GraspPlugin Manual

HiroNXProvider について

月, 10/31/2011 - 18:56 — asahi

HiroNXProvider は、双腕ロボット HIRO の制御 PC 上で動作して、HIRO の機能をRTミドルウェアから利用できるようにするRTコンポーネントです。

起動方法

端末を開いて HiroNXProvider をインストールしたディレクトリに移り、

```
% ./HiroNXProvider.py
```

を実行します。

利用方法

EclipseのRTシステムエディタや rtcon コマンド等を使って、HiroNXポートとHIROポートを別のRTコンポーネントと接続します。

終了方法

以下のどの方法でも終了できます。

1. RTシステムエディタで HiroNXProvider0 を右クリックして、コンテキストメニューで exit を選ぶ
2. rtexit localhost/(PC名).host_cxt/HiroNXProvider0.rtc を実行する
3. 起動した端末で Ctrl-C を打つ

環境設定

Jython(2.2以上)と、OpenRTM-aist-Java(1.0)が必要です。

Jython での実行について

プログラム本体の HiroNXProvider.py は Jython で動くように書かれていますが実機上で使われるインタプリタは /opt/grx/bin/hrpsyspy です。
しかし、開発用の一般のPCでテストを行うときのために、一般の Jython で起動することも可能となっています。
以下に実行例を示します。

```
% jython HiroNXProvider.py
Import failure: rtm
Import failure: hrp.simulator
Not exists: /opt/grx/HIRONX/share/hrpsys/jar/
Import failure: OpenHRP
mymodule init
HiroNX_i.init
hostname
hostname: localhost
```

rtm, hrp.simulator, OpenHRP のインポートに失敗し、Javaライブラリパス /opt/grx/HIRONX/share/hrpsys/jar/ が見つからない旨のメッセージが出ています。

この場合、当然ロボット制御命令は実行できませんが、HiroNXProvider は中断せず、コンソールに例外が発生した旨のメッセージを出力して続行します。

```
localhost
```

```
setupRobot
hostname: localhost
localhost
can not connect to localhost
```

setupRobot 命令を受け取っていますが、ロボットホスト localhost に接続できなかったというメッセージを表示して、再びコマンド待ちに入っています。

インストール方法

Subversion と Java SDK が必要です。

subversion レポジトリからソースを入手する場合

grasp01 からチェックアウトします。

```
% svn co http://grasp01/svn/vmrg/trunk/HiroNXProvider/
```

実行したディレクトリに HiroNXProvider が掘られ、その下にソースファイルが展開されます。

```
% cd HiroNXProvider
```

javac を実行します。

```
% javac *.java
```

これで HiroNXProvider.py を実行できるようになります。

ソースプログラムについて

HiroNXProvider.py: メインプログラム

HIROController_idl_example.py: HIROController の IDL コマンドの実装

HiroNX_idl_example.py: HiroNX の IDL コマンドの実装

rtc.conf: RTコンポーネントの設定ファイル

以下の Jython プログラムは、HIROの制御プログラムです。

bodyinfo.py

bodyinfo2.py

gui.py

guiinfo.py

sample.py

以下のファイルは、開発者がRTコンポーネントのスケルトンを生成するために利用するものです。

HIROController.idl: HIROController の IDL定義ファイル

HiroNX.idl: : HiroNX の IDL定義ファイル

RTC.xml: RT ビルダーのソース XML

以下すべて、IDL から生成された java ファイル。

CommonCommands.java

CommonCommandsHelper.java

CommonCommandsHolder.java

CommonCommandsOperations.java

CommonCommandsPOA.java

HiroNX.java

HiroNXHelper.java

HiroNXHolder.java
HiroNXOperations.java
HiroNXPOA.java
MotionCommands.java
MotionCommandsHelper.java
MotionCommandsHolder.java
MotionCommandsOperations.java
MotionCommandsPOA.java
_CommonCommandsStub.java
_HiroNXStub.java
_MotionCommandsStub.java

./CommonCommandsPackage:

RETURN_ID.java
RETURN_IDHelper.java
RETURN_IDHolder.java

./MotionCommandsPackage:

CarPosWithElbow.java
CarPosWithElbowHelper.java
CarPosWithElbowHolder.java
DoubleSeqHelper.java
DoubleSeqHolder.java
HgMatrixHelper.java
HgMatrixHolder.java
JointPosHelper.java
JointPosHolder.java
JointPosSeqHelper.java
JointPosSeqHolder.java
RETURN_ID.java
RETURN_IDHelper.java
RETURN_IDHolder.java
ULONGHelper.java

開発について

HiroNXProvider に機能を追加する場合、ほとんどは HiroNX.idl か、HIROController.idl に新しい命令を追加することになるはずです。
ここでは例として、HiroNX.idl に newCommand メソッドを追加してみます。
Java SDK が必要です。

IDLファイルに新しい命令を追加する

HiroNX.idl の内容は、以下のようになっています。

```
interface HiroNX {
  void setupRobot();
  void restart();
  void goInitial();
  void goOffPose();
  void servoOn();
  void servoOff();
  void calibrateJoint();
  void servoOnHands();
  void servoOffHands();
  void EngageProtectiveStop();
  void DisengageProtectiveStop();
```

```
    void reboot();
    void shutdown();

    void rhandOpen();
    void rhandClose();
    void lhandOpen();
    void lhandClose();
};
```

この最後の lhandClose のあとに、一行追加することにします。

```
    void lhandClose();
    void newCommand();
};
```

この IDL ファイルから、IDLコンパイラ「idlj」で Java ソースを生成します。

```
% idlj -fall HiroNX.idl
```

生成した java ファイルも、コンパイルしておきます。

```
% javac *.java
注:入力ファイルの操作のうち、未チェックまたは安全ではないものがあります。
注:詳細については、-Xlint:unchecked オプションを指定して再コンパイルしてください。
```

警告が出ますが無視します。

HiroNX_idl_example.py にnewCommandを実装する

HiroNX_idl_example.py を編集して、HiroNX_i クラスに newCommand メソッドを実装します。

HiroNX_idl_example.py の末尾付近に、

```
if __name__ == "__main__":
```

という行があります。この行より前が HiroNX_i クラスの定義部分です。ここでメソッド newCommand を定義するため、「def newCommand(self):
」の一行を挿入します。
defの前の空白の数を、前に定義されている lhandClose に合わせることと、行の末尾の (self): を忘れないことに注意してください。
Python に慣れないと、(self): を忘れますし、エディタ上で空白の数が合ってるように見えても、タブとスペースの違いでエラーになることもあるようです。

```
    def lhandClose(self):
        try:
            print "lhandClose"
            gui.lhandClose()
        except:
            print sys.exc_info()[0]
            print sys.exc_info()[1]
            print traceback.print_tb(sys.exc_info()[2])

    def newCommand(self):

if __name__ == "__main__":
    import sys
```

newCommand メソッドの内容を実装します。

```
def newCommand(self):
    try:
        print 'newCommand!'
    except:
        print sys.exc_info()[0]
        print sys.exc_info()[1]
        print traceback.print_tb(sys.exc_info()[2])
```

命令そのものは単純なprint文一つですが、例外処理を5行も書いています。
通常RTコンポーネントでエラーが出ても明確なエラー表示がないため、デバッグに手間取ることがあります。
例外処理でバックトレースを表示すれば、デバッグの手間をいくらか軽減できます。

別に修正した HiroNXGUI を接続して、newCommand ボタンを押してみます。
GUIで「新しいコマンド」ボタンを押したとき、HiroNXProvider 側の端末で「newCommand!」と表示されれば成功です。

```
% jython HiroNXProvider.py
Import failure: rtm
Import failure: hrp.simulator
Not exists: /opt/grx/HIRONX/share/hrpsys/jar/
Import failure: OpenHRP
mymodule init
HiroNX_i.init
hostname
hostname: localhost
newCommand!
newCommand!
```

うまく行きましたか?

‹ HiroNXGUI について　↑ 上位　HIRONXにおけるビジョンとの連携 ›

印刷用ページ　ログイン(登録)してコメントを投稿