新版Robots框架使用说明

# 编译新版Aris和Robots

从github上下载最新版Aris和Robots项目

git clone <https://github.com/py0330/Aris>

git clone [git@github.com:py0330/Robots.git](mailto:git@github.com:py0330/Robots.git)

第一种地址格式使用https协议，第二种使用ssh。两种格式都可以使用，但是我用https下载下来的项目每次push都要输用户名和密码，比较麻烦，所以建议使用ssh格式。当然下载别人的项目就无所谓了。

编译Aris

编译Robots

编译RobotIV\_Demo

# 范例演示

本版Robots以CMakeDemo目录下的RobotIV\_Demo项目作为演示范例。

## 操作步骤

1. 用Aris目录下的ethercat\_start.sh脚本开启ethercat
2. 进入可执行文件所在目录

cd /usr/Robots/CMakeDemo/Robot\_III/bin/

此时该目录下只有Server和Client两个文件

1. 运行Server

sudo ./Server

该目录下会自动生成en、ds、hm、ad、wk、ro这几个新的可执行文件。

Client只是在生成上述几个新文件时会用到，无需执行。

1. 添加PATH环境变量

PATH=$PATH:/usr/Robots/CMakeDemo/Robot\_III/bin

**注意：**上述方法的PATH在终端关闭后就会消失。

echo $PATH

查看当前PATH环境变量是否已将可执行文件所在路径添加进来。

添加完环境变量后可直接输入文件名执行相应的可执行文件，无需输入"./"。

1. 顺序执行相应的命令

en

功能：enable

参数：

|  |  |  |
| --- | --- | --- |
| 缩写 | 全称 | 含义 |
| -a（默认参数） | --all | 对所有电机进行操作 |
| -f | --first | 对第一组的三条腿进行操作 |
| -s | --second | 对第二组的三条腿进行操作 |
| -m | --motor | 对单个电机进行操作，后接=i，i为电机序号 |

命令实例：

en –a 使能所有电机

en –m=15单电机使能

ds

功能：disable

参数：与en相同

hm

功能：home

参数：与en相同

ad

功能：与原来的home2start相同

参数：-f，-s

wk

功能：walk

参数：

## 单电机调试

**修改home模式设置**

做单电机调试时建议修改initParam.homeTorqueLimit这个参数，减小到100以内。

查找该参数发现它在Robots/src/Robot\_Server目录下的Robot\_Server.cpp和main.cpp都有出现，但main.cpp其实并没有参与编译，所以需要修改的是Robot\_Server.cpp。

**注意：**修改该参数后需要将Robots和自己编写的项目依次重新编译一遍才能生效。

# 编写自己的项目

将Robots/CMakeDemo中的RobotIV\_demo文件夹拷贝到桌面或其他自己习惯的目录，并重命名为自己的项目名。

trajectory\_generator.h和trajectory\_generator.cpp并没有被调用，可以删掉。

## 修改Server主函数

int main**()**

**{**

auto rs **=** Robots**::**ROBOT\_SERVER**::**GetInstance**();**

rs**->**CreateRobot**<**Robots**::**ROBOT\_III**>();**

rs**->**LoadXml**(**"/usr/Robots/CMakeDemo/Robot\_III/resource/HexapodIII\_Move.xml"**);** //替换为自己写的xml文件

rs**->**AddGait**(**"wk"**,**Robots**::**walk**,**parseWalk**);**

rs**->**AddGait**(**"ad"**,**Robots**::**adjust**,**parseAdjust**);**

/\* 我添加的步态，“move”是xml文件里定义的命令，也是生成的可执行文件的文件名；parseMove用于解析用户输入的命令；move2则是步态规划函数 \*/

rs**->**AddGait**(**"move"**,**move2**,**parseMove**);**

rs**->**Start**();**

/\*\*/

std**::**cout**<<**"finished"**<<**std**::**endl**;**

Aris**::**Core**::**RunMsgLoop**();**

**return** 0**;**

**}**

## 修改xml文件

RobotIV\_demo/resource/client.xml文件其实并没有用到，RobotIV\_demo项目实际读取的是Robots/src/HexapodIII/resource/ HexapodIII.xml

自己编写的xml可以在Robots/src/HexapodIII/resource/ HexapodIII.xml的基础上修改。在<Commands></Commands>标签内添加自己的命令。

例如我添加的命令：

<move>

<move\_param type=**"group"**>

<component abbreviation=**"c"** type=**"string"** default=**"bd"**/>

<move\_pos type=**"unique"** default=**"relative"**>

<relative type=**"group"**>

<u abbreviation=**"u"** type=**"double"** default=**"0"**/>

<v abbreviation=**"v"** type=**"double"** default=**"0"**/>

<w abbreviation=**"w"** type=**"double"** default=**"0"**/>

</relative>

<absolute type=**"group"**>

<x abbreviation=**"x"** type=**"double"**/>

<y abbreviation=**"y"** type=**"double"**/>

<z abbreviation=**"z"** type=**"double"**/>

</absolute>

</move\_pos>

</move\_param>

</move>

其中type=**"group"**属性表示下级参数必须都具备。

type=**"unique"**表示下级参数（或参数组）是二选一或多选一，只能输入其中一个（或一组）参数。

Default表示默认参数。

Abbreviation表示缩写，只能有一个字母。

TIPS：

1. 编写完xml文件可先用浏览器打开，若无法正常显示，则说明格式有问题。
2. Server运行时若出现“parse failed”错误提示，可能是parse程序有问题，也可能是xml文件有问题。

## 添加命令解析函数和步态规划函数

可以直接在Server/main.cpp里编写这两个函数。

原来RobotIV\_demo的adjust和walk函数是在Robot\_Gait.cpp里定义的。在main.cpp里定义步态函数，须包含Robot\_Gait.h和Robot\_Base.h两个头文件。

GAIT\_PARAM\_BASE是在命令解析函数（parseXXXX）和步态规划函数之间传递参数的结构体，若GAIT\_PARAM\_BASE的成员不足以表达新步态所需要的所有参数，则可以自己派生一个新的结构体。

附上我添加的代码：

struct MOVES\_PARAM **:**public Robots**::**GAIT\_PARAM\_BASE

**{**

double targetPee**[**18**]{**0**};**

double targetBodyPE**[**6**]{**0**};**

std**::**int32\_t periodCount**;**

int comID**;** //移动的部件（component）序号

bool isAbsolute**{false};** //用于判断移动命令是绝对坐标还是相对坐标

**};**

Aris**::**Core**::**MSG parseMove**(**const std**::**string **&**cmd**,** const map**<**std**::**string**,** std**::**string**>** **&**params**)**

**{**

MOVES\_PARAM param**;**

double targetPos**[**3**];** //移动目标位置

**for(**auto **&**i**:**params**)**

**{**

**if(**i**.**first**==**"component"**)**

**{**

**if(**i**.**second**==**"lf"**)**

**{**

param**.**comID**=**0**;**

**}**

**else** **if(**i**.**second**==**"lm"**)**

**{**

param**.**comID**=**1**;**

**}**

**else** **if(**i**.**second**==**"lr"**)**

**{**

param**.**comID**=**2**;**

**}**

**else** **if(**i**.**second**==**"rf"**)**

**{**

param**.**comID**=**3**;**

**}**

**else** **if(**i**.**second**==**"rm"**)**

**{**

param**.**comID**=**4**;**

**}**

**else** **if(**i**.**second**==**"rr"**)**

**{**

param**.**comID**=**5**;**

**}**

**else** **if(**i**.**second**==**"bd"**)**

**{**

param**.**comID**=**6**;**

**}**

**else**

**{**

std**::**cout**<<**"parse failed"**<<**std**::**endl**;**

**return** MSG**{};**

**}**

**}**

//绝对坐标移动

**else** **if(**i**.**first**==**"x"**)**

**{**

targetPos**[**0**]=**stod**(**i**.**second**);**

param**.**isAbsolute**=true;**

**}**

**else** **if(**i**.**first**==**"y"**)**

**{**

targetPos**[**1**]=**stod**(**i**.**second**);**

param**.**isAbsolute**=true;**

**}**

**else** **if(**i**.**first**==**"z"**)**

**{**

targetPos**[**2**]=**stod**(**i**.**second**);**

param**.**isAbsolute**=true;**

**}**

//相对坐标移动

**else** **if(**i**.**first**==**"u"**)**

**{**

targetPos**[**0**]=**stod**(**i**.**second**);**

**}**

**else** **if(**i**.**first**==**"v"**)**

**{**

targetPos**[**1**]=**stod**(**i**.**second**);**

**}**

**else** **if(**i**.**first**==**"w"**)**

**{**

targetPos**[**2**]=**stod**(**i**.**second**);**

**}**

**else**

**{**

std**::**cout**<<**"parse failed"**<<**std**::**endl**;**

**return** MSG**{};**

**}**

**}**

**if(**param**.**comID**==**6**)**

**{**

std**::**copy\_n**(**targetPos**,** 3**,** param**.**targetBodyPE**);**

**}**

**else**

**{**

std**::**copy\_n**(**targetPos**,** 3**,** **&**param**.**targetPee**[**3**\***param**.**comID**]);**

param**.**legNum**=**1**;**

param**.**motorNum**=**3**;**

param**.**legID**[**0**]=**param**.**comID**;**

int motors**[**3**]** **=** **{** 3**\***param**.**comID**,** 3**\***param**.**comID**+**1**,** 3**\***param**.**comID**+**2 **};**

std**::**copy\_n**(**motors**,** 9**,** param**.**motorID**);**

**}**

param**.**periodCount**=**3000**;**

Aris**::**Core**::**MSG msg**;**

msg**.**CopyStruct**(**param**);**

std**::**cout**<<**"finished parse"**<<**std**::**endl**;**

**return** msg**;**

**}**

int move2**(**Robots**::**ROBOT\_BASE **\*** pRobot**,** const Robots**::**GAIT\_PARAM\_BASE **\*** pParam**)**

**{**

const MOVES\_PARAM **\***pMP **=** **static\_cast<**const MOVES\_PARAM **\*>(**pParam**);**

double realTargetPee**[**18**];**

double realTargetPbody**[**6**];**

std**::**copy\_n**(**pMP**->**beginPee**,** 18**,** realTargetPee**);**

std**::**copy\_n**(**pMP**->**beginBodyPE**,** 6**,** realTargetPbody**);**

//绝对坐标

**if** **(**pMP**->**isAbsolute**)**

**{**

**if(**pMP**->**comID**==**6**)**

**{**

std**::**copy\_n**(**pMP**->**targetBodyPE**,** 6**,** realTargetPbody**);**

**}**

**else**

**{**

std**::**copy\_n**(&(**pMP**->**beginPee**[**pMP**->**comID**\***3**]),** 3**,** **&**realTargetPee**[**pMP**->**comID**\***3**]);**

**}**

**}**

//相对坐标

**else**

**{**

**if(**pMP**->**comID**==**6**)**

**{**

**for(**int i**=**0**;**i**<**6**;**i**++)**

**{**

realTargetPbody**[**i**]+=**pMP**->**targetBodyPE**[**i**];**

**}**

**}**

**else**

**{**

**for(**int i**=**0**;**i**<**18**;**i**++)**

**{**

realTargetPee**[**i**]+=**pMP**->**targetPee**[**i**];**

**}**

**}**

**}**

double s **=** **-(**PI **/** 2**)\***cos**(**PI **\*** **(**pMP**->**count **+** 1**)** **/** pMP**->**periodCount **)** **+** PI **/** 2**;**

/\*插值当前的末端和身体位置\*/

double pEE**[**18**],** pBody**[**6**];**

**for** **(**int i **=** 0**;** i **<** 18**;** **++**i**)**

**{**

pEE**[**i**]** **=** pMP**->**beginPee**[**i**]** **\*** **(**cos**(**s**)** **+** 1**)** **/** 2 **+** realTargetPee**[**i**]** **\*** **(**1 **-** cos**(**s**))** **/** 2**;**

**}**

**for** **(**int i **=** 0**;** i **<** 6**;** **++**i**)**

**{**

pBody**[**i**]** **=** pMP**->**beginBodyPE**[**i**]** **\*** **(**cos**(**s**)** **+** 1**)** **/** 2 **+** realTargetPbody**[**i**]** **\*** **(**1 **-** cos**(**s**))** **/** 2**;**

**}**

pRobot**->**SetPee**(**pEE**,** pBody**);**

/\*返回剩余的count数\*/

**return** pMP**->**periodCount **-** pMP**->**count **-** 1**;**

**}**