WRS2018 トンネル災害対応部門 REL/UOA
チームロボットシステム導入マニュアル

阿部文明

2019 年 10 月 2 日

1　はじめに

このマニュアルは WRS2018 に参加した際のチームのシステムを再現するためのマニュアルです。
必要な開発環境

- Ubuntu 16.04 (インストール方法については省略)
- Choreonoid 1.7 以上のバージョン

2　環境構築

2.1　Choreonoid のインストール

2.1.1　最低限必要なパッケージのインストール

はじめに、インストール途中で必要となる機能等のインストールを行います。

```
sudo apt install openssh-server
sudo apt install git
sudo apt install ibus_mozc
```

ホームディレクトリ以下のディレクトリ名が日本語だと不便なので、英語表記に変更する。

```
LANG=C xdg-user-dirs-gtk-update
```

Choreonoid リポジトリの取得

```
git clone https://github.com/s-nakaoka/choreonoid.git
```

最新の Choreonoid のソースコードにアップデートする際は以下のコマンドを入力

```
cd choreonoid
git pull
```

依存パッケージのインストール

```
misc/script/install-requisites-ubuntu-16.04.sh
```

2.1.2 CMake によるビルド設定

ここでは Choreonoid をビルドするために必要な Makefile を生成します。ビルドの際に中間ファイルが生成されるため, buidl ディレクトリを作成する。

```
cd ~/choreonoid
mkdir build
cd build
cmake ..
```

2.1.3 Choreonoid のビルド

CMake により Makefile の生成が成功すれば、make コマンドで Choreonoid をビルドする。CMake を実行下ディレクトリで

```
make
```

を実行する。

マルチコア CPU であれば、”-j” オプションにより並列ビルドを行うことでビルド時間を短縮できる。

```
make -j4
```

2.2 OpenRTM プラグイン

2.2.1 OpenRTM-aist のインストール

WRS2018 ではロボットの通信に OpenRTM を使用した。そこで OpenRTM のインストールを行う。ここでは Ubuntu 用のパッケージをダウンロードし、インストールを行う。(2019-09-27 現在)

はじめに以下のパッケージのインストール。

```
sudo apt-get install gcc g++ make python-yaml
sudo apt-get install libomniorb4-dev omniidl omniorb-nameserver
sudo apt-get install python-omniorb-omg omniidl-python
sudo apt-get install cmake doxygen
sudo apt-get install default-jdk
```

以下の URL にあるパッケージをダウンロードし、任意のディレクトリに保存、解凍を行う。
(URL: https://openrtm.github.io/v112_download.html)

- OpenRTM-aist_1.1.2_ubuntu_16.04_amd_package.tar.gz
- OpenRTM-aist-Python_1.1.2_ubuntu16.04_amd_package.tar.gz

以下のスクリプトを実行し OpenRTM のインストールを行う。

```
cd ~/(download package path)/OpenRTM-aist_1.1.2_ubuntu_16.04_amd_package
sudo sh install-openrtm-deb-packages.sh
cd ~/(download package path)/OpenRTM-aist-Python_1.1.2_ubuntu16.04_amd_package
sudo sh install-openrtm-deb-packages.sh
```

2.2.2 OpenRTM-aist 1.1.2 の OutPort.h の修正

Choreonoid 公式のマニュアルには OpenRTM-aist 1.1.2 にはバグがあるとの報告がある。
具体的には以下 2 つの項目があげられている。

- Choeronoid 上で OpenRTM 関連機能を使用する際に落ちてしまう
- C++ で作成した RTC を RTShell で操作する際に MARSHAL_InvalidEnumValue 例外が出る

上記の不具合の解消のために OutPort.h を修正する。OutPort.h は /usr/include/openrtm-1.1/rtm/ ディレクトリにあるので、このディレクトリに、以下からダウンロードした修正版の OutPort.h をコピーして、上書きを行う。
(修正版 OutPort.h: https://choreonoid.org/ja/manuals/1.7/_downloads/OutPort.h)

2.2.3 CORBA の設定

カメラ画像やセンサデータのサイズを考慮して omniORB の最大メッセージサイズを増やす。設定ファイル /etc/omniORB.cfg の中の以下の項目を編集する。

```
giopMaxMsgSize = 2097152   #  2 MBytes
giopMaxMsgSize = 20971520  # 20 MBytes
```

2.2.4 OpenRTM プラグインのビルド

OpenRTM プラグインは Choreonoid のソースコードに同梱されている。Choreonoid のビルド前に行う CMake によるビルド設定 の cmake オプションで以下のオプションを ON にすることでビルドすることができます。

```
cd ~/choreonoid/build
ccmake ..
```

- ENABLE_CORBA
- BUILD_CORBA_PLUGIN
- BUILD_OPENRTM_PLUGIN

続いて、Choreonoid のビルドを行う。

```
make -j4
```

2.3 動作確認

ここで、OpenRTM プラグインの動作確認を行う。はじめに Choreonoid を立ち上げる。

```
cd ~/choreonoid/build
bin/choreonoid
```

「ファイル」→「プロジェクトの読み込み」をクリック。

~/choreonoid/sample/OpenRTM/OpenRTM-Tank.cnoid をクリック

シミュレーションを開始し、Tank モデルの動作が確認できれば、Choreonoid で OpenRTM の使用が可能となる。