ARCS脚本编程

日录 1.ARCS 脚本简介 2.LUA语法 2.1 常用变量类型 2.2 自定义函数 2.3 条件和循环语句 3. 查看ARCS脚本API定义 4. 在示教器中新建和编写脚本的操作步骤 5.SDK脚本编程操作步骤

1. ARCS 脚本简介

ARCS的脚本编程是指用脚本语言来控制机器人。基于LUA,遵循LUA语言的语法规则。支持 示教器脚本编程,支持SDK发送本地脚本程序到控制器。

2. LUA语法

LUA语法教程可参考 https://www.runoob.com/lua/lua-tutorial.html

2.1 常用变量类型

类型	说明	示例
nil	表示一个无效值, 在条件表 达式中相当于 false。	nil
Boolean	true或false。	true false
number	数字。包括整型、浮点型。	1 -5 3.1415
string	字符串。 一对双引号或者单引号之间 的字符。	"abc" 'abc'
function	自定义函数。	print
table	表。用{}表示。	{1, 2, 3}

2.2 自定义函数

- LUA 函数可以无返回值,也可以有一个或者多个返回值
- 举例:



2.3 条件和循环语句

- LUA 函数可以无返回值,也可以有一个或者多个返回值
- 举例:
 - 自定义多返回值函数:
 function add(a.b)
 return a.b.(a+b)
 end
 函数调用:
 x.y.z = add(a.b)

3. 查看ARCS脚本API定义

和C++SDK 的API一样,ARCS脚本API定义在SDK软件包的include文件夹的头文件里。 以motion_control.h中moveJoint接口为例,注释中的Lua函数原型就是Lua函数定义。



4. 在示教器中新建和编写脚本的操作步骤

注:示教器中的.pro文件和.lua文件均保存在 /root/arcs_ws/program路径下

1. 在示教器中新建一个脚本,命名为test_movej,点击保存

rob1	🛐 仿真 🛛 手动			🚺 正常	2023-07-27 06:45:07 am
一 1	★★ 图形 支量	6		下子 下 打开 保存	程序 Untitle*
	脚本代码行	文件	1 💿 初始变量		
く/2 编程	脚本代码将从选定的文件中引入:		2		
V	打开 编辑 清除	保存为	(5)		
配置	1		\smile		
中 移动					
IV0		查看:	🚞 /root/arcs_ws/program	•	G O O 🎑 🗄 🗏
1/0		📕 计算机	名称	▼ 大小	类型 修改日期
,,0		🖾 /	📄 circle.lua	2.29	KiB lua File 7/10/44 PM
			example_move_circle.lua	4.00	KiB lua File 7/19/44 P
			example_move_circle2.lua	3.78	KiB lua File 7/10/48 Pl
			example_move_circle3.lua	4.81	AB lua File 7/10/48 Pr
			test 20230721 lua	72 88 hv	es lua File 7/24/13 Al
			test pro 20230721.lua	2.71	KiB lua File 7/21/50 Al
	3		Untitled_0.lua	3.70	KiB lua File 7/18/06 Pt
		1	Untitled_1.lua	2.72	KiB lua File 7/24/44 Al
	基础 高级 模板 插件	•	Untitled_2.lua	3.39	KiB lua File 7/24/39 PM
100% 0 mm/s	四 脚本 码垛 轨途回放	8	4		9
		文件名称(<u>N</u>):	test_movej		<mark>,</mark> 保存(<u>S</u>)
	. Contraction of the second se	文件类型:	脚本(*.lua)		🔪 🗶 取消
0000					

2. 点击编辑,编写test_movej.lua程序

		rob1 🔇	自伤真 幻手	动					T	运行 🛛 正常	2023-07-24 1	1:10:21 am	¢ Line: 30, Col: 14 🛛 🖯
· · · · ·		2.0 2.0 (1) (1) (1) (1) (1) (1) (1) (1) (1) (1)	戦形 戦形 脚本代码 脚本代码将从选定 打开 1	変量 的文件中引入: 協調 (品語 2	: 文件 xample_movej.lua ま 保存为	編辑 2. 1 ⑥ 初始变量 2 ○ 前 机器人编辑 3 ◎ example	授索 🌇 🔤 I	Fille 📄	打开 📷 🤇 🕅	ā		程序 Unti*	
1 p 2 - 3 W 4 W 5 W 6 7 - 8 m 9 tr 10	i = 3.141592653589 - 关节角,单位: 弧度 Vaypoint_1_q = {-0.9 Vaypoint_2_q = {-16. Vaypoint_3_q = {30.2 - 以关节运动的方式依 noveJoint(Waypoint_ extmsg("joint move t	79323846 [13/180*pi, -15.7 67/180*pi, -21 12/180*pi, -14.9 以次经过3个路点: 1_q, 1, 1, 0, 0) io waypoint_1")	77/180*pi, 97. I.25/180*pi, 56 97/180*pi, 61.4 : Waypoint_1、	57/180*pi, 23.33/ i.19/180*pi, −12.5 42/180*pi, −13.62 Waypoint_2和Way	180*pi, 89.99/180*p 7/180*pi, 89.99/180 7/180*pi, 89.99/180 /180*pi, 89.99/180 point_3	ıl, −1.82/180*pi})*pi, −17.56/180*pi} pi, 29.33/180*pi}							×
	q	w	Θ		r	t	у	, i	u	i	c		Р
	a		s	d	f	g		h	1		k	1	
	\diamond	:	z	x	с	v		b	n		m		
	? 123	Ă	ŧ文	₽				←	\rightarrow		ب (ع		×

test_movej.lua代码:

```
pi = 3.14159265358979323846
 1
       -- 关节角, 单位: 弧度
 2
       Waypoint_1_q = {-0.93/180*pi, -15.77/180*pi, 97.57/180*pi,
 3
   23.33/180*pi, 89.99/180*pi, -1.82/180*pi}
       Waypoint 2 g = {-16.67/180*pi, -21.25/180*pi, 56.19/180*pi,
 4
   -12.57/180*pi, 89.99/180*pi, -17.56/180*pi}
       Waypoint_3_q = {30.22/180*pi, -14.97/180*pi, 61.42/180*pi,
 5
   -13.62/180*pi, 89.99/180*pi, 29.33/180*pi}
 6
 7
       -- 设置运动速度比率
       setSpeedFraction(0.8)
 8
       -- 以关节运动的方式依次经过3个路点: Waypoint_1、Waypoint_2和Waypoint_3
 9
       moveJoint(Waypoint 1 q, 1, 1, 0, 0)
10
       -- 将信息打印到aubo_control日志中
11
       textmsg("joint move to waypoint 1")
12
13
14
       moveJoint(Waypoint 2 q, 1, 1, 0, 0)
       textmsg("joint move to waypoint_2")
15
16
       moveJoint(Waypoint_3_q, 1, 1, 0, 0)
17
       textmsg("joint move to waypoint_3")
18
19
```

3. 保存.pro工程,命名为pro_movej。注意.pro文件名不能和.lua文件名相同。

rob1	S 仿真 手动	1 运行	示 [【] 〕正常 2023-07-24 12:06:42
—	\$\$\$ \$\$\$	 (4) (4) (4) (4) (4) (4) (4) (4) (4) (5) (5) (5) (5) (5) (5) (5) (5) (5) (5) (5) (5) (5) (5) (5) (5) (5) (5) (5) (5) <th>ار Untitle</th>	ار Untitle
		1 ⑧ 初始变量 2 ☐ 國 机器人编程 3 @ example_movej.lua	
中 移动			
1/0 1/0		4	

因为在保存.pro工程的时候,同时会在/root/arcs_ws/program路径下生成一个与.pro文件名相同的.lua文件,机器人实际执行的是这个lua文件。

rob1	S 仿真 2 手动	-	「运行	🕑 正常	2023-07-27 07:01:10 am
	命令 图形 变量 脚本代码 行 文件	編 上 上 上 編 援索 1 ③ 約 2 二 1 1 1		「王」 「 打开 保存	程序 序 pro_m
编程	脚本代码将从选定的文件中引入: test_movej.lua	3 @ test	movei lua		
● ■ ● ● ● ● ● ● ● ● ● ● ● ● ●	打开 编辑 清除 保存为 1 pi = 3.14159265358979323846		(novej.iua		
<u>100%</u>					
robl	S 仿真 予 手动		ゴ运行	◎ 正常	2023-07-27 07,02:09 am
		▲	新建	 -	2 存 Pro_m
主页	脚本代码 行 文件	1 local aubo = req	uire('aubo)' <mark>)</mark>	-
	脚本代码将从选定的文件中引入: test_movej.lua 打开 编辑 清除 保存为	2 local sched = sc 3 local math = aut 4 5 local sleep = sch 6 local thread = sc	hed or au bo.math o ned. sleep ched.threa	bo.sched r math 机器人道 运行这个 ad	运行工程实质上是)脚本文件
	1 pi = 3.14159265358979323846	7 local sync = sch	ed.sync d.run		
+	2 ≂ /> // // 300 3 Waypoint_1_q = {-0.93/180*pi, -15.77/180*pi, 97.5 4 Waypoint_2 a = { 1.8.27/190*pi, -0.105/190*pi, 97.5	9 local kill = sched	d.kill		
移动	 Waypoint_s_q = (-10.67/100°pi, -21.29/100°pi, 56. Waypoint_3_q = (30.22/180°pi, -14.97/180°pi, 61.4: 	11 function p_pro_r	novej() sched sel	ect_robot(1)	
1/0 1/0	 以关节运动的方式依次经过3个路点: Waypoint_1、 moveJoint(Waypoint_1_q, 1, 1, 0, 0) textmsg("joint move to waypoint_1") moveJoint(Waypoint_2_q, 1, 1, 0, 0) textmsg("joint move to waypoint_2") moveJoint(Waypoint_3_q, 1, 1, 0, 0) textmsg("joint move to waypoint_3") 	13 setCollisionSt 14 setCollisionL 15 setHomePosi 16 modbusDelett 17 setDigitalInpu 18 setDigitalOut 19 setPayload(0, 20 setTcpOffset 21 setToolVoltag 22 setToollolnpu 23 setToollolnpu 24 setToollolnpu	topType(0 avel(6) tion({0,-0 eAllSignal rtActionDe putRunsta , {0,0,0}, { ({0,0,0,0}, ({0,0,0,0,0}) ({0,0,0,0,0}) ({0,0,0,0,0}) ({0,0,0,0,0,0}) ({0,0,0,0,0,0}) ({0,0,0,0,0,0}) ({0,0,0,0,0,0}) ({0,0,0,0,0,0}) ({0,0,0,0,0,0,0}) ({0,0,0,0,0,0,0}) ({0,0,0,0,0,0,0,0}) ({0,0,0,0,0,0,0,0,0}) ({0,0,0,0,0,0,0,0,0,0,0}) ({0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,	.261799387 s() sfault() teDefault() 0,0,0}, {0,0,0 ,0}) Domain(0)	7991494,1.7453292519943 ,0,0,0,0,0,0 <mark>})</mark>
100%	基础 高级 模板 插件	25 setToolloInpu 26 setPlanConte	t(0, false) xt(sched.c	current threa	d id(), 1, "初始变量")
0.00%	0 8 9	27 u57fau5ea7 :	= {0,0,0,0,	,0,0}	

所以,如果在工程中添加的.lua程序的文件名与.pro工程相同的话,在某些情况下,很有可能 覆盖掉这个机器人实质要运行的脚本文件,导致工程无法运行。

4. 运行工程



5. SDK脚本编程

5.1 介绍

ARCS SDK中的SCRIPT模块可以发送本地脚本程序至机器人控制器,来完成对机器人的控制。SCRIPT模块的接口定义在script.h中,其中sendFile和sendString接口可以实现机器人运行本地脚本。



5.2 SDK脚本编程操作步骤

- 1. 在本地编写Lua脚本程序。
 - (1) 程序开头要导入aubo等库



(2) 程序结尾要添加启动插件

```
67 local app = {}
68
69
70 app.PRIORITY = 1000 -- set the app priority, which determines app execution order
71 app.VERSION = "0.1"
72 app.VENDOR = "Aubo Robotics"
73
74 function app:start(api)
75 ---
76 self.api = api
77 print("start-
78 p_Untitled_2()
                     被调用的函数
79 end
80
81 function app:robot_error_handler(name, err)
82 ---
83 print("An error hanppen to robot "..name)
84 end
85
86 -- return our app object
87 return app
```

(3) 之前在示教器脚本编写的test_movej.lua可以修改如下,保存为test_movej2.lua

```
\checkmark
 1 local aubo = require('aubo')
 2 local sched = sched or aubo.sched
 3 local math = aubo.math or math
 Δ
 5 local sleep = sched.sleep
 6 local aubo = require('aubo')
 7 local sched = sched or aubo.sched
 8 local math = aubo.math or math
 9
10 local sleep = sched.sleep
11 local thread = sched.thread
12 local sync = sched.sync
13 local run = sched.run
14 local kill = sched.kill
15 local halt = sched.halt
16
17 function p_movej()
       local ENV = sched.select robot(1)
18
       pi = 3.14159265358979323846
19
20
       -- 关节角, 单位: 弧度
21
       Waypoint_1_q = {-0.93/180*pi, -15.77/180*pi, 97.57/180*pi,
22
   23.33/180*pi, 89.99/180*pi, -1.82/180*pi}
```

```
Waypoint_2_q = {-16.67/180*pi, -21.25/180*pi, 56.19/180*pi,
23
   -12.57/180*pi, 89.99/180*pi, -17.56/180*pi}
24
       Waypoint_3_q = {30.22/180*pi, -14.97/180*pi, 61.42/180*pi,
   -13.62/180*pi, 89.99/180*pi, 29.33/180*pi}
25
26
       -- 设置运动速度比率
27
       setSpeedFraction(0.8)
28
       -- 以关节运动的方式依次经过3个路点: Waypoint_1、Waypoint_2和Waypoint_3
29
       moveJoint(Waypoint_1_q, 1, 1, 0, 0)
30
       -- 将信息打印到aubo control日志中
31
       textmsg("joint move to waypoint 1")
32
33
       moveJoint(Waypoint_2_q, 1, 1, 0, 0)
34
       textmsg("joint move to waypoint 2")
35
36
       moveJoint(Waypoint 3 q, 1, 1, 0, 0)
37
       textmsg("joint move to waypoint 3")
38
    end
39
40
41 local app = {}
42
43
44 app.PRIORITY = 1000 -- set the app priority, which determines app
   execution order
45 app.VERSION = "0.1"
46 app.VENDOR = "Aubo Robotics"
47
48 function app:start(api)
49
     _ _
50 self.api = api
51
    print("start---")
52
     p movei()
53 end
54
55 function app:robot_error_handler(name, err)
56
57
     print("An error hanppen to robot "..name)
<sup>58</sup> end
59
60 -- return our app object
61 return app
62
63
64
```

或者也可以直接用前面示教器脚本编程中的pro_movej.lua文件。

2. 用QtCreator 打开 example_script.cpp 示例,将test_movej2.lua或者pro_movej.lua拷贝 至虚拟机中。在运行代码前要确保机器人已上电,处于运行状态。或者在 example_script.cpp增加机器人上电的代码。

这里有两种方法去执行test_movej2.lua文件:

方法1:将lua文件放到程序的可执行目录中,即example/c++/build/bin文件夹中,运行 example_script.cpp代码之后,在终端输入test_movej2.lua即可



方法2:将lua文件放到非可执行目录中,例如桌面,那么运行example_script.cpp代码

之后,在终端输入test_movej2.lua的路径(/root/Desktop/test_movej2.lua)即可

3. 由于example_script.cpp示例中没有用到rpc服务,所以可以将代码中的相关部分注释掉 或去掉。

∆ 25 ▼ 26	<pre>int main(int argc, char **argv) {</pre>
27	#ifdef WIN32
28	// 将Windows控制台输出代码页设置为 UTF-8
29	SetConsoleOutputCP(CP_UTF8);
30	// auto rpc = std::make shared <rpcclient>():</rpcclient>
32	// // 接口调用: 连接到 RPC 服务
33	<pre>// rpc->connect(LOCAL_IP, 30004);</pre>
34	// // 接口调用: 登录
35	// rpc->login("aubo", "123456");
30	auto script = std::make shared(Script(lient)();
38	// 接口调田· 连接到 SCRIPT 服务
39	script->connect(LOCAL IP. 30002):
40	// 接口调用: 登录
41	<pre>script->login("aubo", "123456");</pre>
42	
43	// 输入脚本文件名
44	char file_name[20];
46	cm // rrce_name,
47	// 打开文件
48	ifstream file;
49	file.open(file_name);
50	// 如果打开文件失败,则退出程序
51 •	if (!file) {
52	cout << "open fail." << endl; exit(1):
54	}
55	
56	// 读取并打印脚本文件中的内容
57	<pre>std::string str_all;</pre>
58 ¥	while (!file.eof()) {
60	getline(file, str line);
61	<pre>str_all += str_line;</pre>
62	<pre>str_all += "\n";</pre>
63	$\begin{cases} \\ str all += \ \langle r \rangle a \langle r \rangle a \ , \end{cases}$
65	cout << "啣木内容・" << end] << str all << end] ·
66	file.close():
67	
68	<pre>// rpc->getRuntimeMachine()->abort();</pre>
70	// 友达脚本内容到控制器 script_>sendString(str all);
71	ser ipe-vsendser ing(ser_att);
72	// 增加阻塞来保证执行完脚本
73 🔻	while (1) {
74	<pre>std::this_thread::sleep_for(std::chrono::seconds(1));</pre>
75	ł
77	return 0:
78	}
79	-