模块通信和日志、故障诊断使用说明

ROS 系统有一个小乌龟程序来演示系统运行的方式,我们编写了类似程序来演示模块 通信的运行方式。

本文档会使用 Tool 下的数个程序,包括 sample1, sample2, ivdiagnosis, view ivlog 四个模块。

- 编写 sample1 和 sample2 之间通信的消息

在 src/include/proto 下增加了一个 proto 文件, 名称为 samplemsg.proto, 定义传

递的消息。如下:

syntax = "proto2";

package iv;

```
message samplemsg
{
    required int32 mvalue = 1;
    optional int64 msendtime = 2;
```

};

执行 src/include/proto 下的 protomake.sh 脚本生成头文件和源文件。

二 编写 sample1 模块

1) 增加库引用

在.pro 文件内加入如下代码

```
LIBS += -lprotobuf
INCLUDEPATH += $$PWD/../../include/msgtype
```

INCLUDEPATH += \$\$PWD/../../include/

LIBS += -L\$\$PWD/../../bin/ -Ixmlparam -Imodulecomm -livlog -livfault

增加了对 xmlparam 库、modulecomm 库、ivlog 库、ivfault 库的引用。

2) 修改 main.cpp 文件

包含头文件。

#include "xmlparam.h"
#include "ivlog.h"
#include "ivfault.h"

定义全局变量

iv::lvlog * givlog; iv::lvfault * givfault; std::string gstrmemname;

从 xml 配置文件导入模块名称定义和共享内存消息名称定义。

```
QString strpath = QCoreApplication::applicationDirPath();
if(argc < 2)
    strpath = strpath + "/sample1.xml";
else
    strpath = argv[1];
std::cout<<strpath.toStdString()<<std::endl;
iv::xmlparam::Xmlparam xp(strpath.toStdString());
gstrmodulename = xp.GetParam("modulename","sample1");
gstrmemname = xp.GetParam("msgname","sample1");
```

初始化日志变量和故障诊断变量。

givlog = new iv::lvlog(gstrmodulename.data()); givfault = new iv::lvfault(gstrmodulename.data());

3) 修改 mainwindow.h 文件

增加变量定义

extern iv::lvlog * givlog; extern iv::lvfault * givfault; extern std::string gstrmemname;

定义共享内存的句柄

void * mpa;

4) 修改 mainwindow.cpp 文件

在构造函数内初始化共享内存句柄。

mpa = iv::modulecomm::RegisterSend(gstrmemname.data(),1000,1);

```
将当前 Slider 的状态写入到共享内存中。
```

```
iv::samplemsg xsam;
xsam.set_mvalue(value);
xsam.set_msendtime(QDateTime::currentMSecsSinceEpoch());
```

```
int ndatasize = xsam.ByteSize();
char * str = new char[ndatasize];
std::shared_ptr<char> pstr;pstr.reset(str);
if(!xsam.SerializeToArray(str,ndatasize))
```

```
{
```

```
std::cout<<"MainWindow::on_horizontalSlider_valueChanged serialize
error."<<std::endl;</pre>
```

return;
}
iv::modulecomm::ModuleSendMsg(mpa,str,ndatasize);

写入日志和更新当前故障状态。

```
givlog->debug("send slider value %d",xsam.mvalue());
givfault->SetFaultState(0,0,"Run OK");
```

三 编写 sample2 模块

1) 增加库引用

在.pro 文件内加入如下代码

LIBS += -lprotobuf INCLUDEPATH += \$\$PWD/../../include/msgtype

INCLUDEPATH += \$\$PWD/../../include/ LIBS += -L\$\$PWD/../../bin/ -Ixmlparam -Imodulecomm -livlog -livfault

增加了对 xmlparam 库、modulecomm 库、ivlog 库、ivfault 库的引用。

2)修改 main.cpp 文件

包含头文件。

#include "xmlparam.h"
#include "ivlog.h"
#include "ivfault.h"

定义全局变量

```
iv::lvlog * givlog;
iv::lvfault * givfault;
std::string gstrmemname;
```

从 xml 配置文件导入模块名称定义和共享内存消息名称定义。

```
QString strpath = QCoreApplication::applicationDirPath();
if(argc < 2)
    strpath = strpath + "/sample2.xml";
else
    strpath = argv[1];
std::cout<<strpath.toStdString()<<std::endl;
iv::xmlparam::Xmlparam xp(strpath.toStdString());
gstrmodulename = xp.GetParam("modulename","sample2");
gstrmemname = xp.GetParam("msgname","sample1");
```

初始化日志变量和故障诊断变量。

givlog = new iv::lvlog(gstrmodulename.data()); givfault = new iv::lvfault(gstrmodulename.data());

3) 修改 mainwindow.h 文件

增加变量定义

extern iv::lvlog * givlog; extern iv::lvfault * givfault; extern std::string gstrmemname;

定义共享内存的句柄

void * mpa;

增加回调函数。

void UpdateSlider(const char * strdata,const unsigned int nSize,const unsigned int index,const QDateTime * dt,const char * strmemname);

4) 修改 mainwindow.cpp 文件

在构造函数内初始化共享内存句柄,并关联回调函数。

ModuleFun funupdate

=std::bind(&MainWindow::UpdateSlider,this,std::placeholders::_1,std::placeholder s::_2,std::placeholders::_3,std::placeholders::_4,std::placeholders::_5);

mpa = iv::modulecomm::RegisterRecvPlus(gstrmemname.data(),funupdate);

在回调函数内解析接收到的数据,并控制 Slider 的位置。

```
iv::samplemsg xsam;
if(!xsam.ParseFromArray(strdata,nSize))
{
    std::cout<<" MainWindow::UpdateSlider parese error."<<std::endl;
    return;
}
```

ui->horizontalSlider->setValue(xsam.mvalue()); ui->lineEdit->setText(QString::number(QDateTime::currentMSecsSinceEpoch() - xsam.msendtime()));

写入日志和更新当前故障状态。

givlog->debug("send slider value %d",xsam.mvalue()); givfault->SetFaultState(0,0,"Run OK");

四 效果

在 sample1 内拖动 Slider , sample2 内的 Slider 会同步运动。

活动	🔳 sample1 🔻	星期二 14:22	+					
-	sample2/mainwindow.cpp @ sample2 [124946:1249663] - Qt Creator							
	samplet 0 f	(1) (2) (2) (2) (2) (2) (2) (2) (2) (2) (2						
-	Jonipier	Sample2 Sample2 Sider(const char *, const unsigned int, const QDateTime *, const char *): void	¢ Line: 29, Col: 1 ⊟+					
9			Î					
_								
•								
0		Latterity(ins) 1						
-		<pre>ateSider.this.std::placeholders::_1,std::placeholders::_2,std::placeholders::_3,std::placeholders::_4,st ime.data().fnundate):</pre>	d::placeholders::_5);					
==								
	amain.cpp	18						
-		13] 20 ,						
	2 mainwindow.ui	21 22 * Mastelendeur-shoftelindeur()						
	ample2.pro	23 { 23 }						
	 Headers /home/suchuli/gt/modularization 	24 delete ut; 25 j						
<u>M</u>	mainwindow.h	20 27 * void MaimMindow::UpdateSlider(const char *strdata, const unsigned int nSize, const unsigned int index, const QDateTime *dt, const char *strmemname)						
	 Sources image: A contract of the second second	20 { 22 {v::sarolersr.xsan:						
• 🙆	🖻 main.cpp	30 + if(lxsan.ParsoFromArray(strdata,nSize))						
_	* Porms	32 std::cout<<" MainWindow::UpdateSlider parese error."< <std::endl;< th=""><th></th></std::endl;<>						
	Z mainwindow.ui	33 return; 34)						
	a ui_ads_hmi.pro	<pre>35 ui->horizontalSlider->setValue(xsam.mvalue()); 36 ui->linedit->setTavt(Ostring::number(Obateline::currentMSecsSinceEooch() - xsam.msendtime()));</pre>						
V7	 iiii common iiii Headers 	37 miulam-bidebund"renewad slider value 56 latency is 51d" yaam mualus() ObtaTime+rurrentMSarsSinaEnorh() = yaam meandtime())+						
		givfault-Ssefaultstate(9,6, "Run OK");						
- Qt	17.ML		*					
		位用技巧地出 = 1 · / F · / F · / F · · · · · · · · · · ·	~ □					
• 📝		Sample's Sample's Vision Sample's A Vision Sample's Sam						
		/home/yuchuli/qt/modularization/src/tool/build-sample2-Debug/sample2.xml						
>_	sample2							
· — ·	_ ↓ •							
	Debug							
	b.							
:::	100							
	P Type to locate (Ctrl 1 (200) 2		• n					

五 日志系统

打开 view_ivlog,将 Class 调到 Verbose 来显示所有日志。



六 故障诊断系统

在故障诊断系统的 yaml 标定文件内增加 sample1 和 sample2。在 module 下增加

模块,然后增加模块的定义,moudlename(模块名称)和titile(显示名称)。

module:

- driver_lidar
- driver_can
- driver_gps
- driver_map
- detect_radar
- detect_lidar
- decition
- contoller
- ui
- platform
- sample1
- sample2

driver_lidar:

modulename: driver_lidar_rs16

title: 激光雷达驱动

driver_can:

```
modulename: driver_can_kvaser
```

```
title: can 卡
driver gps:
 modulename: driver_gps_hcp2
 title: 惯导
driver map:
 modulename: driver_map_trace
 title: 轨迹地图
detect_radar:
 modulename: detection_radar_delphi_esr
 title: 毫米波雷达感知
detect_lidar:
 modulename: detect_lidar_grid
 title: 激光雷达感知
decition:
 modulename: decition_ge3
 title: 决策
contoller:
 modulename: contoller_ge3
 title: 控制
ui:
 modulename: ui_ads_hmi
 title: 界面
platform:
 modulename: platform
 title: 平台通信
sample1:
 modulename: sample1
 title: 范例 1
sample2:
 modulename: sample2
 title: 范例 2
```

将 ivdiagnosis.yaml 放到 ivdiagnosis 的运行路径。运行效果如图所示。

活动	🗉 view_i	vlog v		星期二 14:	28		中▼ ♥╡⊾@▼	Ŧ
-				sample2/mainwindow.cpp @ sample2	[124946:124969M] - Qt Creator			
	文件(E)	编辑(E) 构建(B) 调试(D) Analyze 工具(I)	控件(W) 帮助(H)					
-				10173-016-16-16		0	Line: 15, Col: 31	8•
		1		DX (54 15) E1 77 C1/4				^
	102							
	E	激光雷达驱动	can卡	惯导	轨迹地图			
	1610	-1	-1	-1	-1			
		尤反頭	大反策	尤权读	尤反馈			
0						al and all designs of a second second		
	ŵ					placenoiders::_3,sto::place	<pre>iotders::_4,std::ptacenotders::_5);</pre>	1
==		毫米波雷达感知	激光雷达感知	决策	控制			
		-1	-1	-1	-1			
-0-		无反馈	无反馈	无反馈	无反馈			
A	0							
-	植物							
?		界面	平台通信	范例1	范例2			
		-1	-1	0	0			
h		无反馈	无反馈	Run OK	Run OK			
						'ine *dt, const char *strmem	nane)	
						view ivion		50
•		sample1	000	Sample2 🔤 😳 😳		incu_intog	-	-
					[sample1 2020-04-07 14:24:11:935 14830 1]: send s	lider value 36		
• 🔕					[sample1 2020-04-07 14:24:11:943 14830 1]: send s [sample1 2020-04-07 14:24:11:951 14830 1]: send s	lider value 37 lider value 39	Module All	
					[sample1 2020-04-07 14:24:11:959 14830 1]: send s	ilider value 41	RID All	
• 7/2			-		[sample2 2020-04-07 14:24:11:936 146411]; recew	ed slider value 37 latency is 1	FID AIL	
			Lattency(ms) 0	[sample2 2020-04-07 14:24:11:951 14841 1]: receiv [sample2 2020-04-07 14:24:11:960 14841 1]: receiv	ed slider value 39 latency is 0 ed slider value 41 latency is 0	Class Verbose	
0					[sample2 2020-04-07 14:24:11:967 14841 1]: receive	ed slider value 43 latency is 0		
					sample2 2020-04-07 14:24:11:975 14841 1 J: recew sample2 2020-04-07 14:24:11:983 14841 1 J: recew	ed slider value 46 latency is 0 ed slider value 48 latency is 0		
					[sample2 2020-04-07 14:24:11:992 14841 1]: receiv	ed slider value 51 latency is 1		
• 🖌					[sample2 2020-04-07 14:24:11:999 14:041 1]: receiv	ed slider value 56 latency is 0		
		4			[sample1 2020-04-07 14:24:11:967 14830 1]: send s [sample1 2020-04-07 14:24:11:975 14830 1]: send s	lider value 43 lider value 46		
>_	ivdiagnosis		Thave module size is 12	in a second s	[sample1 2020-04-07 14:24:11:983 14830 1]: send s	lider value 48		
	<u> </u>		connect is 1		[sample1 2020-04-07 14:24:11:991 14830 1]: send s	lider value 55		
			相比中共和国第一 The process was ended forcefully.			lider value 56		
			/home/yuchuli/qt/modularization/	<pre>src/tool/build-ivdiagnosis-Desktop_Qt_5_10_1_</pre>	[sample1 2020-04-07 14:24:12:023 14830 1]: send s	lider value 60		
_			Starting /home/yuchuli/qt/modula	rization/src/tool/build-ivdiagnosis-Desktop_Q	Isample1 2020-04-07 14:24:12:047 14830 1 : send s [sample2 2020-04-07 14:24:12:015 14841 1 : received]	lider value 61 ed slider value 58 latency is 0		
	h.		connect is 1		[sample2 2020-04-07 14:24:12:023 14841 1]: recew	ed slider value 60 latency is 0		
					[sample2 2020-04-07 14:24:12:048 14841 1]: recew	ro silder value of latency is 1		
		P, Type to locate (Ctrl 1 0 2	Search Results 3 应用程序输出	4 编译辗出 5 Debugger Console 7 Version C	a			

在故障诊断系统界面上能看到每个模块的状态。颜色对应的状态分别为:

- 灰色 不在线
- 黄色 警告
- 红色 故障
- 绿色 正常