ASTROBRICK

Guide d'utilisation

Michel Pujol
23/03/2015

Table des matières

1. Présentation

Ce guide s'adresse aux développeurs de programmes qui souhaitent utiliser une *Astrobrick* dans leur programme.

Une *Astrobrick* est une librairie dynamique écrite en C++ pour Audela pouvant être utilisée par d'autres programmes écrits dans différents langages (C++, Tcl, Python, C#) et pouvant être exécutée sous Window, Linux et Macosx.

Chaque *ASTROBRICK* est fournie avec:

- un fichier binaire pour chaque système d'exploitation Windows, Linux et Macosx (.dll , .so , .dylib)
- un fichier header .h et un fichier .lib pour l'utilisation par un programme écrit en C++
- un exemple de fichier .cs pour l'utilisation par un programme écrit en C#
- un exemple de fichier .py pour l'utilisation par un programme écrit en Python 3
- une librairie complémentaire pour l'utilisation par un programme écrit en Tcl 8.6 (cette librairie complémentaire sert à convertir tous les paramètres des fonctions en chaînes de caractères, seul format accepté pat l'interpréteur Tcl)

2. Exemple absimple

L'astrobrick **absimple** contient les fonctions suivantes:

```
int absimple_processAdd(int a, int b);
int absimple_processSub(int a, int b);
```

Cette Astrobrick est fournie avec les fichiers:

```
/audela
   /astrobrick_user
      /csharp
        absimple.cs           // interface d'import C# de la brique
        absimple_test.cs      // exemple de programme d'utilisation de la brique
        libabsimple.dll       // librairie de la brique
        libabsimple.dylib     // librairie de la brique
        libabsimple.so        // librairie de la brique

      /doc
        /html                 // documentation des briques
        astrobrick.html       // point d'entrée de la documentation

      /python
        absimple.py           // interface d'import Python de la brique
        absimple_test.py      // exemple de programme d'utilisation de la brique
        libabsimple.dll       // librairie de la brique
        libabsimple.dylib           // librairie de la brique
        libabsimple.so        // librairie de la brique

   /bin
        libabsimple_tcl.dll   // interface d'import TCL de la brique
        libabsimple_tcl.dylib // interface d'import TCL de la brique
        libabsimple_tcl.so    // interface d'import TCL de la brique
        libabsimple.dll       // binaire de la brique
```

```
        libabsimple.dylib           // binaire de la brique
        libabsimple.so        // binaire de la brique

   /src
      /include
        absimple.h            // interface d'import C de la brique
                              // de la brique
      /lib
        absimple.lib          // interface d'import C++ par chargement statique
                              // de la brique dans un programme utilisateur C++
                              // Windows
```

3. Utilisation dans un programme C++

Exemple de source C++ utilisant absimple

```
// absimple_test.cpp

#include "absimple.h"

int main(int argc, char* argv[])
{
   int a = 8;
   int b = 4;

   int result = absimple_processAdd(a, b);
   cout << "absimple_processAdd: " << a << " + " << b << " = " << result <<
endl;
   // printf("absimple_processAdd: %d + %d = %d\n",a, b, result);

   int result2 = absimple_processSub(a, b);
   cout << "processAdd: " << a << " - " << b << " = " << result2 << endl;
   // printf(""absimple_processSub: %d + %d = %d\n",a, b, result2);
}
```

Le programme C++ importe le fichier **absimple.h** pour pourvoir utiliser les fonctions de absimple

3.1 Windows Visual C++

Compilation du projet absimple_test.vcproj

- ajouter le répertoire de **absimple.h** dans les paramètres du compilateur
- ajouter **absimple.lib** dans les paramètres du linker

Exécution

```
>absimple_test.exe
absimple_processAdd: 8 + 4 = 12
absimple_processSub: 8 - 4 = 4

>
```

3.2 Linux

Compilation :

- ajouter le répertoire de **absimple.h** dans les paramètres de compilation du programme (paramètre -I)

- ajouter **libabsimple.so** dans les paramètres du linker (paramètres -L et -l)

```
$ make
*** Compiling absimple_test.cpp
g++ -O2  -fPIC  -fno-stack-protector -c -Wall  -I../../absimple/src -o
absimple_test.o ../src/absimple_test.cpp
*** Linking library absimple_test
gcc absimple_test.o -ldl -lm -lstdc++ -Wl,-rpath,. -L../bin  -lsimple -o
absimple_test
$
```

Exécution

```
$ ../bin/absimple_test
absimple_processAdd: 8 + 4 = 12
absimple_processSub: 8 - 4 = 4

$
```

3.3 Macos X

Compilation :

- ajouter le répertoire de **absimple.h** dans les paramètres du compilateur (paramètre -I)
- ajouter **libabsimple.dylib** dans les paramètres du linker (paramètres -L et -l)

```
$ make
*** Compiling absimple_test.cpp
g++ -O2  -fPIC  -fno-stack-protector -c -Wall  -I../../absimple/src -o
absimple_test.o ../src/absimple_test.cpp
*** Linking library absimple_test
gcc absimple_test.o -ldl -lm -lstdc++ -Wl,-rpath,. -L../bin  -lsimple -o
../bin/absimple_test
$
```

Exécution

```
$ ../bin/absimple_test
absimple_processAdd: 8 + 4 = 12
absimple_processSub: 8 - 4 = 4

$
```

4. Utilisation dans un programme C#

Exemple de source utilisant absimple (projet audela-csharp.csproj)

```
// audela_csharp.cs

using System;
using System.Text;

namespace audela_csharp
{
    class absimple_test
    {
        static void Main(string[] args)
        {
```

```
            int a = 4;
            int b = 8;

            int result = absimple.absimple_processAdd(a, b);
            Console.WriteLine("absimple_processAdd: " + a + " + " + b + " = " +
result);

            int result2 = absimple.absimple_processSub(a, b);
            Console.WriteLine("absimple_processSub: " + a + " + " + b + " = " +
result2);

        }
    }
}
```

Le programme C# doit être compilé avec le fichier **absimple.cs** pour pourvoir utiliser les fonctions de absimple.

4.1 Windows Visual C#

Compilation du projet audela-csharp.csproj

- ajouter absimple.cs dans le projet
- compiler le projet ab_test.vcsproj
- copier **libabsimple.dll** dans le même répertoire que audela-csharp.exe

Exécution

```
> audela_csharp.exe
absimple_processAdd: 8 + 4 = 12
absimple_processSub: 8 - 4 = 4

>
```

4.2 Linux Mono C#

Compilation :

- compiler le programme audela-csharp.cs avec **absimple.cs**

```
$ cd audela-csharp/linux
$ make
```

Exécution

- lancer audela-csharp avec l'interpréteur mono

Résultat

```
$ mono audela_csharp
absimple_processAdd: 8 + 4 = 12
absimple_processSub: 8 - 4 = 4

$
```

4.3 Macosx Mono C#

Compilation :

- compiler le programme audela-csharp.cs avec **absimple.cs**

```
$ cd audela-csharp/macosx
$ make
```

Exécution

- copier **libabsimple.so** dans le même répertoire que audela-csharp
- lancer audela-csharp

Résultat

```
$ ./audela_csharp
absimple_processAdd: 8 + 4 = 12
absimple_processSub: 8 - 4 = 4

$
```

5. Utilisation dans un programme Python

Exemple de source Python utilisant absimple

```
# file: audela.py

import absimple

a = 8
b = 4
result = absimple.processAdd(a,b)
print ("absimple.processAdd : %d + %d = %d" % (a,b,result))

result2 = absimple.processSub(a,b)
print ("absimple.processAdd : %d - %d = %d" % (a,b,result2))
```

Le programme python **audela.py** importe le fichier **absimple.py** qui montre comment charger la librairie

5.1 Windows

Exécution

- lancer l'exécution de absimple_test.py

Résultat

```
> python.exe -i audela_test.py
absimple_processAdd: 8 + 4 = 12
absimple_processSub: 8 - 4 = 4

>>>
>
```

5.2 Linux

Exécution

- lancer l'exécution de absimple_test.py

Résultat

```
$ python3 absimple_test.py
absimple_processAdd: 8 + 4 = 12
absimple_processSub: 8 - 4 = 4

>>>
$
```

5.3 Macosx

Exécution

- lancer l'exécution de absimple_test.py

Résultat

```
$ python3 absimple_test.py
absimple_processAdd: 8 + 4 = 12
absimple_processSub: 8 - 4 = 4

>>>
$
```

6. Utilisation dans un programme Tcl

Exemple de source TCL utilisant absimple :

```
#--- file: absimple_test.tcl

package require absimple_tcl

set a 8
set b 4
set result [ absimple_processAdd $a $b ]
console::disp "absimple_processAdd  : $a + $b = $result \n"

set result2 [ absimple_processSub $a $b ]
console::disp "absimple_processAdd : $a + $b = $result2 \n"
```

Le programme **absimple_test.tcl** importe le package **absimple_tcl** qui charge la librairie.

6.1 Windows

Exécution avec Audela

- créer le répertoire **audela\lib\absimple**
- copier dans ce répertoire **absimple_tcl.dll , libabsimple.dll** et **pkgIndex.tcl**
- copier **absimple_test.tcl** dans le répertoire de travail de Audela
- lancer l'exécution de Audela
- lancer “source absimple.tcl” dans la console de Audela

Résultat

```
> source absimple_test.tcl
absimple_processAdd: 8 + 4 = 12
absimple_processSub: 8 - 4 = 4

>
```

6.2 Linux

Exécution avec Audela

- créer le répertoire **audela/lib/absimple**
- copier dans ce répertoire **absimple_tcl.so libabsimple.so** et **pkgIndex.tcl**

- copier **absimple_test.tcl** dans le répertoire de travail de Audela
- lancer l'exécution de Audela
- lancer “source absimple.tcl” dans la console de Audela

Résultat

```
> source absimple.tcl
absimple_processAdd: 8 + 4 = 12
absimple_processSub: 8 - 4 = 4

>
```

Exécution avec l'interpréteur Tcl standard de Linux

- créer le répertoire **tcl86/lib/absimple**
- copier dans ce répertoire **absimple_tcl.so , libabsimple.so** et **pkgIndex.tcl**
- copier **absimple_test.tcl** dans un répertoire de travail
- remplacer “console::disp” par “puts” dans **absimple_test.tcl**
- lancer l'exécution de tclsh depuis ce répertoire de travail
- lancer “source absimple.tcl” dans la console de Tcl

Résultat

```
$ tclsh
% source absimple_test.tcl
absimple_processAdd: 8 + 4 = 12
absimple_processSub: 8 - 4 = 4

% exit
$
```