# 程序说明文档

目录

[程序说明文档 1](#_Toc90157059)

[概述 1](#_Toc90157060)

[文件说明 1](#_Toc90157061)

[**1** **data目录** 1](#_Toc90157062)

[**2** **log目录** 2](#_Toc90157063)

[**3** **DSL可执行文件** 2](#_Toc90157064)

[**4** **autoInput自动测试程序** 2](#_Toc90157065)

[程序执行流程 3](#_Toc90157066)

[脚本解释流程 4](#_Toc90157067)

[脚本设计说明 5](#_Toc90157068)

[**1** **状态（Step）** 5](#_Toc90157069)

[**2** **执行指令** 6](#_Toc90157070)

[**3** **跳转指令** 7](#_Toc90157071)

[程序使用说明 9](#_Toc90157072)

[脚本示例 10](#_Toc90157073)

[脚本测试样例 13](#_Toc90157074)

[日志与调试 14](#_Toc90157075)

## 概述

本程序定义一个领域特定脚本语言，这个语言能够描述在线客服机器人的自动应答逻辑，并设计实现了一个解释器解释执行这个脚本，可以根据用户的不同输入，根据脚本的逻辑设计给出相应的应答。

面向用户为脚本编程人员，为编程人员提供了一种较为简单的编程语言，用于设计机器人应答脚本。

## 文件说明

**程序包含四个文件：data目录、log目录、DSL.exe可执行文件、autoInput自动测试程序。**

1. **data目录**

该文件目录下包含grammar.txt, proc.txt, user.txt， input.txt三个文本文件。

* 1. **grammar.txt**

该文件为文法文件，用于检查脚本的语法错误。

不能修改或删除，否则会导致解释器程序无法检查语法，不能正常运行。

* 1. **proc.txt**

该文件为编程人员设计的脚本文件，编程人员设计好脚本后，放到此目录下并命名为proc.txt，执行DSL.exe即可解释执行该脚本。或者放在任意位置，执行解释器时由编程人员输入脚本的位置。

* 1. **user.txt**

用户数据文件，存放编程人员面向的用户的用户数据。比如你需要设计一个联通客服机器人应答脚本，则该文件下存放的数据为联通用户的相关数据。存储和编写方式会在下面的脚本设计说明中具体指出。

* 1. **input.txt**

该文件存放脚本测试样例，由程序员自己编写，交由自动检测程序执行，检查脚本逻辑是否正确。

1. **log目录**

该文件目录下包含syntaxLog.txt, interpretLog.txt， errors.txt两个文本文件。

* 1. **syntaxLog.txt**

该文件为具体的语法检测调试文件，若你了解一些编译原理的语法分析知识，可以在脚本语法出现错误时在这里找到具体语法分析步骤，帮助你了解出错具体原因。使用方式会在日志与调试中具体指出。

* 1. **interpretLog.txt**

该文件记录与用户交互时每一个过程的执行报告。包含用户数据变化的具体情况。

* 1. **errors.txt**

该文件存放简略的错误信息。只会在脚本解释出错时生成。

1. **DSL可执行文件**

该程序会自动解释脚本文件，若脚本没有错误，则会进入与用户的交互界面。

1. **autoInput自动测试程序**

该程序可自动打开并执行DSL.exe文件，并自动输入提前准备好的测试语句（data目录下的input.txt），供编程人员测试脚本功能。

## 程序执行流程

读取用户数据

读取脚本文件

语法分析

构建脚本语法树

当前状态

下一状态

用户反馈

解释执行

逐行分析

词法分析

## 脚本解释流程

Start

express

Speak？

N

Y

Listen？

empty？

Default Proc

get answer

Y

N

Silence？

warning

N

Silence Proc

Y

Y

brunch？

Brunch Proc

Y

N

N

Compare？

Compare

Right Proc

Wrong Proc

True

False

Default？

Y

N

Default Proc

warning

Y

N

Default？

Y

Back？

Exit？

Y

Y

N

N

N

Exit

注：’?’表示判断指令是否存在（除了empty是判断answer是否为空）。图中未展示Compute、Add、Delete三个必定执行的执行指令。

## 脚本设计说明

**词法上，脚本文件由关键字、变量名（id）、数字（num）、字符串（str）、符号组成。**

**关键字：**Start Step Listen Default Speak Silence Brunch Compare Compute Wrong

Right Exit Back Add Delete

用于定义一个状态或者命令，只能出现在每一行的开头。为了脚本的可读性，不建议关键字与变量名同名。

**变量名：**由下划线、数字、字母组成的，以下划线或字母开头的任意字符串。

在变量名前需要加上’&’符号以标识该字符串为一个变量名。下面以id表示一

个变量名。

**数字：**由小数点、数字字符、指数字母（e或E）、正负号组成。数字部分应以小数点或数字

字符开头，下面以num表示一个数字。

**字符串：**用双引号包含的字符串。下面以str表示一个字符串。

**符号：**+ - , &

除了上述说明的符号串外，其他任何字符都会导致报错。

**语法上，脚本由多个状态（Step）组成，每一个状态可包含多条指令。**

指令包括：

**执行指令：**Speak，Listen，Compare，Compute，Add，Delete。

**跳转指令：**Brunch，Default，Silence，Wrong，Right等条件跳转指令；

Back，Exit等无条件跳转指令。

由上面程序执行流程给出的流程图可以看出：程序只是逐行检查脚本文件，不关心各行之间的相互逻辑联系。这意味着：在特定区域内（即每个状态内，以及每个状态之间），编程人员可以不用考虑每行语句（或每个状态）的顺序。每一行即是一个完整的指令。程序只会判断该状态是否包含某个指令，故每个状态相同指令只能有一个，若有多个，则会以最后一个作为执行目标。

1. **状态（Step）**

状态分为普通状态（Step），开始状态（Start）。脚本必须指定开始状态，且保证其唯一性。

定义状态：Step id

定义开始状态：Start id

id即状态名。注意：定义在该Step下的所有指令都必须写在该定义语句之后，直到文件末尾或者遇到另一个状态定义语句。

例：Step welcome 表示定义一个名为welcome的状态

1. **执行指令**

解释器会判断每一条执行指令是否存在，然后执行存在的执行指令（Compare指令可能不会执行）。虽然编写指令的顺序没有要求，但指令执行的顺序是已经规定好的。比如你可以先写Listen再写Speak，但执行时会总是先执行Speak再执行Listen。详细执行流程可见脚本解释流程图。

* 1. **Speak**

格式：Speak str

其中str可以只是由双引号括起来的字符串，也可以是字符串和变量通过’+’号连接组合。在解释执行字符串和变量的组合时，会将变量替换为对应的字符串，再与其他字符串拼接，最后输出。

例: Speak “你本月账单为”+ &bill +“元。”

假变量bill存储的值为10.00，则最后输出的语句是：你本月账单为10.00元。

* 1. **Listen**

格式：Listen num1,num2

其中num1表示至少聆听num1秒， num2表示至多聆听num2秒。

用户的回答以回车结束。并且一旦检测到回车符号，则终止Listen，无论是否聆听了num2秒。若到了num2秒，用户仍没有输入回车，则会读入缓冲区里的所有内容作为用户的answer。

例：Listen 5,20

* 1. **Compute**

格式：Compute &id + num 或者 Compute &id + &id

该指令为数值计算指令，结果会存储在第一个变量（id）中。

例：Compute &balance + 10.00

假设当前balance为5.00，则执行完后，balance值为15.00（5.00+10.00）

* 1. **Compare**

格式：Compare &id

隐含参数为用户给出的answer。

若变量id指向的数据为单值数据，则判断answer是否等于id指向的值。

若变量id指向的数据为多值数据，则判断id指向的值中是否存在answer。

通常，Compare语句之后会接上Right和Wrong两个条件跳转指令。通过Compare返回的真假值，进入相应的命令。

注意:Compare指令不会每次都执行，在设定有Compare指令的状态中，用户的answer通常不会是固定的、可以匹配的字符串，但当用户answer成功匹配设定的Brunch时，状态会根据Brunch跳转，而不会执行Compare指令，例如用户输入“退出”，若Brunch可以匹配“退出”，则程序应当直接跳转到退出程序而不是进行Compare。只有Brunch匹配失败时，解释器才会考虑用户的answer可能需要进行比较。详细信息可见脚本解释流程图以及Brunch指令说明。

例1：Compare &password

若此时用户answer为111111，而password值为111111，则answer等于password，返回true。

例2：Compare &business

若此时用户answer为“语音通话”，business为“语音通话|套餐A|套餐B|”（多值数据每一个值后都要有‘|’）answer在business中，结果返回true。

* 1. **Add**

格式：Add &id

变量id必须指向多值数据。功能为将用户answer加入到id指向的数据中。

要注意此状态下answer是否被改变、是否是可以正确添加的值。

例：Add &business

若此时answer为“语言通话”， business为“套餐A|”，则执行Add指令后，business值为“套餐A|语音通话|”。

* 1. **Delete**

格式 Delete &id

变量id必须指向多值数据。功能为将用户answer从id指向的数据中删除。

要注意此状态下answer是否被改变、是否是可以正确删除的值。

例：Delete &business

若此时answer为“语言通话”， business为“套餐A|语音通话|”，则执行Delete指令后，business值为“套餐A|”。

1. **跳转指令**

在每个Step中，跳转指令只会执行一次，执行完毕后则立刻跳转到相应的状态。

* 1. **条件跳转。**

条件跳转是根据用户的反馈做出相应的跳转的指令。

* + 1. **Silence**

格式：Silence id

若用户输入为空，则会跳转到状态名为id的状态下。

例：Silence silenceProc

* + 1. **Brunch**

格式：Brunch str,id

str为需要匹配的字符串，id为匹配成功后跳转到的状态的状态名。

Brunch可以有多个，但只能匹配一个，也即：状态只跳转一次。

例：Brunch “余额”,balanceProc

Brunch “账单”,billProc

若用户输入“我想要查询余额是多少”,提取到关键词“余额”后，即可跳转到balanceProc。

若用户输入“余额账单”，虽然两个关键词都可以匹配，但程序只会跳转到balanceProc和billProc中的一个，具体哪一个由C++中set容器内部实现决定。

* + 1. **Default**

格式：Default id

当answer不为空而又无法匹配Brunch，同时又没有设置Compare指令时，则会转入名为id的状态。

注：若当前Step未设置Listen指令，则Default可以看作无条件跳转指令。

例：Default defaultProc

* + 1. **Right**

格式：Right id

此指令只能搭配Compare指令使用，否则无效。若Compare返回为True，则跳转到id对应的状态。

例：Right addBusiness

* + 1. **Wrong**

格式：Wrong id

此指令只能搭配Compare指令使用，否则无效。若Compare返回为False，则跳转到id对应的状态。

例：Wrong addWrong

* 1. **无条件跳转。**

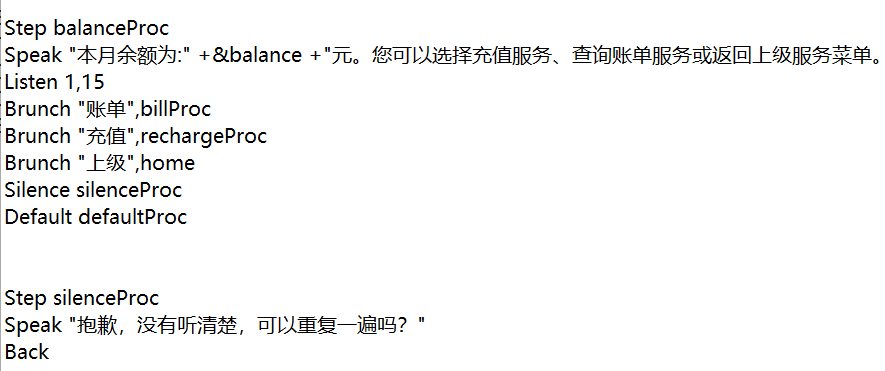
无条件跳转指令有Back和Exit，其中Back优先级比Exit高。

无条件跳转指令的优先级都比条件跳转指令低，当条件跳转指令都不满足跳转条件时，才会考虑无条件跳转指令。

* + 1. **Back**

Back指令会返回上一个状态，并跳过上一个状态的Speak。

例：



当执行完balanceProc中的Speak语句过后，若用户沉默，则跳转到

silenceProc，Speak ：“抱歉，没有听清楚，可以重复一遍吗？”，Back

指令返回到balanceProc后，可以直接执行Listen。若你仍想从头开始

执行，可以把Back更改为Default balanceProc。

* + 1. **Exit**

明确地说是退出指令，脚本中应至少有一个Step包含Exit。另外，Exit优先级最低，故确保至少有一个Step中没有除Exit以外的其他跳转指令。执行Exit后，脚本运行结束。

## 程序使用说明

程序并没有提供输入变量的接口，故脚本中出现的所有变量名只能为user.txt中已有的变量名。为了便于引导用户，程序额外提供了一个recommend变量。使用方法与普通变量一致，需要在引用前加上&符号。该变量对应的值为上一状态的所有brunch分支。

例如：

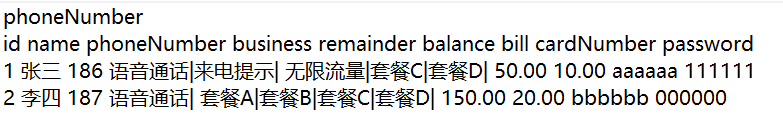
1. Step defaultProc
2. Speak "非常抱歉，暂时未开放此业务。您可以对我说" + &recommend
3. Back

如果是下示Step调用了defaultProc，则程序会输出：

“非常抱歉，暂时未开放此业务。您可以对我说 账单 投诉 余额 业务列表 套餐列表 充值 退出”

1. Step home
2. Speak "请问有什么可以帮您?"
3. Listen 1,15
4. Brunch "账单",billProc
5. Brunch "投诉",complainProc
6. Brunch "余额",balanceProc
7. Brunch "业务列表",businessList
8. Brunch "套餐列表",userBusiness
9. Brunch "充值",rechargeProc
10. Brunch "退出",thanksProc
11. Silence silenceProc
12. Default defaultProc
13. **填写用户数据user.txt：**
    1. 第一行为一个变量名，作为索引名。
    2. 第二行为n个变量名，表示用户用有的n的属性。每个变量名用一个空格分开
    3. 其余行为用户n个属性的具体值，需要与第二行的变量名一一对应，且均不可为空，同样用一个空格隔开。
    4. 第一行的变量名（索引名）必须存在于第二行的变量名中，当用户接入时，需要输入自己索引名对应的值，程序会通过该值找到用户的所有数据。
    5. 多值属性每一个值后面需要加上’|’符号。

示例：



用户共有9个属性，当用户接入自动应答系统后，需要输入自己的phoneNumber值。

1. **编写脚本文件proc.txt**

根据脚本设计说明，编写脚本。放入data目录下。

1. **运行脚本解释器DSL.exe**

运行后，程序会报告脚本存在的词法或语法错误，编程人员根据错误提示修改脚本，直到无错误报告。

成功运行后，会要求用户输入索引变量对应的值，程序在user.txt中找到该用户的数据后，对话才正式开始。

1. **运行自动测试程序autoInput.exe**

提前编写好用户的回答逻辑，存储在input.txt文件里，放在data目录下，注意编码格式为ANSI。运行autoInput.exe自动测试程序，原理通过鼠标位置定位到输入框，故请保持运行时鼠标位于命令行窗口内。

## 脚本示例

user.txt

1. phoneNumber
2. id name phoneNumber business remainder balance bill cardNumber password
3. 1 张三 186 语音通话|来电提示| 无限流量|套餐C|套餐D| 50.00 10.00 aaaaaa 111111
4. 2 李四 187 语音通话| 套餐A|套餐B|套餐C|套餐D| 150.00 20.00 bbbbbb 000000

proc.txt

1. Start welcome
2. Speak "尊敬的" + &name + ",你好!"
3. Default home
5. Step home
6. Speak "请问有什么可以帮你?"
7. Listen 1,15
8. Brunch "账单",billProc
9. Brunch "投诉",complainProc
10. Brunch "余额",balanceProc
11. Brunch "业务列表",businessList
12. Brunch "套餐列表",userBusiness
13. Brunch "充值",rechargeProc
14. Brunch "退出",thanksProc
15. Silence silenceProc
16. Default defaultProc
18. Step billProc
19. Speak "你本月账单是" + &bill +"元。请问你还有什么需求？"
20. Back
22. Step complainProc
23. Speak "你的意见是我们改进工作的动力，请问你有什么补充？"
24. Listen 1,15
25. Default thanksProc
27. Step balanceProc
28. Speak "本月余额为:" +&balance +"元。你可以选择充值服务、查询账单服务或返回上级服务菜单。"
29. Listen 1,15
30. Brunch "账单",billProc
31. Brunch "充值",rechargeProc
32. Brunch "上级",home
33. Silence silenceProc
34. Default defaultProc

37. Step silenceProc
38. Speak "抱歉，没有听清楚，可以重复一遍吗？"
39. Back
41. Step defaultProc
42. Speak "非常抱歉，暂时未开放此业务。你可以对我说" + &recommend
43. Back
45. Step thanksProc
46. Speak "感谢你的来电，祝你生活愉快。"
47. Exit
49. Step businessList
50. Speak "你可以对我说" + &recommend
51. Back
53. Step rechargeProc
54. Speak "目前支持充值卡充值，请输入充值卡卡号，或对我说返回上级菜单"
55. Listen 1,15
56. Brunch "上级",home
57. Compare &cardNumber
58. Right inputPsw
59. Wrong invalid
60. Silence silenceProc
61. Default invalid
63. Step invalid
64. Speak "格式错误或输入有误，请重新输入。"
65. Back
67. Step inputPsw
68. Speak "请输入充值卡密码"
69. Listen 1,15
70. Compare &password
71. Right success
72. Wrong wrongPwd
73. Default invalid
75. Step success
76. Speak "充值成功！"
77. Compute &balance , 50
78. Default home
80. Step wrongPwd
81. Speak "密码错误！，请重新输入。"
82. Back
84. Step userBusiness
85. Speak "你办理的套餐有：" +&business +"你可以取消已有套餐、办理新套餐或返回上级菜单"
86. Listen 1,15
87. Brunch "取消",cancelBusiness
88. Brunch "办理", addBusiness
89. Brunch "上级",home
90. Silence silenceProc
91. Default defaultProc
93. Step cancelBusiness
94. Speak "请说出想要取消的套餐。返回上级请说’上级‘"
95. Listen 1,15
96. Brunch "套餐列表",userBusiness
97. Brunch "上级",home
98. Compare &business
99. Right cancelSuccess
100. Wrong cancelWrong
101. Silence silenceProc
102. Default cancelBusiness
104. Step cancelSuccess
105. Speak "取消套餐成功。即月起生效"
106. Delete &business
107. Add &remainder
108. Default home
110. Step cancelWrong
111. Speak "套餐不存在。你可以对我说”套餐列表“获取已有套餐名称"
112. Back
114. Step addBusiness
115. Speak "请说出想要添加的套餐,或者对我说'套餐列表'获取可以添加的套餐，返回上级请说‘上级’"
116. Listen 1,15
117. Brunch "套餐列表",allBusiness
118. Brunch "上级",home
119. Compare &remainder
120. Right addSuccess
121. Wrong addWrong
122. Silence silenceProc
123. Default addBusiness
125. Step addSuccess
126. Speak "添加套餐成功，即月起生效"
127. Add &business
128. Delete &remainder
129. Default home
131. Step addWrong
132. Speak "套餐不存在。"
133. Default addBusiness
135. Step allBusiness
136. Speak "你还可以办理：" + &remainder
137. Back

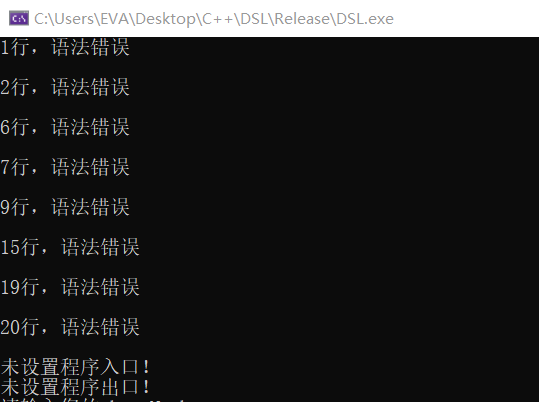
## 脚本测试样例

1. 186
2. 查看业务列表
3. 我要充值
4. bbbbbb
5. aaaaaa
6. 222222
7. 111111
8. 我要干什么
9. 查看套餐列表
10. 返回
11. 我要办理套餐
12. A1
13. 查看可添加的套餐列表
14. 套餐C
15. 我的套餐列表
16. 办理
17. 查看套餐列表
18. 返回上级菜单
19. 啊啊啊啊啊啊
20. 余额
21. 查询账单
22. 我要充值
23. 有有有有有有
24. 返回上级菜单
25. 套餐列表
26. 取消套餐C
27. 套餐列表
28. 取消套餐
29. 套餐C
30. 套餐列表
31. 办理
32. 套餐列表
33. 返回上级
34. 我要投诉你
35. 你是个机器人

## 日志与调试

1. **解释脚本时出现错误**

程序会出现以下提示：

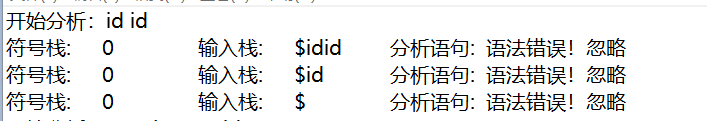


根据行号，可以定位到错误的位置，再阅读本文档的脚本设计说明部分查找原因。

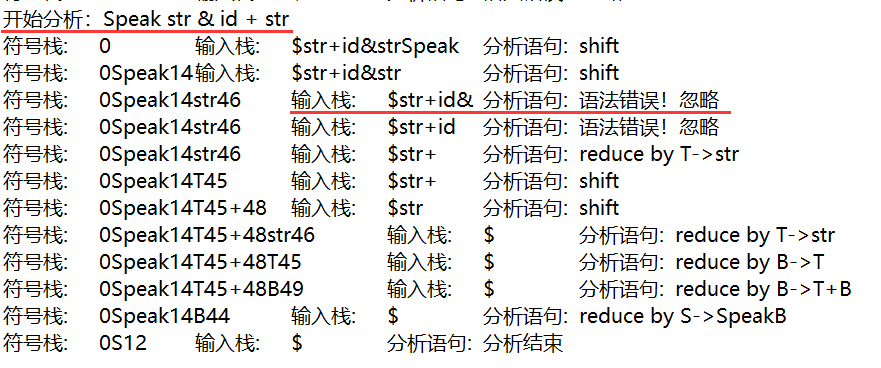
如果仍查找不出错误原因，若你了解编译原理的文法相关知识，可以直接查看grammar.txt内定义的文法。查看错误语句是否是能由该文法产生。

否则，你可以直接阅读syntaxLog.txt文件。

如下图：



分析的语句是 id id，表明该行只有两个变量名，可能是关键字拼写错误导致误认为变量名。



分析语句为：Speak str & id + str

找到第一处语法错误，此处输入栈为：$str+id& 表示分析到&符号时出现错误。

也即str后面不应该为&符号，而字符串和变量拼接应该用＋号连接。

1. **执行脚本时不合逻辑**

可以查看interpretLog.txt查看执行情况。

