum d'information sur l'erreur

- Les lignes en cause
- Les conditions d'erreur
- Les valeurs causant une erreur

Objectifs

Avoir un maximum d'information sur l'erreur

- Les lignes en cause
- Les conditions d'erreur
- Les valeurs causant une erreur

```
1 bool result = x == 10;
2 TEST(result); // pas bon
```

tests/utils/test.cpp(42): **error** in "TestRect": check result has failed

Objectifs

Avoir un maximum d'information sur l'erreur

- Les lignes en cause
- Les conditions d'erreur
- Les valeurs causant une erreur

```
1 bool result = x == 10;
2 TEST(result); // pas bon
```

tests/utils/test.cpp(42): **error** in "TestRect": check result has failed

```
1 TEST(x == 10); // bon
```

tests/utils/test.cpp(42): **error** in "TestRect": check x == 10 has failed (110 != 10)

Un test sans `TEST` n'est pas un test !

Pas de if

- Un bon test limite les `if`.

Pas de if

- Un bon test limite les `if`.
- En dehors des boucles, **0 `if`**, car le résultat de la condition **est** connue à l'avance.

Pas de if

- Un bon test limite les `if`.
- En dehors des boucles, **0 `if`**, car le résultat de la condition **est** connue à l'avance.

```
1 if (x == 10) { // pas bon
2    ....
3 }
```

Pas de if

- Un bon test limite les `if`.
- En dehors des boucles, **0 `if`**, car le résultat de la condition **est** connue à l'avance.

```
1 REQUIRE(x == 10); // bon
2 ....
```

Notre framework

- Basé sur **Boost.Test**

Notre framework

- Basé sur **Boost.Test**
- Utilise le préfixe `RED` plutôt que `BOOST`

Notre framework

- Basé sur **Boost.Test**
- Utilise le préfixe `RED` plutôt que `BOOST`
- Ajoute des macros de test spécialisées

3 niveaux de sévérité

- `WARN`: simple avertissement

3 niveaux de sévérité

- `WARN`: simple avertissement
- `CHECK`/`TEST`: incrémente le compteur d'erreur et continue

3 niveaux de sévérité

- `WARN`: simple avertissement
- `CHECK`/`TEST`: incrémente le compteur d'erreur et continue
- `REQUIRE`: arrêt immédiat

Macro de comparaison de valeur

- `TEST(expr)` ou `TEST_<level>(expr)`

```
1 TEST(a == b); // CHECK((a) == (b))
2 TEST(a < b); // CHECK(a < b)
3 TEST(a == b +- 2_v); // b - 2 <= a && a <= b + 2
4 TEST(a < b +- 2_percent); // a < b + b * 2 / 100
```

tests/utils/test.cpp(8): **error** in "Test": check a < b +- 2_v has failed (7 >= 5+-2 (3, 7)

Macro de comparaison de valeur

- `TEST(expr)` ou `TEST_<level>(expr)`
- `<level>_CLOSE[_FRACTION](a, b, tolerance)`

```
1 // seulement pour nombre flottant
2 CHECK_CLOSE(v1, v2, 0.0001); // 0.0001%
3 CHECK_CLOSE_FRACTION(v1, v2, 0.0008999);
```

Macro de comparaison de valeur

- `TEST(expr)` ou `TEST_<level>(expr)`
- `<level>_CLOSE[_FRACTION](a, b, tolerance)`
- `<level>_MESSAGE(bool_expr, ostream_expr)}`

```
1 CHECK_MESSAGE(a[i], "a[" << i << "] = " << a[i]);
```

tests/utils/test.cpp(8): **error** in "Test": a(1)

Macro de comparaison de valeur

- `TEST(expr)` ou `TEST_<level>(expr)`
- `<level>_CLOSE[_FRACTION](a, b, tolerance)`
- `<level>_MESSAGE(bool_expr, ostream_expr)}`
- `<level>_{EQ, NE, LE, LT, GE, GT}(a, b)`

```
1 CHECK_EQ(a, b); // CHECK(a == b)
```

Macro de comparaison de valeur

- `TEST(expr)` ou `TEST_<level>(expr)`
- `<level>_CLOSE[_FRACTION](a, b, tolerance)`
- `<level>_MESSAGE(bool_expr, ostream_expr)}`
- `<level>_{EQ, NE, LE, LT, GE, GT}(a, b)`
- `<level>_FILE_CONTENTS(filename, contents)`

```
1 #include "test_only/get_file_contents.hpp"
2 CHECK_FILE_CONTENTS("recorder-chunk.pgs",
3     R"js({"percentage":100,"eta":0,"videos":1})js");
```

tests/utils/test.cpp(8): **error** in "Test": check contents__ == get_file_contents(filename__) has failed (100 0 != {"percentage":100,"eta":0,"videos":1})
Failure occurred in a following context:
filename: recorder-chunk.pgs

Error: invalid operands to binary

```
1  struct A
2  {
3      bool operator == (A const&) const
4      {
5          return true;
6      }
7  };
8
9  AUTO_TEST_CASE(TestA)
0  {
1      // invalid operands to binary expression
2      // ('const A' and 'const A')
3      TEST(A{} == A{});
4  }
```

Les valeurs doivent pouvoir être affichées, on peut:

- Tricher

```
1 TEST((A{} == A{}));
```

Les valeurs doivent pouvoir être affichées, on peut:

- Tricher
- Fournir une surcharge de `std::ostream`

```
1 std::ostream& operator<<(std::ostream& out, A const&)
2 {
3     return out << "A";
4 }
```

Les valeurs doivent pouvoir être affichées, on peut:

- Tricher
- Fournir une surcharge de `std::ostream`
- Spécialiser une classe `TEST_PRINT_TYPE_STRUCT_NAME`

```
1 template<> struct TEST_PRINT_TYPE_STRUCT_NAME<A>
2 {
3     void operator()(std::ostream& out, A const& /*x*/)
4     {
5         return out << "A";
6     }
7 };
```

Les valeurs doivent pouvoir être affichées, on peut:

- Tricher
- Fournir une surcharge de `std::ostream`
- Spécialiser une classe `TEST_PRINT_TYPE_STRUCT_NAME`
- Utiliser `TEST_DELEGATE_PRINT`

```
1 TEST_DELEGATE_PRINT(A, "A");
2
3 TEST_DELEGATE_PRINT(ns::YYY,
4     "YYY{" << int(x) << "}");
5
6 TEST_DELEGATE_PRINT_ENUM(ns::YYY);
```

Les valeurs doivent pouvoir être affichées, on peut:

- Tricher
- Fournir une surcharge de `std::ostream`
- Spécialiser une classe `TEST_PRINT_TYPE_STRUCT_NAME`
- Utiliser `TEST_DELEGATE_PRINT`
- Ne pas afficher avec `TEST_DONT_PRINT_LOG_VALUE(A)`

Macro de comparaison de séquence

- `<level>_EQ_COLLECTIONS(first1, last1, first2, last2)`

Macro de comparaison de séquence

- `<level>_EQ_COLLECTIONS(first1, last1, first2, last2)`
- `<level>_EQ_RANGES(rng1, rng2)`

```
1 CHECK_EQ_COLLECTIONS(rng1, rng2);
```

tests/utils/test.cpp(8): **error** in "Test": check ::boost::test_tools::tt_detail::make_it_pair((void("rng1"), begin(a__)), end(a__)) == ::boost::test_tools::tt_detail::make_it_pair((void("rng2"), begin(b__)), end(b__)) has failed
Mismatch at position 0: 1 != 0.
Mismatch at position 2: 3 != 2.

Macro de comparaison de séquence

- `<level>_EQ_COLLECTIONS(first1, last1, first2, last2)`
- `<level>_EQ_RANGES(rng1, rng2)`
- `<level>_[SBHC]MEM(bytes1, bytes2)`

```
1 CHECK_MEM(rng, "viking"_av);
```

tests/utils/test.cpp(8): **error** in "Test": check rng1__.size() == rng2__.size() has failed (4 != 6)
Failure occurred in a following context:
(rng) == ("viking"_av)
tests/utils/test.cpp(8): **error** in "Test": check rng1__ == rng2__ has failed ("\x76\x01\x02\x03" != "\x76"\x69\x6b\x69\x6e\x67")
Failure occurred in a following context:
(rng) == ("viking"_av)

Macro de comparaison de séquence

- `<level>_EQ_COLLECTIONS(first1, last1, first2, last2)`
- `<level>_EQ_RANGES(rng1, rng2)`
- `<level>_[SBHC]MEM(bytes1, bytes2)`

```
1 CHECK_SMEM(rng, "viking"_av);
```

tests/utils/test.cpp(8): **error** in "Test": check rng1__.size() == rng2__.size() has failed (4 != 6)
Failure occurred in a following context:
(rng) == ("viking"_av)
tests/utils/test.cpp(8): **error** in "Test": check rng1__ == rng2__ has failed (v\x01\x02\x03" != viking)
Failure occurred in a following context:
(rng) == ("viking"_av)

Macro de comparaison de séquence

- <level>_EQ_COLLECTIONS(first1, last1, first2, last2)
- <level>_EQ_RANGES(rng1, rng2)
- <level>_[SBHC]MEM(bytes1, bytes2)

```
1 CHECK_BMEM(rng, "viking"_av); // byte (hexa)
2 CHECK_HMEM(rng, "viking"_av); // dump format
3 CHECK_CMEM(rng, "viking"_av); // ascii format
```

Macro pour les appels de fonction

- `<level>_FUNC(function)((args...) @ xxx)`

```
1 TEST_FUNC(strlen)((str) == len);
```

tests/utils/test.cpp(8): **error** in "Test": check fctx__8(str) == len has failed (strlen(plop)= 4 != 3)

Macro pour les appels de fonction

- <level>_FUNC(function)((args...) @ xxx)
- <level>_FUNC_CTX(function)

```
1 // globale ou local
2 auto fsize = TEST_FUNC_CTX(filesize);
3 TEST(fsize(filename) == size);
```

Macro pour les appels de fonction

- <level>_FUNC(function)((args...) @ xxx)
- <level>_FUNC_CTX(function)
- <level>_INVOKER(function)

```
1 // local
2 auto foo = TEST_INVOKER(obj.foo);
3 TEST(foo(x) == y);
```

Vérification d'exception

- `<level>_NO_THROW(stmt)`

```
1 CHECK_NO_THROW(something());
```

tests/utils/test.cpp(8): **error** in "Test": exception thrown by something()

Vérification d'exception

- `<level>_NO_THROW(stmt)`
- `<level>_THROW(stmt, exception)`

```
1 CHECK_THROW(something(), std::runtime_error);
```

tests/utils/test.cpp(8): **error** in "Test": exception std::runtime_error is expected

Vérification d'exception

- `<level>_NO_THROW(stmt)`
- `<level>_THROW(stmt, exception)`
- `<level>_EXCEPTION(stmt, exception, predicate)`

```
1 CHECK_EXCEPTION(something(), std::runtime_error,
2     [](auto& e) {
3         return e.what() == "Bibidi Bou"; });
```

tests/utils/test.cpp(8): **error** in "Test": exception std::runtime_error is expected

Vérification d'exception

- `<level>_NO_THROW(stmt)`
- `<level>_THROW(stmt, exception)`
- `<level>_EXCEPTION(stmt, exception, predicate)`
- `<level>_EXCEPTION_ERROR_ID(stmt, id)`

```
1 CHECK_EXCEPTION_ERROR_ID(
2     trans.send("message"_av),
3     ERR_TRANSPORT_WRITE_FAILED);
```

tests/utils/test.cpp(8): **error** in "Test": check e__.id == ERR_TRANSPORT_WRITE_FAILED has failed (10 != 1502)

Autre macros

- `TEST_CONTEXT(ostream_expr) { ... }`

```
1  for (auto&& s : {"a", "b", "c"})
2  {
3      TEST_CONTEXT(s)
4      {
5          std::string x = s;
6          x += x;
7          TEST(size(x) == 2);
8          x.pop_back();
9          TEST(size(x) == 1);
10     }
11 }
```

Autre macros

- `TEST_CONTEXT(ostream_expr) { ... }`
- `TEST_CHECKPOINT(ostream_expr)`

Autre macros

- `TEST_CONTEXT(ostream_expr) { ... }`
- `TEST_CHECKPOINT(ostream_expr)`
- `TEST_PASSPOINT()`

Autre macros

- `TEST_CONTEXT(ostream_expr) { ... }`
- `TEST_CHECKPOINT(ostream_expr)`
- `TEST_PASSPOINT()`
- `FAIL(ostream_expr)`

Autre macros

- `TEST_CONTEXT(ostream_expr) { ... }`
- `TEST_CHECKPOINT(ostream_expr)`
- `TEST_PASSPOINT()`
- `FAIL(ostream_expr)`
- `ERROR(ostream_expr)`

Autre macros

- `TEST_CONTEXT(ostream_expr) { ... }`
- `TEST_CHECKPOINT(ostream_expr)`
- `TEST_PASSPOINT()`
- `FAIL(ostream_expr)`
- `ERROR(ostream_expr)`
- `ERROR_COUNT`

```
1 for (size_t i; i < len && ERROR_COUNT == 0; ++i)
2 {
3     TEST(R[i] == refR[i]);
4     TEST(G[i] == refG[i]);
5     TEST(B[i] == refB[i]);
6 }
```

Test case

- `AUTO_TEST_CASE(test_name) { ... }`

```
1 AUTO_TEST_CASE(TestFoo)
2 {
3     int x = 1;
4     TEST(x == 1);
5 }
```

Test case

- `AUTO_TEST_CASE(test_name) { ... }`
- `DATA_TEST_CASE(test_name, dataset, varname) { ... }`

```
1  DATA_TEST_CASE(TestBitmap, (std::array{
2      FIXTURES_PATH "/color_image.bmp",
3      FIXTURES_PATH "/logo-redemption.bmp"
4      FIXTURES_PATH "/red_box.png",
5      FIXTURES_PATH "/wablogoblue_220x76.png",
6      FIXTURES_PATH "/red_box_20x20.png"
7  }), filename)
8  {
9      TEST(bitmap_from_file(filename).is_valid());
0  }
```

Test case

- `AUTO_TEST_CASE(test_name) { ... }`
- `DATA_TEST_CASE(test_name, dataset, varname) { ... }`
- `BIND_DATA_TEST_CASE(test_name, dataset, varnames...)`
  `↪ { ... }`

```
1 BIND_DATA_TEST_CASE(TestRdpLogonInfo, (std::array{
2     std::tuple{3, "3"},
3     std::tuple{5, "5"}
4 }), number, string)
5 {
6     TEST(std::to_string(number) == string);
7 }
```

test_remoteFx

```
1  struct CustomGdi : public gdi::NullGraphic
2  {
3      void draw(RDPSurfaceContent const &content)
4      {
5          testPassed = true;
6          testResult = fuzzyCompareImage(content.data);
7      }
8
9      bool testPassed = false;
0      bool testResult = false;
1  };
2
3  decoder.recv(tilesetStream, cmd, gdi);
4  TEST(gdi.testPassed);
5  TEST(gdi.testResult);
```

tests/utils/test.cpp(42): **error** in "TestRemoteFx": check

test_remoteFx

```
1  bool fuzzyCompare(uint8_t b1, uint8_t b2)
2  {
3      return (b1 > b2) ? b1 - b2 : b2 - b1;
4  }
5
6  bool fuzzyCompareImage(const uint8_t * img)
7  {
8      for(uint32_t pixel : refImage) {
9          if (fuzzyCompare(*img++, (pixel >> 16) & 0xff) >
       ↪ 1)
0              || fuzzyCompare(*img++, (pixel >> 8 ) & 0xff) >
       ↪ 1)
1              || fuzzyCompare(*img++, (pixel >> 0 ) & 0xff) >
       ↪ 1)
2          return false;
3      }
4    return true;
5  }
```

test_remoteFx

```
1 void checkCompareImage(const uint8_t *img)
2 {
3     for(uint32_t pixel : refImage) {
4         RED_TEST(*img++ == ((pixel >> 16) & 0xff) +- 1_v)
      ↪ ;
5         RED_TEST(*img++ == ((pixel >> 8 ) & 0xff) +- 1_v)
      ↪ ;
6         RED_TEST(*img++ == ((pixel >> 0 ) & 0xff) +- 1_v)
      ↪ ;
7         if (RED_ERROR_COUNT > 0) return;
8     }
9 }
```

test_remoteFx

```
1  struct CustomGdi : public gdi::NullGraphic
2  {
3      void draw(RDPSurfaceContent const & content)
4      {
5          testPassed = true;
6          checkCompareImage(content.data);
7      }
8
9      bool testPassed = false;
0  };
1
2  decoder.recv(tilesetStream, cmd, gdi);
3  TEST(gdi.testPassed);
```

test_remoteFx

tests/utils/test.cpp(42): **error** in "TestRemoteFx": check $R == ((*reflmage >> 16)\&0xFF) + -1\_vhas$ failed (0xde != 0x22+-0x1 (0x21, 0x23))
tests/utils/test.cpp(44): **error** in "TestRemoteFx": check $B == ((*reflmage >> 0)\&0xFF) + -1\_vhas$ failed (0x22 != 0xdf+-0x1 (0xde, 0xe0))

test_remoteFx

tests/utils/test.cpp(42): **error** in "TestRemoteFx": check gdi.testResult has failed

tests/utils/test.cpp(42): **error** in "TestRemoteFx": check $R == ((*reflmage >> 16)\&0xFF) + -1\_vhas$ failed (0xde != 0x22+-0x1 (0x21, 0x23))
tests/utils/test.cpp(44): **error** in "TestRemoteFx": check $B == ((*reflmage >> 0)\&0xFF) + -1\_vhas$ failed (0x22 != 0xdf+-0x1 (0xde, 0xe0))

Table of Contents

Les conflits de nom

- Plusieurs tests peuvent générer le même fichier: les tests se courcircuitent

Les conflits de nom

- Plusieurs tests peuvent générer le même fichier: les tests se courcircuitent
- La **compilation matricielle** génère les mêmes fichiers

```
1 bjam -j7 toolset=gcc-7 toolset=gcc-8 toolset=clang
    ↪ toolset=icc debug san release
```

Les conflits de nom

- Plusieurs tests peuvent générer le même fichier: les tests se courcircuitent
- La **compilation matricielle** génère les mêmes fichiers
- Plusieurs **dépôts** génèrent les mêmes fichiers (Jenkins)

Une maintenance plus complexe

- La génération de fichier supplémentaire n'est pas vérifiée

Une maintenance plus complexe

- La génération de fichier supplémentaire n'est pas vérifiée
- Il y a du code doublon pour supprimer les tests (avant et après le test)

Une maintenance plus complexe

- La génération de fichier supplémentaire n'est pas vérifiée
- Il y a du code doublon pour supprimer les tests (avant et après le test)
- Des fichiers peuvent subsister dans le dépôt suite à des échecs (`git status`)

WorkingDirectory

- Un espace de travail isolé par test
  (`$TMPDIR/TestName@WDName@ModuleName@compiler`)

WorkingDirectory

- Un espace de travail isolé par test (`$TMPDIR/TestName@WDName@ModuleName@compiler`)
- Ajout et suppression de fichier au fur et à mesure + test sur chacun

WorkingDirectory

- Un espace de travail isolé par test (`$TMPDIR/TestName@WDName@ModuleName@compiler`)
- Ajout et suppression de fichier au fur et à mesure + test sur chacun

```
1  AUTO_TEST_CASE(TestA)
2  {
3    WorkingDirectory wd{/*name*/};
4    TEST(wd.contents("file1") == str);
5    TEST(wd.add_file("file2"));
6    TEST(wd.add_files({"file2", "file3", "file4"}));
7    TEST(wd.remove_file("file_a"));
8    TEST(wd.remove_files({"file_b", "file_c"}));
9    RED_CHECK_WORKSPACE(wd);
0  }
```

WorkingDirectory

- Un espace de travail isolé par test (`$TMPDIR/TestName@WDName@ModuleName@compiler`)
- Ajout et suppression de fichier au fur et à mesure + test sur chacun

```
1 AUTO_TEST_CASE_WD(TestA, wd)
2 {
3   CHECK_FILE_CONTENTS(wd.add_file("file1"), str);
4   wd.add_file("file2");
5   wd.add_files({"file2", "file3", "file4"});
6   // ...
7   wd.remove_file("file_a"));
8   wd.remove_files({"file_b", "file_c"});
9 }
```

WorkingFile

```
1 AUTO_TEST_CASE(TestA)
2 {
3   WorkingFile wf{name};
4   CHECK_FILE_CONTENTS(wf, str);
5 }
```

```
1 AUTO_TEST_CASE_WF(TestA, wf)
2 {
3   CHECK_FILE_CONTENTS(wf, str);
4 }
```

Table of Contents

- Les tests sont lancés en même temps que la compilation

- Les tests sont lancés en même temps que la compilation
- L'ensemble des tests seuls se compilent avec `bjam tests`

- Les tests sont lancés en même temps que la compilation
- L'ensemble des tests seuls se compilent avec `bjam tests`
- Chaque dossier peut être compilé avec son chemin
  `bjam tests/mod/rdp`

- Les tests sont lancés en même temps que la compilation
- L'ensemble des tests seuls se compilent avec `bjam tests`
- Chaque dossier peut être compilé avec son chemin
  `bjam tests/mod/rdp`
- Il est possible de compiler un dossier sans ses sous-dossiers
  `bjam tests/mod.norec`

- Les tests sont lancés en même temps que la compilation
- L'ensemble des tests seuls se compilent avec `bjam tests`
- Chaque dossier peut être compilé avec son chemin
  `bjam tests/mod/rdp`
- Il est possible de compiler un dossier sans ses sous-dossiers
  `bjam tests/mod.norec`
- Ou spécifier un test directement
  `bjam tests/core/RDP/test_remotefx` ou
  `bjam test_remotefx`

- Les tests sont lancés en même temps que la compilation
- L'ensemble des tests seuls se compilent avec `bjam tests`
- Chaque dossier peut être compilé avec son chemin
  `bjam tests/mod/rdp`
- Il est possible de compiler un dossier sans ses sous-dossiers
  `bjam tests/mod.norec`
- Ou spécifier un test directement
  `bjam tests/core/RDP/test_remotefx` ou
  `bjam test_remotefx`
- Des scripts d'auto-complétion pour `zsh` et `bash` se trouvent dans `tools/bjam/bjam_completion.*`

Table of Contents

Bjam utilise les variables d'environnement ou l'option `-s`

- `export BOOST_STACKTRACE=1`

Bjam utilise les variables d'environnement ou l'option `-s`

- `export BOOST_STACKTRACE=1`
- `BOOST_STACKTRACE=1 bjam`

Bjam utilise les variables d'environnement ou l'option `-s`

- `export BOOST_STACKTRACE=1`
- `BOOST_STACKTRACE=1 bjam`
- `bjam -sBOOST_STACKTRACE=1`

- `BOOST_STACKTRACE=1` affiche la pile d'appel lorsque qu'un type `Error` est construit

- `BOOST_STACKTRACE=1` affiche la pile d'appel lorsque qu'un type `Error` est construit
- `NO_FFMPEG`, `FFMPEG_INC_PATH`, `FFMPEG_LIB_PATH`, `FFMPEG_LINK_MODE`

- `BOOST_STACKTRACE=1` affiche la pile d'appel lorsque qu'un type `Error` est construit
- `NO_FFMPEG`, `FFMPEG_INC_PATH`, `FFMPEG_LIB_PATH`, `FFMPEG_LINK_MODE`
- toutes les variables liées à l'installation: `INSTALLDIR`, `PREFIX`, etc

- `BOOST_STACKTRACE=1` affiche la pile d'appel lorsque qu'un type `Error` est construit
- `NO_FFMPEG`, `FFMPEG_INC_PATH`, `FFMPEG_LIB_PATH`, `FFMPEG_LINK_MODE`
- toutes les variables liées à l'installation: `INSTALLDIR`, `PREFIX`, etc

```
1    sed -E 's/.*\[ setvar ([^ ]+).*\] ;/\1/;t;d' jam/
   ↪ defines.jam
```

- `REDEMPTION_LOG_PRINT=1` permet d'activer ou non les logs dans les tests

- `REDEMPTION_LOG_PRINT=1` permet d'activer ou non les logs dans les tests
- `REDEMPTION_FILTER_ERROR=ERR_TRANSPORT_NO_MORE_DATA,` `↪ ERR_SEC` supprime la trace de certaines erreurs

Table of Contents

- tools/rdpproxy_color.awk

```
rdpproxy: INFO (11844/11844) -- ModuleManager::internal module Close ready
rdpproxy: INFO (11844/11844) -- Authentifier (-1): total_received=2785, total_sent=741
rdpproxy: ERR (11844/11844) -- ◢ In src/core/session.cpp:177
rdpproxy: ERR (11844/11844) -- Proxy data wait loop raised error 9 : Bad file descriptor
rdpproxy: WARNING (11844/11844) -- ◢ In src/transport/socket_transport.cpp:296
rdpproxy: WARNING (11844/11844) -- SocketTransport::Send failed on RDP Client (6) errno=9 [Bad file descriptor]
rdpproxy: DEBUG (11844/11844) -- Create Error: Exception ERR_TRANSPORT_WRITE_FAILED no: 1502
rdpproxy: DEBUG (11844/11844) -- #0  Error at src/core/error.cpp:109
rdpproxy: DEBUG (11844/11844) -- #1  SocketTransport::do_send(unsigned char const*, unsigned long) at src/transport/socket_transport.cpp:297
```

- tools/rdpproxy_color.awk
- tools/bjam/unit_test_color.awk

```
testing.unit-test redemption/clang-linux-7.0.0/debug/tests/utils/test_rect.passed
tests/utils/test_rect.cpp(36): error in "TestRect": check 11 == r.x has failed [11 != 10]

*** 1 failure is detected in the test module "./tests/utils/test_rect.cpp"

    LD_LIBRARY_PATH="/home/jpoelen/projects/build/redemption-public/redemption/clang-linux-7.0.0/debug:$LD_LIBRARY_PATH"
export LD_LIBRARY_PATH

    "/home/jpoelen/projects/build/redemption-public/redemption/clang-linux-7.0.0/debug/tests/utils/test_rect"  && touch  "/home
```

- tools/rdpproxy_color.awk
- tools/bjam/unit_test_color.awk
- tools/bjam/bjam_filter.awk

```
...found 5644 targets...
...updating 1239 targets...
[5%] [65/1239] clang-linux.compile.c++.without-pth redemption/clang-linux-7.0.0/debug/tests/mod/rdp/channels/test_sespro_clipboard_based_launcher.o
src/mod/rdp/channels/cliprdr_channel.hpp:76:29: warning: unused parameter 'channel_files_directory' [-Wunused-parameter]
        const std::string & channel_files_directory,
                            ^
src/mod/rdp/channels/cliprdr_channel.hpp:65:21: warning: private field 'session_reactor' is not used [-Wunused-private-field]
    SessionReactor& session_reactor;
                    ^
2 warnings generated.
[8%] [101/1239] clang-linux.compile.c++.without-pth redemption/clang-linux-7.0.0/debug/tests/mod/rdp/test_rdp.o
```

- tools/rdpproxy_color.awk
- tools/bjam/unit_test_color.awk
- tools/bjam/bjam_filter.awk
- tools/bjam/bjam_autocompletion.*

- tools/rdpproxy_color.awk
- tools/bjam/unit_test_color.awk
- tools/bjam/bjam_filter.awk
- tools/bjam/bjam_autocompletion.*
- tools/c++-analyzer/*