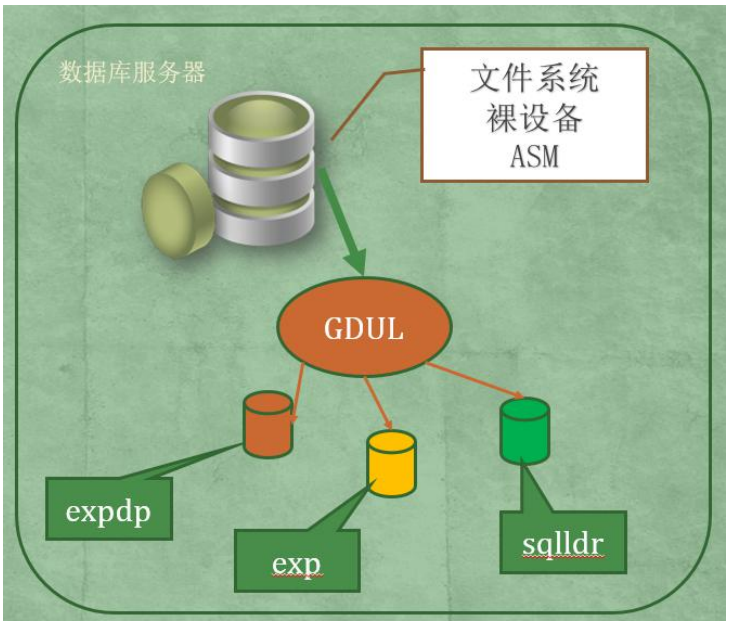
GDUL 用户指南 V1.8

目录

简介

GDUL 是老耿开发的一款类 dul 工具，当数据库由于某种原因无法打开时，可以利用 GDUL 把表数据直接读取出来。



功能特点：

1. 支持绝大多数列类型，支持常见类型，如 NUMBER，CHAR，VARCHAR2，DATE，LOB，LONG、XMLTYPE 等。其中 SecureFile LOB 支持压缩，尚不支持去重和加密。
2. 支持导出常规表、IOT、Cluster 表、分区表、压缩表。
3. 完整支持多种格式导出，包括 expdp, exp, text 格式。目前市面上的类 dul 工具只有 GDUL 支持 expdp 格式。
4. 支持 ASM 文件系统，并内置 asmcmd 命令，支持 copy ASM 文件到文件系统。
5. 支持 truncated 表、删除行、drop 表恢复。
6. 支持无 SYSTEM 表空间恢复。
7. 支持常规表空间和 bigfile 表空间。
8. 支持主流硬件平台(HP-UX, AIX, Solaris, Linux, Windows)，各个平台仅需单一的可执行文件，方便分发。
9. 免费使用，无需额外费用。

文件说明

GDUL 介质列表

安装介质	说明
gdul<版本号>.tar.gz	GDUL for Unix/Linux 版本，各版本可执行文件位于 bin_file 目录。 使用时请 ln -s bin_file/gdul_<平台> gdul 建立符号链接。
gdul<版本号>_win64.zip	GDUL for Windows 软件介质。
gdul_user_guide_<版本>.pdf	用户操作手册。

gdul 目录结构

目录	文件	说明
根目录	osetup	配置 ASM 磁盘、数据文件列表
	gdul	可执行文件，每个平台 1 份
conf	gdul.ini	参数文件
	datafile.ini	数据文件列表
	asmdisk.ini	asm 磁盘列表
dict	*.dat	存放 SYS 用户下的字典表数据
dump	*.dmp, *.dat	存放导出 dmp、text 文件
log	gdul.log	主日志文件
	unload_table.log	表导出日志
sample	*.dict, *.dat	无 system 表空间时存放行数据采样文件
bin_file	*	各平台的 gdul 可执行文件

gdul.ini 文件

参数	描述	取值范围	默认值
db_compat_version	数据库版本	sqlplus -v 输出	11.1.0.0
db_block_size	默认数据块大小	4096, 8192, 16384, 32768	8192
file_raw_offset	默认文件头偏移量，仅适用于 AIX	0 或 4096	0
reverse_byte	读取不同 CPU 平台的数据文件	false, true	false
export_format	导出数据格式	SQLLDR EXP	EXPDP

		EXPDP	
ldr_enclose_char	文本方式导出时，字段分隔符	ascii 字符	|
trace_block	导出时跟踪块	false, true	false

注意：12.1 和 12.2 版本的字典表发生变动，需要明确指定 db_compat_version 的值，即 sqlplus -v 的输出。

最佳实践

提示：如果数据库处于打开状态，请执行 alter system checkpoint; 以便刷新数据块到数据文件。

1 生成配置文件

osetup 命令在 Linux/Unix 下执行，数据库 mount 或 open 时，可以生成 GDUL 参数文件(gdul.ini)、ASM 磁盘列表(asmdisk.ini)、数据文件列表(datafile.ini)。

$ ls -rlt

```
-rwxr-xr-x 1 andy andy 4732 Feb 18 14:38 osetup*
drwxrwxr-x 2 andy andy 4096 Feb 18 14:45 bin_file/
lrwxrwxrwx 1 andy andy      21 Feb 18 14:46 gdul -> bin_file/gdul_linux64*
```

1) $ *./osetup*

```
GDUL setup program, version 4.0.0.1
---------------------------------------------------------------------
ORACLE_HOME:       /u01/app/oracle/product/12.1.0
ORACLE_SID:        db12101
ORACLE_VERSION: 12.1.0.1.0
---------------------------------------------------------------------
...
setup has completed sucessfully.
```

2) 设置导出格式，建议导出成 expdp(conf/gdul.ini)

```
export_format:     EXPDP
```

导出格式说明

导出格式	说明	限制
EXPDP	导出成 10.1 expdp 格式	1. 无法导入到 9i 数据库中
EXP	导出成 8.1.6 exp 格式	1. 不支持 NCLOB 2. 不支持 XMLTYPE AS BINARY
SQLLDR	导出成 sqlldr 加载的纯文本格式	1. 不支持 XMLTYPE AS BINARY 列和 xmltype 表。

示例：

1)impdp

```
impdp andy/andy directory=GDUL_DIR tables=t_secure_clob
dumpfile=ANDY_t_secure_clob.dmp remap_schema=andy:andy2
table_exists_action=truncate
```

2)imp

```
imp andy/andy TABLES=t_secure_clob file=<GDUL_DIR>\dump\ANDY_t_secure_clob.dmp
FROMUSER=andy TOUSER=andy2 ignore=y
```

3)sqlldr

```
cd <GDUL_DIR>\dump
sqlldr userid=andy/andy control=ANDY_t_secure_clob.ctl
```

ASM 注意事项

asmdisk.ini 文件中请只保留已位于磁盘组中的 ASM 磁盘列表。

kfod disks=all 输出包含了完整的 asm 磁盘列表，而 v$asm_disk 下磁盘列表可能包含不完整，或者有未使用的磁盘，请对照 kfod 输出和 asmdisk.ini 文件手工去除多余或不在任何 DG 中的磁盘。

平台相关注意事项

➢ AIX 平台

AIX 下的裸设备如果使用裸 LV 的话，需要先确认是否带有 4k 头。

1)数据文件使用 VG 中的裸 LV

如果数据文件使用的裸 LV，可以使用 dbfsize 命令可看是否有 4k 头。

下面是包含 offset 的裸设备

$dbfsize /dev/<RLV_DATAFILE_NAME>

```
Database file: /dev/<RLV_DATAFILE_NAME>
Database file type:raw device   具有 4k 头时显示
Database file type:raw device without 4K starting offset   无 4k 头时显示
Database file size: ##### #### byte blocks
```

说明：如果有 4k 头，就需要把 datafile.ini 中对应数据文件的 offset 设置成 4096。

2)ASM 使用 VG 中的裸 LV

如果 ASM 磁盘使用的裸 LV，可以查看 lslv 命令的 DEVICESUBTYPE 确认。

$lslv -L <LV_NAME>

通过查看 lv 的属性 DEVICESUBTYPE：DS_LVZ 说明没有偏移的 lv，DS_LV 说明的 4k 头；如果未显示 DEVICESUBTYPE 属性，也说明具有 4k 头。

说明：如果具有 4k LV 头，需要把 asmdisk.ini 中对应的 offset 改成 4096。

➢ Linux 平台

1)ASM 使用 ASMLIB

如果用到 ASMLIB，需要把 asmdisk.ini 文件中的 ORCL***磁盘名替换成对应的/dev/物理磁盘文件。

1. v$asm_disk 视图

```
select   ' <row> '          ||
         '<disk_number>' ||disk_number ||'</disk_number> ' ||
         '<path>'            ||path            ||'</path> '            ||
           '<raw_offset>'   ||0                ||'</raw_offset> '   ||
         '</row>'
  from v$asm_disk;
```

2. 查看所有 ASM 磁盘列表

   $kfod disks=all

3. 查找 ORCL 盘和物理磁盘对应关系

   $oracleasm listdisks

   VOL1

   $oracleasm querydisk -d VOL1

➢ Windows 平台

Windows 下无法执行 osetup 脚本，需要手工创建目录和 conf/下的配置文件

1)创建目录

在 gdul 可执行文件目录下，创建 conf, dict, log, dump, sample 子目录，并且在 conf 子目录下创建 gdul.ini, datafile.ini, asmdisk.ini。

2)创建 gdul.ini 文件

注意：需要更改 db_compat_version 为当前数据库版本，注意 12c 及以上版本需要指定详细版本号，如 12.1.0.1，请参见 sqlplus -v 输出。

```
<!--DB compatiable version, 9, 10, 11, 12-->
<db_compat_version>11.1.0.1</db_compat_version>

<!--default block size, 4096, 8192, 16384, 32768-->
<db_block_size>8192</db_block_size>

<!--default datafile raw offset, 0, 4096-->
<file_raw_offset>0</file_raw_offset>

<!--extract DB files on different endian platform-->
<reverse_byte>false</reverse_byte>

<block_checksum>true</block_checksum>

<!--export to SQLLDR, EXP, EXPDP-->
<export_format>EXPDP</export_format>

<!--text format splite char-->
<ldr_enclose_char>|</ldr_enclose_char>

<!--unload block trace-->
<trace_block>false</trace_block>
```

3)创建 datafile.ini 文件

SQL 语句：

```
set linesize 200
select ' <row> '            ||
            '<file#>'          ||file#          ||'</file#> '          ||
            '<ts#>'             ||ts#             ||'</ts#> '             ||
            '<rfile#>'        ||rfile#        ||'</rfile#> '        ||
            '<blocks>'        ||blocks        ||'</blocks> '        ||
            '<block_size>' ||block_size ||'</block_size> ' ||
            '<name>'           ||name           ||'</name> '           ||
            '<raw_offset>' ||0                ||'</raw_offset> ' ||
            '</row>'
  from v$datafile;
```

文件最终格式：

```
<datafile>
 <row> <file#>1</file#> <ts#>0</ts#> <rfile#>1</rfile#> <blocks>256000</blocks>
<block_size>8192</block_size> <name>/datafiles/a110a64f/data/system01.dbf</name>
<raw_offset>0</raw_offset> </row>
 <row> <file#>2</file#> <ts#>1</ts#> <rfile#>2</rfile#> <blocks>211744</blocks>
<block_size>8192</block_size> <name>/datafiles/a110a64f/data/sysaux01.dbf</name>
<raw_offset>0</raw_offset> </row>
</datafile>
```

4)创建 asmdisk.ini 文件

SQL 语句：

```
set linesize 200
select ' <row> '            ||
            '<disk_number>' ||disk_number ||'</disk_number> ' ||
            '<path>'             ||path             ||'</path> '             ||
          '<raw_offset>'   ||0                ||'</raw_offset> '   ||
            '</row>'
  from v$asm_disk;
```

文件最终格式：

```
<asmdisk>
```

asm disk 列表，非 asm 环境为空行。

</asmdisk>

2 数据字典(SYSTEM 表空间)正常时，恢复方法

1 执行 GDUL 命令，初始化数据字典

$./gdul

GDUL> bootstrap

```
Bootstrap finish.
```

说明：bootstrap 读取 SYSTEM 表空间中的数据字典，如果数据库处于 OPEN 状态，可能需要执行 alter system checkpoint;命令来同步数据文件，然后再执行该命令。

2 显示数据库信息

显示数据文件列表

GDUL> info

FILE#	TS#	RFILE#	BIGFILE	SIZE(GB)	NAME
1	0	1	FALSE	0.68	D:\APP\ORADATA\TEST32\SYSTEM01.DBF
2	1	2	FALSE	0.59	D:\APP\ORADATA\TEST32\SYSAUX01.DBF
3	2	3	FALSE	1.01	D:\APP\ORADATA\TEST32\UNDOTBS01.DBF
4	4	4	FALSE	8.88	D:\APP\ORADATA\TEST32\USERS01.DBF
5	5	5	FALSE	0.13	D:\APP\ORADATA\TEST32\TBS01.DBF
6	6	6	FALSE	0.01	D:\APP\ORADATA\TEST32\DMP01.DBF
7	4	7	FALSE	4.00	D:\APP\ORADATA\TEST32\USERS02.DBF
8	7	9	FALSE	0.01	D:\APP\ORADATA\TEST32\TBSFNO01.DBF
… …					

显示数据库用户列表

GDUL> list user

ID	NAME	TABLE_CNT
0	SYS	1258
7	AUDSYS	1
8	SYSTEM	178

```
…
109             OE                          14
110             SCOTT                       4
…
```

显示当前用户下的表

GDUL>set user andy

GDUL>list table

```
ID              NAME                                DICT_ROWS    DICT_BLOCKS
---------- ------------------------------ ---------- ----------
15727           T_OBJ1                              0            0
16814           T_TEST_TYPE                         2            13
16824           T_PERSON                            2            5
17065           T_ROWID                             2            5
17070           T_RAW                               5            9
17252           T_HELLO                             0            0
18969           T_TEST                              3            5
18972           T_EXT_TABLE3                        0            0
19449           T_BIG_TABLE                         8099838      105819
19552           T_FLM2                              1            1
20269           T_CLOB4                             2            4
```

显示表结构

GDUL>desc andy.t_test

```
object_id: 18969, dataobj#: 35558, cluster tab#: 0
segment header: (ts#: 4, rfile#: 4, block#: 138))

Seg Column#    Column#       Name                      Null?              Type
------------ ---------- -------------------- --------------- --------------
1                 1          OWNER                                  VARCHAR2(30)
2                 2          OBJECT_NAME                            VARCHAR2(128)
3                 3          SUBOBJECT_NAME                         VARCHAR2(30)
4                 4          OBJECT_ID                             NUMBER
```

5	5	DATA_OBJECT_ID	NUMBER
6	6	OBJECT_TYPE	VARCHAR2(19)
7	7	CREATED	DATE
8	8	LAST_DDL_TIME	DATE
9	9	TIMESTAMP	VARCHAR2(19)

3 表导出

说明：

1. 如果数据库处于打开状态，建议先执行 SQL>alter system checkpoint;以便刷新磁盘数据块。
2. 导出的表位于 dump 目录，然后可以根据导出类型(expdp, exp, sqlldr)，再导入到表中。

导出单张表

GDUL> unload table andy.t_blob

```
2016-02-18 14:59:27...unloading table T_BLOB 2 rows unloaded.
```

导出用户下所有表

GDUL>unload user andy

```
About to unload ANDY's tables, total cnt: 43
 2016-02-18 14:55:37 unloading table DEPT...
 2016-02-18 14:55:37 unloaded 4 rows.
 2016-02-18 14:55:37 unloading table EMP...
 2016-02-18 14:55:37 unloaded 14 rows.
 2016-02-18 14:55:37 unloading table DEMO_TAGS...
 2016-02-18 14:55:37 unloaded 6 rows.
 2016-02-18 14:55:37 unloading table DEMO_TAGS_TYPE_SUM...
 2016-02-18 14:55:37 unloaded 3 rows.
 2016-02-18 14:55:37 unloading table DEMO_TAGS_SUM...
 2016-02-18 14:55:37 unloaded 3 rows.
```

恢复 delete 的表

GDUL>undelete table andy.t_test

```
2016-04-12 13:17:00...undeleting table T_TEST 6 rows unloaded.
```

限制：

1．对于链接行，如果单列(如 LONG, VARCHAR)也出现链接，由于已删除的行片中无法判断列链接标记，会导致问题。

2. 对于 OLTP 压缩表，如果已删除的行片对应的符号表行已被删除，将无法导出。

恢复 truncated 的表

1) 查看表所在的表空间为 7

GDUL> desc andy.t_test

```
object_id: 108775, dataobj#: 112379, cluster tab#: 0
segment header: (ts#: 7, rfile#: 5, block#: 562))

Seg Column#    Column#       Name                          Null?              Type
------------ ---------- -------------------- --------------- --------------
1                   1              OWNER                                     VARCHAR2(30)
…
```

2) unload 发现行数为 0

GDUL> unload table andy.t_test

```
2016-02-18 15:03:39...unloading table T_TEST 0 rows unloaded.
```

3) 扫描表空间 7

GDUL> scan tablespace 7

```
start scan tablespace 7...
scan tablespace completed.
```

说明：如果磁盘性能较好，且表空间下有多个数据文件，可以采用并行方式。

GDUL> scan tablespace 7 parallel 10

4) 导出 truncated 的行

4.1) truncate 后无数据进来，可以直接恢复

GDUL> untrunc table andy.t_test

```
2016-02-18 15:04:03...untruncating table T_TEST 99998 rows unloaded.
```

4.2) truncated 后，有新数据进来，将无法自动判断出旧的 data_object_id。需要先找到 truncate 前的 data_object_id，再恢复。

1. 查找 truncate 前的 data_object_id。

查找 data_object_id 方法：

方法 1：使用闪回查询

如果 SYS UNDO 段未被覆盖的话，可以闪回查找旧的信息。

```
select obj#, dataobj# from sys.obj$
  as of timestamp to_timestamp('2017-02-08 19:00:00','YYYY-MM-DD
HH24:MI:SS')
  where owner# = (select user_id from dba_users where username='ANDY')
  and name='T_TEST';
```

此外，还可以获取 drop 掉的表结构：

```
select column_name, data_type, data_length
  from dba_tab_columns as of timestamp to_timestamp('2017-02-08
19:00:00','YYYY-MM-DD HH24:MI:SS')
 where owner = 'ANDY'
  and table_name='T_TEST';
```

方法 2：AWR 表中可能会记录有旧的 data_object_id.

```
select s.snap_id, s.begin_interval_time, o.objno, o.tsno, o.dataobjno
from wrh$_seg_stat_obj o
 inner join wrm$_snapshot s
    on o.snap_id = s.snap_id
 where owner='ANDY' and object_name = 'T_TEST'
 order by s.snap_id;
```

方法 3：手工 sample 表空间，查找

GDUL>sample segment all

$cd sample，手工分辨所有的采样数据，最后确认 data_object_id。

方法 4：使用 logminer

先找到 truncate 时的归档或在线日志，然后从 logminer 输出中查找 truncate 的 DDL 语句，truncate 后几行记录 update "SYS"."TAB$"和 update "SYS"."OBJ$"中会记录有旧的 data_object_id。

```
select * from v$log;
select * from v$logfile;
select thread#, sequence#, name, first_time, next_time from
```

```
v$archived_log
where to_date('2017-02-08 19:00:00','YYYY-MM-DD HH24:MI:SS') between
first_time and next_time;

exec
dbms_logmnr.add_logfile('/oradata/db10205/redo03.log',dbms_logmnr.new)
;
exec
dbms_logmnr.start_logmnr(options=>dbms_logmnr.dict_from_online_catalo
g);
drop table test_log;
create table test_log tablespace sysaux
as select * from v$logmnr_contents;
execute dbms_logmnr.end_logmnr;
```

2. 找到 data_object_id 后，便可以恢复

GDUL>unload table andy.t_test object_id <data_object_id>

恢复 drop 表

1) 创建一张临时的表，表结构和 drop 掉的表相同。
SQL>create table t_test2... ... tablespace system;
说明：需要借助上述新建表的表结构来恢复 drop 的表，建议表空间设置为非 drop 表所在的表空间，以免数据块被覆盖。

2) 初始化数据字典，以识别新创建的表。
GDUL>bootstrap

3) 扫描 drop 掉的表所在表空间，得到表空间内所有 data_object_id 及对应的数据块。
GDUL> scan tablespace 7
说明：该步骤建立指定表空间下所有 data_object_id 和数据块的对应关系，在接下来的步骤 4，步骤 5 都需要用到。

说明 2：如果磁盘性能较好，且表空间下有多个数据文件，可以采用并行方式。

GDUL> scan tablespace 7 parallel 10

4) 查找 drop 表当时的 data_object_id。
方法请参见上述 truncate 表恢复的《查找 data_object_id 方法》

5) 借助步骤 1 创建的表结构，用步骤 4 找到的 drop 掉的表的 data_object_id 导出表数据

GDUL>unload table t_test2 object_id <data_object_id>

4 有坏块场景恢复

导出单张表，字典正常，但段头损坏

段头损坏症状：

GDUL> unload table andy.t_test_corrupt_header

```
2016-05-09 11:13:19 unloading table "ANDY"."T_TEST_CORRUPT_HEADER"...
  unload table error: segment header parse error, object_id: 36399, dba[6, 10, 130]: block tail checksum error, consistency value in tail: 0xc9312301
2016-05-09 11:13:19 unloaded 0 rows.
```

扫描表空间#：

GDUL> desc andy.t_test_corrupt_header

```
object_id: 36399, dataobj#: 36399, cluster tab#: 0
segment header: (ts#: 6, rfile#: 10, block#: 130))

Seg Column#    Column#       Name                          Null?              Type
------------ ---------- -------------------- --------------- --------------
1                 1          OWNER                                    VARCHAR2(30)
2                 2          OBJECT_NAME                              VARCHAR2(128)
3                 3          SUBOBJECT_NAME                           VARCHAR2(30)
4                 4          OBJECT_ID                               NUMBER
…
```

GDUL> scan tablespace 6

```
start scan tablespace 6...
scan tablespace completed.
```

使用 scan 选项导出表：

GDUL> unload table andy.t_test_corrupt_header scan

```
2016-05-09 11:13:51 unloading table "ANDY"."T_TEST_CORRUPT_HEADER"...
2016-05-09 11:13:51 unloaded 124 rows.
```

3 无 SYSTEM 表空间恢复

1) 设置字符集等参数(gdul.ini)

参数	值
db_charset	ZHS16GBK
db_ncharset	AL16UTF16
db_timezone	+08:00

说明：可以查询其它相似库来获取参数值：

```
select name, value$
  from sys.props$
 where name in ('NLS_CHARACTERSET',
'NLS_NCHAR_CHARACTERSET',
'DBTIMEZONE');
```

2) 扫描表所在表空间

GDUL> scan tablespace ###

说明：该操作会输出到 dict/scan_*.dat 文件中。

默认扫描线程数为 1，如果磁盘性能较好，且表空间下有多个数据文件，可以采用并行方式，以加快扫描速度。

示例：*GDUL> scan tablespace 7 parallel 10*

3) 采样已扫描的表空间

GDUL>sample segment all

$cd sample

说明：该操作执行采样，并在 sample 目录下生成 segment 列定义及采样数据。

注意：

采样的列类型可能不准确，如果有测试库可以获取表结构，可以替换掉 sample/seg_<data_object_id>.dict 中<column_def></column_def>内的采样列定义，再执行 unload

从其它库获取表定义语句：

```
SQL>select '<row>'
```

```
    || '<COL#>'               || col#               || '</COL#>'
    || '<SEGCOL#>'            || segcol#            || '</SEGCOL#>'
    || '<INTCOL#>'            || intcol#            || '</INTCOL#>'
    || '<TYPE#>'              || type#              || '</TYPE#>'
    || '<NAME>'               || name               || '</NAME>'
    || '<LENGTH>'             || length             || '</LENGTH>'
    || '<SEGCOLLENGTH>' || segcollength || '</SEGCOLLENGTH>'
    || '<PRECISION#>'      || precision#      || '</PRECISION#>'
    || '<SCALE>'              || scale              || '</SCALE>'
    || '<CHARSETID>'       || charsetid       || '</CHARSETID>'
    || '<CHARSETFORM>'    || charsetform    || '</CHARSETFORM>'
    || '</row>'
  from sys.col$
 where obj# = (select object_id from dba_objects where owner = '<OWNER>' and
object_name = '<TABLE_NAME>')
 order by col#;
```

4) unload 采样出的数据段

GDUL>unload segment all | <data_object_id>

说明：用指定的 data_object_id unload 表数据，指定 all 会恢复所有 segment

5) 演示

命令参考

1 内置 asmcmd

说明：内置 asmcmd 可以在 ASM 实例未打开，但文件头和目录块完整时，查看 ASM 文件信息，并且可以 copy 到本地文件系统。

GDUL> asmcmd

```
Enter internal asmcmd.
```

1) ls、cd 命令

ASMCMD> ls

```
NORMAL_DATA/
DATA/
```

ASMCMD> cd data

ASMCMD> ls

```
test-cluster/
TEST/
```

ASMCMD> cd test

ASMCMD> ls -l

```
Type          Redund        Striped       Sys           Name
---------- ---------- ---------- ---------- ----------
                                          Y             DATAFILE/
                                          Y             CONTROLFILE/
                                          Y             ONLINELOG/
                                          Y             TEMPFILE/
                                          Y             PARAMETERFILE/
                                          N              spfiletest1.ora => DATA.265
```

2) copy 命令

ASMCMD> cp spfiletest1.ora d:\test.ora

```
copy SPFILETEST1.ORA to d:\test.ora sucessfully, bytes: 2560
```

常见问题

1. ASM 常见问题：

1) 执行 gdul 时如果出现磁盘文件头无法解析的问题，可能是磁盘文件尚未加入到磁盘组中，需要从 gdul.log 中查找最后一个成功的磁盘文件，然后从 asmdisk.ini 中把它下一个文件(即未成功打开的文件)手工删除。

2)