



同濟大學
TONGJI UNIVERSITY

大型主机系统管理虚拟实验平台-DFSMS实验

主机系统管理

Version 1.0

2021.6

小组成员：

1753402 谢康，174133 黄金鑫，1754188 谢尚汝

所在院系： 软件学院

学科专业： 软件工程

指导老师： 高珍

1、项目简介

本项目是在学长所做的主机实验仿真平台上进行，我们主要负责的是存储管理系统DFSMS实验的相关内容。这个实验在本课程中已经有过相关介绍，在这里需要我们将其在仿真平台上进行相关功能的实现。

因此，本小组的主要工作就是，首先回顾DFSMS存储管理实验的相关内容，然后在仿真平台上，主要对于实验三（创建一个简单的SMS环境实验），按照上面展示的实验步骤一步步进行实验，然后总结出仿真平台缺少的一些功能，再根据本组成员对于实验的理解情况，选择其中的部分功能进行接口实现，并相应添加新的界面，以此来完善该仿真平台的功能。

2、实验简介

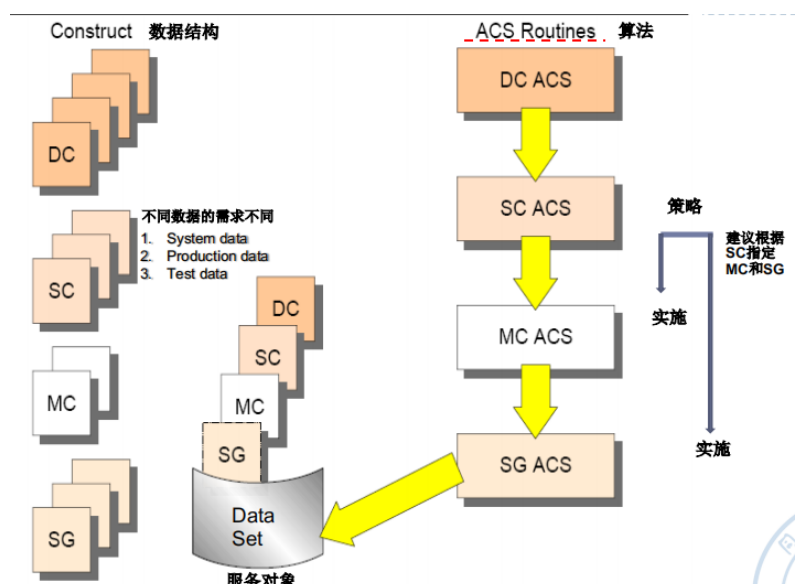
本小组主要研究的是DFSMS目录下的实验三，这个实验也就对应之前的作业9，主要的实验目的就是构建一个简单的DFSMS配置。下面对实验步骤进行简单介绍。

首先，创建SCDS(Source Control Dataset)，作为控制数据集，之后创建的各种Construct都会放在控制数据集中；然后为SCDS创建一个基本的配置，填写部分参数，比如Default Unit,Default Device Geometry等。

接下来，就是对四种Construct(DC,SC,MC,SG)的创建了。首先是创建Data Class，这个数据结构用来存储创建数据集的相关参数，比如数据集的类型。本实验需要创建三个Data Class，分别对应着创建数据集的三种类型：顺序数据集、分区数据集和KSDS数据集，创建所需的参数在手册上已经全部给出；然后是创建Storage Class(存储数据集的性能需求、可用性等标签)、Management Class(存储各种策略，比如指定备份次数，过期时间等)、Storage Group(存储Volume)。

然后，在创建的Storage Group中添加一个或多个DASD卷，下面就是创建ACS Routine了。

ACS Routine是算法，因为四种数据结构中，每一个都有很多种，ACS Routine的任务就是根据代码逻辑及判断条件，在每一种数据结构中选择一个分配给创建的数据集，且严格按照一定的顺序执行，如下图所示：



本实验手册中，对于ACS Routine中的每一个成员的创建都给出了相应的JCL代码，然而只是源码，无法直接执行，需要之后的Translate操作，将ACS Routine 源代码翻译为目标代码(可执行码)，这一目标代码也会存放在控制数据集中。

在完成四个ACS成员的Translate之后，需要对ACS Routine及其所用到的Construct进行验证，确认是否成功。

最后是ACS Routine的测试，分别用三个测试用例，测试三种类型的数据集的创建，得到相应的测试结果，以上便是整个实验的大致过程。

3、仿真平台功能研究

在基本了解了本实验相关内容及原理的基础之上，本小组成员开始对仿真平台上的存储管理的相关功能进行熟悉与测试，在此期间，我们也与学长进行了沟通，然后对目前缺失的功能进行了统计。

首先，在DFSMS实验这一块，与其他实验模块最大的不同就是每个实验都有一个模拟的ISPF窗口，而其他实验通常只是一个JCL代码提交框。模拟的ISPF窗口与主机窗口较为相似，在ISMP菜单下，有各个选项可供选择，这里我们对3、4、5、6这四个选项进行了研究，这些选项分别对应四种Construct的相关操作，下图为Management Class的界面：

MANAGEMENT CLASS APPLICATION SELECTION

To perform Management Class Operations, Specify:

CDS Name

Management Class Name

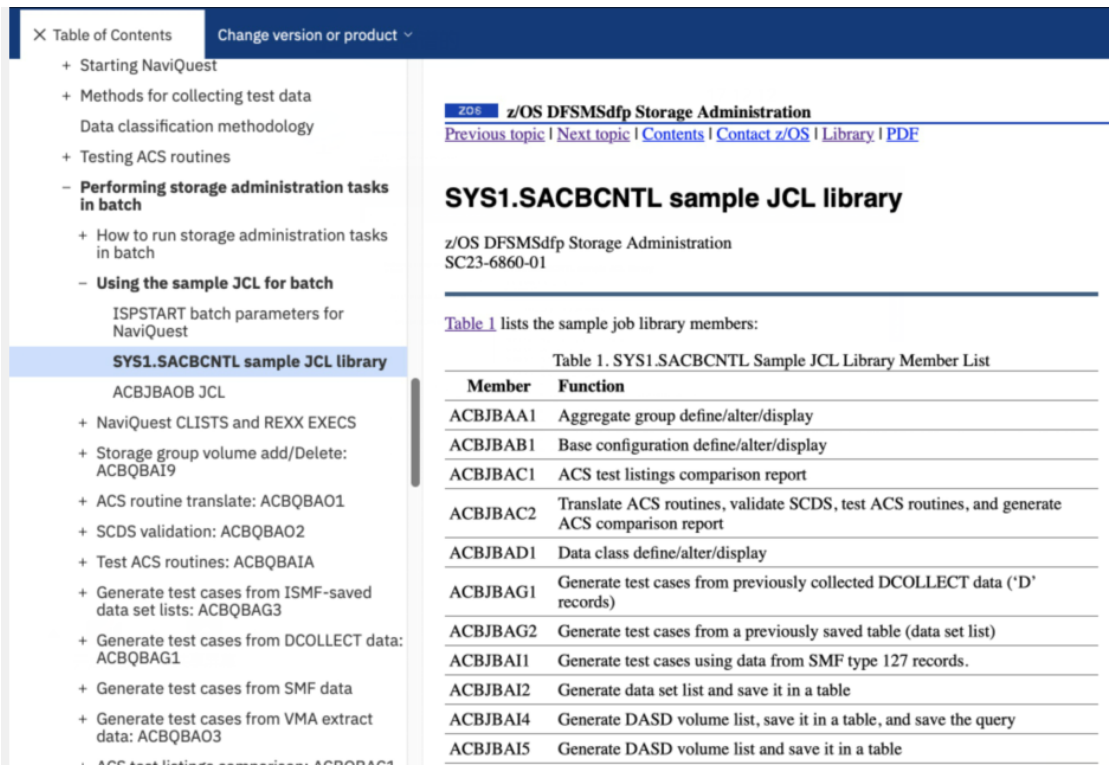
Select one of the following options: Only support option 3

| | | |
|---|---------|---|
| 1 | List | - Generate a list of Management Classes |
| 2 | Display | - Display a Management Classes |
| 3 | Define | - Define a Management Classes |
| 4 | Alter | - Alter a Management Classes |

其他几个的界面组成与此基本类似，在实际的使用中我们发现，只有3选项，也就是定义相关的数据结构是可以正常使用的，其他功能都还未开发出对应的接口及前端界面。另外，在ISMP的主菜单下，其他的几个选项也都有部分尚未实现的情况，但由于各方面的原因，本小组决定只对四个数据结构的部分进行功能的完善。

4、功能完善过程与难点介绍

4.1 后端接口实现



经过以上的研究，我们共总结出了需要实现的接口12个，分别是四种数据结构(DC,SC,MC,SG)的alter,display以及list。接着，本小组的成员开始仔细阅读后端相关源码，并总结出相应的规律，参照学长的做法来编写新的接口并实现相关功能。

同时，我们参照了IBM官网中的样例JCL代码列表以及对应的功能（如上图），然后在主机服务器的对应位置找到相关JCL源码，并将其整合到项目的源码中。同时，我们也在该网站参考了相关功能编写的教程，并将所学的东西运用到了项目的编码之中。

由于在我们实现的12个接口中，同一操作类型的接口的实现逻辑基本是类似的，因此这里将每一种操作类型选出一个接口，介绍它们的实现，并简要叙述本小组在实现过程中遇到的各种困难。

4.1.1 Display Management Class接口实现

该接口的实现应该算是本小组所遇到的第一个难点。在之前，Data Class以及Storage Class的display已经被我们成功实现了，所使用的方法主要是参考学长在SmsService文件及SmsController文件中的相关功能的写法，再结合自己要实现的功能的实际情况进行编写。JCL的参考代码已经被下载到了本地，我们需要参照代码进行相应的修改，将逻辑完成。

之前的display参照以上的写法都成功实现了功能，但是到了Management Class的display时却出了问题，导致返回的结果为空。后端返回400 Bad Request。我们在对错误进

行初步分析后，认定问题出在jcl代码上。但在经过几次改动后仍然没有任何效果，之后，我们又对比了主机上的jcl代码与自己写在函数中的代码，有个MGMTCLAS()这一行，样例JCL中括号里没有任何内容，但我们经过研究，注意到这里应当填上输入的management class，才能告诉主机需要展示的management class是哪一个。在修改完成后，我们解决了相关的问题。这一部分的主要代码逻辑如下所示：

```
public String displayManagementClass(HttpSession session, ManagementClass managementClass) {
    if (prepareTable2(session)) {
        String uid = session.getAttribute("S: \"ZOSMF_Account\").toString();
        // add quotes to avoid prefix
        managementClass.setScds("'" + managementClass.getScds() + "'");
        String jcl = getHead(uid) +
            "//STEP1 EXEC ACBJBAOB,\n" +
            "//      TABL2=" + uid + ".TEST.ISPTABL\n" +
            "//SYSUDUMP DD SYSOUT=* \n" +
            "//SYSTSIN DD *\n" +
            "PROFILE PREFIX(IBMUSER)\n" +
            "ISPSTART CMD(ACBQBAJ1 DISPLAY +\n" +
            "SCDS(" + managementClass.getScds() + ") +\n" +
            "MGMTCLAS(" + managementClass.getMgmtclas() + ") +\n" +
            ")\n" +
            "/*";
        return js.submitJCL(session, jcl, id: 104);
    }
    return "";
}
```

可以看到，主题就是一个jcl字符串的拼写，至于之后的字符串处理及jcl提交学长都已经写好，可以直接拿来使用。可以看到，display一个management class需要输入它对应的控制数据集的名字，以及management class本身的名字。

4.1.2 Alter Storage Group接口实现

该接口的作用是对之前已经创建的Storage Group的相关属性进行修改。它的jcl代码与define十分相似，具体代码逻辑如下所示：

```
public String alterPoolStorageGroup(HttpSession session, PoolStorageGroup poolStorageGroup) {
    if (prepareTable2(session)) {
        String uid = session.getAttribute("S: \"ZOSMF_Account\").toString();
        poolStorageGroup.setScds("'" + poolStorageGroup.getScds() + "'");
        String jcl = getHead(uid) +
            "//STEP1 EXEC ACBJBAOB,\n" +
            "//      TABL2=" + uid + ".TEST.ISPTABL\n" +
            "//SYSUDUMP DD SYSOUT=* \n" +
            "//SYSTSIN DD *\n" +
            "PROFILE NOPREFIX\n" +
            "ISPSTART CMD(ACBQBAJ2 ALTER +\n" +
            fieldsResolver(poolStorageGroup) +
            ")\n" +
            "/*\n" +
            "//TEMPFILE DD DSN=&&TEMPFILE,DISP=(MOD,PASS),\n" +
            "// SPACE=(TRK,(1,1)),LRECL=300,RECFM=F,BLKSIZE=300\n" +
            "//STEP2 EXEC ACBJBAOB,\n" +
            "//      TABL2=" + uid + ".TEST.ISPTABL\n" +
            "//SYSUDUMP DD SYSOUT=* \n" +
            "//SYSTSIN DD DSN=&&TEMPFILE,DISP=(OLD,DELETE,DELETE)\n" +
            "/*";
        return js.submitJCL(session, jcl, id: 108);
    }
    return "";
}
```

其中fieldsResolver是对前端输入的一个封装，对比Display，主要区别就是将ACBQBAJ2之后的DISPLAY换成了ALTER。再根据Service编写对应的接口，成功实现了修改的相关功能。ALTER功能的实现基本没有遇到过大的问题，可能是有了之前DISPLAY的铺垫。

4.1.3 List Data Class接口实现

list类型的接口的JCL代码相对来说就要复杂一些了。这也是我们消耗时间和精力最多的接口，由于其中的JCL代码过长，这里只截取了其中的一部分，如下图所示：

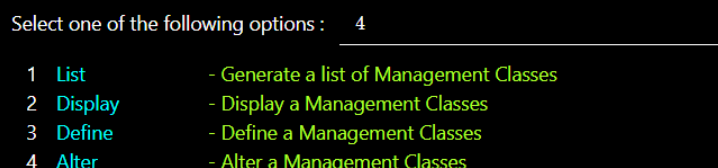
```
public String listDataClass(HttpSession session, DataClass dataClass) {
    if (prepareTable2(session)) {
        String uid = session.getAttribute("S: ZOSMF_Account").toString();
        String jcl = getHead(uid) +
            "//GENDCLST EXEC ACBQBAOB,\n" +
            "//      PLIB1=SYS1.DGTPLIB,\n" +
            "//      TABL2=" + uid + ".TEST.ISPTABL\n" +
            "//SYSSTSIN DD *\n" +
            "PROFILE NOPREFIX\n" +
            "ISPSTART CMD(ACBQBAIC SAVE DCNAMES +\n" +
            "SCDS(" + dataClass.getScds() + ") DATACLAS(" + dataClass.getDcname() + ")) +\n" +
            "NEWAPPL(DGT) BATSCRW(132) BATSCRD(27) BREDIMAX(3) BDISPMAX(99999999)\n" +
            "/*\n" +
            "//DELREPDS EXEC PGM=IDCAMS\n" +
            "//SYSPRINT DD SYSOUT=*\n" +
            "//SYSIN DD *\n" +
            "DELETE DCNAMS.REPORT\n" +
            "/*\n" +
            "//ALCISPFL EXEC PGM=IEFBR14\n" +
            "//ISPFIL DD DSN=DCNAMS.REPORT,DISP=(NEW,CATLG),\n" +
            "//      BLKSIZE=0,SPACE=(TRK,(3,1)),RECFM=FBA,LRECL=133,UNIT=SYSDA\n" +
            "//SYSPRINT DD SYSOUT=*\n" +
            "//SYSIN DD *\n" +
    }
```

这里的jcl同样是对照官方给的文件目录，然后在主机的相应位置找到的，但由于代码过长，因此产生了许多问题，为此，我们不得不现在主机上进行JCL代码的提交，看是否能够正常运行，这个过程便耗费了大量的时间，然后终于在主机上运行成功后，还需要再将JCL代码搬到上图的代码中，测试能否在前端正常显示，这又是一个不断debug的过程。

总体而言，list功能是要比display和alter难很多的。

4.2 前端界面

本目前端总体使用VUE框架，对于本小组所做的相关功能，前端的主要工作是以下界面中除define(学长已做好)之外其他几个选项的选取之后的逻辑：



```
Select one of the following options : 4
1 List          - Generate a list of Management Classes
2 Display       - Display a Management Classes
3 Define        - Define a Management Classes
4 Alter         - Alter a Management Classes
```


原本选取1、2、4选项时直接提示该选项暂不支持，我们前端的任务就需要让输入的数字对应调用后端的接口，总体的实现逻辑还是比较清晰的，但由于本组成员之前没有怎么接触过VUE框架，因此在熟悉框架上花费了一些时间。

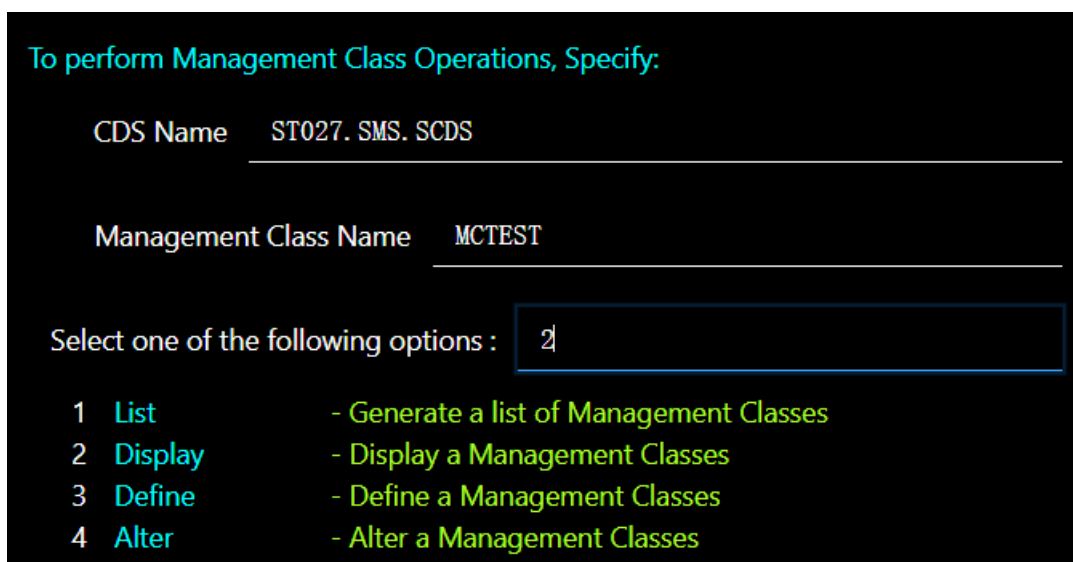
4.3 接口测试

如何测试接口也是一开始困扰本组成员的一大问题。我们尝试了swagger、postman、Yapi等相关工具，发现均不能正常使用，于是，我们选择了直接通过前端进行测试，由于前端原本只有一个接口调用的地方，我们在进行测试时，就每次通过前端调用的url来相应地调用后端的各个接口，相对来说是比较繁琐的，一直到成功实现了前端的其他选项逻辑，测试才变得简单起来。

5、运行结果示例

通过本小组成员的努力，我们最终实现了之前决定的12个接口的编写及相应前端逻辑，这里同样DISPLAY、ALTER和LIST分别选择一种数据结构进行展示，运行结果如下所示。

5.1 Display Management Class



```
To perform Management Class Operations, Specify:

CDS Name      ST027. SMS. SCDS
Management Class Name  MCTEST

Select one of the following options : 2

1 List          - Generate a list of Management Classes
2 Display       - Display a Management Classes
3 Define        - Define a Management Classes
4 Alter         - Alter a Management Classes
```

如上，在相应的面板输入相关参数，然后选择2选项，回车，其中CDS和MC都是之前已经创建成功的。由于显示结果过长，这里只展示其中的一部分。得到的部分运行结果如下：

```

-----
MANAGEMENT CLASS DISPLAY

CDS Name . . . . . : ST027.SMS.SCDS
Management Class Name . : MCTEST

Description : MC TEST

Expiration Attributes

Expire after Days Non-usage . : NOLIMIT
Expire after Date/Days . . . : NOLIMIT
Retention Limit . . . . . : NOLIMIT

```

5.2 Alter Storage Group

为了比较是否修改成功，首先对Storage Group进行一次Display，结果如下所示：

```

CDS Name . . . : ST027.SMS.SCDS
Pool Storage Group Name: SGTEST

Description : SG TEST

Auto Migrate. . . . . : YES
Auto Backup . . . . . : YES
Auto Dump . . . . . : NO
Overflow. . . . . : NO

```

可以看到此时的Description为：SG TEST。

然后选择alter选项

To perform Storage Group Operations, Specify:

CDS Name ST027.SMS.SCDS

Storage Group Name SGTEST

Storage Group Type P00L

Select one of the following options :

| | | |
|---|---------|---|
| 1 | List | - Generate a list of Storage Groups |
| 2 | Display | - Display a Storage Group |
| 3 | Define | - Define a Storage Group |
| 4 | Alter | - Alter a Storage Group |
| 5 | Volume | - Display, Define, Alter or Delete Volume Information |

出现如下界面：

POOL STORAGE GROUP ALTER

SCDS Name :

ST027.SMS.SCDS

Storage Group Name :

SGTEST

To ALTER Storage Group, Specify:

Description:

SG TEST1

Auto Migrate:

Y, N, I or P

Auto Backup:

Y or N

Auto Dump:

Y or N

Overflow:

Y or N

Guaranteed Backup Frequency:

Required, 1 to 9999 or NOLIMIT

Command >

Press enter to submit

在Description栏中填入修改的描述内容，Guaranteed Backup Frequency也需要填入一个数，再回车。为了显示修改结果，需要再一次进行Display，得到如下界面：

```
CDS Name . . . : ST027.SMS.SCDS
Pool Storage Group Name: SGTEST

Description : SG TEST1

Auto Migrate. . . . . : YES
Auto Backup . . . . . : YES
Auto Dump . . . . . : NO
Overflow. . . . . : NO
```

可以看到已经修改成功。

5.3 List Data Class

DATA CLASS APPLICATION SELECTION

To perform Data Class Operations, Specify:

CDS Name

ST027. SMS. SCDS

Data Class Name

DCSDS

Select one of the following options:

1

1 List

- Generate a list of Data Classes

2 Display

- Display a Data Class

3 Define

- Define a Data Class

4 Alter

- Alter a Data Class

在这里进行相关参数的输入，选项中填1，然后得到的部分结果如下：

1

| STATUS OF DATA CLASSES AS ON 12/01/12 | | | | | | | | | | | |
|---------------------------------------|--------|-------|-------|--------|--------|--------|-------|---------|----------|----------|--------|
| DATACLAS | RECORG | RECFM | LRECL | KEYLEN | KEYOFF | AVGREC | AVGVL | SPCEPRI | SPACESEC | SPACEDIR | EXPDAT |
| DCSDS | -- | FB | 80 | --- | ---- | U | 2 | 3 | 3 | 3 | ---- |

1

L E G E N D

DATACLAS - DATA CLASS.

RECORG - RECORD ORGANIZATION.

RECFM - RECORD FORMAT.

LRECL - LOGICAL RECORD LENGTH.

KEYLEN - KEY LENGTH.

KEYOFF - KEY OFFSET.

AVGREC - AVERAGE RECORD LENGTH.

6、结论

这次的主机项目对我们的帮助还是很明显的。首先，通过这次的仿真平台的使用，我们对主机的存储管理相关的组件及功能有了更加深入的理解，对于主机如何与其他部分进行交互有了了解，这种仿真平台对于使用了一学期的主机界面的我们来说还是较为新颖的，这也鼓励我们去努力完善这个平台。

另一方面，由于时间紧迫，我们小组在进行开发时，伴随着大量的沟通交流，使得各个成员对于自己的任务十分明确，并且在共同讨论的情况下，问题的解决也会变得更加简单，这充分提高了我们的编码效率，锻炼了我们的合作能力。同时我们的主机编程能力和前端编程能力也有所提升。

总之，我们会将这次实验的所得充分利用起来，提升个人的能力，从而在今后解决类似问题的时候能够得心应手。

7、工作量分配

| | 工作 | 占比 |
|-----|---------------------------------------|-----|
| 谢康 | 后端alter、display、alter接口各一个、PPT、版本管理测试 | 1/3 |
| 黄金鑫 | 文档编写、PPT、后端三个alter接口、一个display接口 | 1/3 |
| 谢尚汝 | 前端、后端list三个接口、一个display接口 | 1/3 |