> 共享内存工具是实习的第一个需求，以下将会简述该小工具的相关技术细节
> https://github.com/cnsunshine/share_mem_analyse

1. 需求和功能

1.1. 需求

导师提出的需求是希望通过小工具连接到共享内存，查看到共享内存中的数据，如blob里面的数据，方便进行问题定位和查看。

1.2. 设计

导师没有限定需求的详细功能，收到需求之后，我自己首先做了简单的分析。梳理后，我觉得这个小工具至少应当实现以下几个功能

- 可以定义要查询的数据结构
- 可以根据定义的结构体里的某一个成员的内容进行搜索
- 可以支持项目已知的基础数据类型，并且应当方便进行扩展
- 可以将查询的数据打印到屏幕或输出到文件

1.3. 实现的功能

在经过两次版本迭代后，目前小工具实现了以下的功能

- 可以在xml定义结构体进行查询，结构可以递归
- 可以在xml中以block标记屏蔽不需要查询的内容
- 可以多条件复合查询
- 可以查询数组结构，数组可以递归

- 可以输出到文件
- 使用类SQL语句进行解析，方便后续扩展，如进行模糊匹配

通过对项目中常用类型的定义，暂时对以下基础结构进行了支持

```
int,
char,
char*(string), //该结构在使用时应当注意，因为'\0'禁止使用char*字符串类型验证char[]数组
char[],
uint8_t,
uint16_t,
uint32_t,
uint64_t,
int8_t,
int16_t,
int32_t,
int64_t,
unsigned char,
short,
float,
double,
```

2. 实现细节

2.1. 工具结构

2.1.1. ShareMemoryAnalyse

ShareMemoryAnalyse对象负责管理和调用各对象，完成共享内存搜索的整个操作。

2.1.2. XmlLoader

XmlLoader根据config/Config.h里定义的xml文件路径加载用户定义的数据结构，进行结构读入、结构树生成、偏移计算等操作。

2.1.3. SqlAnalyse

SqlAnalyse对用户输入的类SQL语句进行词法分析、语法分析，并生成和展开查询条件。

2.1.4. MemManager

MemManager将使用XmlLoader和SqlAnalyse生成的结果进行内存查询和展示、输出到文本的工作。

2.2. 实现方法

2.2.1. 结构树生成

1. `loadAllStruct()` 将xml中所有结构加载到structList中，其中structList[0]项（定义在xml中的第一个）是我们要在内存中查看的数据主struct，后面项是主struct中出现的自定义struct的定义。借用编译原理中中

间代码生成的思想，会将所有等待回填的结构使用pBackpatchChain链串起。
2. backpatchType() 回填structList的类型，如果是已实现的基础类型，则直接设置type值，如果不是基础类型，则自动**递增**设置type类型，此处类似编译中的生成变量名表的思想。
3. backpatchSize() **循环**遍历回填structList的尺寸，回填尺寸总是先从自定义数据结构中完全已知的数据类型开始回填，每次循环都是从可计算或已计算的结构往调用其结构的结构体上升。每次回填必定会回填成功一个结构体，如果回填失败，则说明有结构体中包含了不可识别的基础类型。
4. createIndex() 内存中相等判断需要获取结构体某个成员的起始位置，通过强制转换进行比较。因为叶子节点是基础类型，比较的时候需要对叶子节点进行比较，这里将会通过队列下降到树的叶子节点计算所有结构体成员的起始位置。(ps:这也是char*作为string来判断，char[]字符数组不可以这样判断的原因)
5. 其他辅助操作/函数 如挂载结构过程需要进行深拷贝，及时释放不使用的heap上内存。

2.2.2. 类SQL语句解析

1. analyseTextStep1()词法分析 通过该过程对输入语句进行词法分析
2. analyseTextStep2()语法分析 对词法分析结果进行语法分析。借用了递归下降和词法分析思想，采用循环下降的方法，避免stack溢出。
3. expandQueryCondition() 该函数在v1.0没有实现数组查询的未使用，增加了数组功能后，查询数组相当于将数组当成有数组成员个数个结构，对其进行展开，设置一个queryId作为一组查询标识。
4. 其他辅助操作/函数 大量使用了heap上数据，需要在查询后进行释放，不影响后续查询。重载了一些基础的函数。

2.2.3. 内存查询和匹配

1. linkMem() 连入共享内存，设置好查询的起始位置。
2. setPrintNode() 设置打印节点，这里为了方便，暂时只打印叶子节点(基础数据类型节点)，后续扩展可以在这里进行更新。
3. search() 搜索，通过输入数据的读入转换，和内存中固定位置的数据进行比较判断相等。
4. print() 打印到控制台或文件。
5. 其他辅助操作/函数 通过传入lh,rh,type进行判断，这里有涉及到用户输入的转换问题，例如文本向整数转换，对于不能直接转换的类型采用先转换到高精度，再强制转换到需要类型的方式。

2.3. 示例

2.3.1. xml定义

```
<?xml version="1.0" encoding="utf-8"?>
<root>
    <!-- 内存块结构定义 -->
    <struct name="RoleInfo">
        <!-- block 类型用来标记跳过 -->
        <entry name="block" type="block" size="0"/>
        <entry name="age" type="int"/>
        <entry name="ch" type="char"/>
        <!-- string 类型必须指定长度 -->
        <entry name="str" size="100" type="string"/>
        <entry name="student" type="StudentInfo"/>
        <entry name="location" type="UserLocation"/>
    </struct>
```

```
    <!--自定义数据结构声明-->
    <struct name="StudentInfo">
        <entry name="id" type="int"/>
    </struct>
    <struct name="UserLocation">
        <entry name="loc" type="CommonLocation"/>
        <entry name="description" type="string" size="255"/>
    </struct>
    <struct name="CommonLocation">
        <entry name="country" type="string" size="50"/>
        <entry name="province" type="string" size="20"/>
    </struct>
</root>
```

2.3.2. 操作

```
input shmid: 3375144
start load xml config for data structure.
(CommonLocation)[70]
-(province)[20]
-(country)[50]
(UserLocation)[325]
-(description)[255]
-(loc)[70]
--(province)[20]
--(country)[50]
(StudentInfo)[4]
-(id)[4]
(RoleInfo)[434]
-(location)[325]
--(description)[255]
--(loc)[70]
---(province)[20]
---(country)[50]
-(student)[4]
--(id)[4]
-(str)[100]
-(ch)[1]
-(age)[4]
-(block)[0]
input mem offset(查询的数据段的首条数据在内存中的起始位置): 0
sql>select * from all where RoleInfo.age = 5555
age: 5555
ch: d
str: dgefe
id: 1333
description: ged
country: chin
province: gaada
```

2.4. 总结

在这次小工具开发基本经历了三个阶段，一是前期需求分析和技术预演后和导师确定方向正确与否，二是初版v1的编码，请导师验收，三是v2对部分功能进行更正和增加。
这次小工具开发过程中也遇到了一个问题，即在获取共享内存信息时，Unix采用拷贝信息到指针指向的空间，但使用了pack(1)后会因为字节对齐的问题导致拷贝进来的值读出后有误。这个小问题让我更加意识到在使用系统接口时要尽可能了解其实现手段，不能盲目调用。 本次项目代码1500+行，实际编码时间<7天，编码速度较慢，主要原因在在于结构树加载和数组实现耗费了一定时间。 本次项目代码目前尚有较多不够优雅的地方，仍然需要进行更好的设计，功能上如string类型需要进行更好的处理，编码上应当避免循环的嵌套，输出到文件更改成只打开成一次等。