**Frequently used M\_\* Commands in Abinito:**

Abinito provides various m\_\* commands for performing data manipulation. These commands will be useful for debugging the data problem. Abinito is mostly used in Unix environment all these M\_\* commands can be used from the command prompt.

**m\_dump:**

**Purpose:**

This command is used to print information about data records, their record format, and the evaluations of expressions. This command can be used to check whether data file is good and is according to dml specified. These commands are very useful to debug the output and reject files directly from the Unix machines without using the Abinitio GDE. m\_dump command without any options will not be useful for a huge file, instead using the –start and –end option to got to a specific record will be useful.

**Syntax:**

***m\_dump <metadata/DML Location> <data/Data file name>[action]***

**Examples:**

This command displays all the records from test1.dat file:

$m\_dump /export/home/rrobert/test1.dml /export/home/rrobert/test1.dat|more

This command displays records starting from 5th row to 9th row in xyz.dat file:

$ m\_dump xyz.dml xyz.dat -start 5 -end 9

**m\_eval:**

**Purpose:**

This command is used to evaluate DML expressions and displays their derived types. This command can be executed from Shell or used inside the GDE for validation purpose.

**Syntax:**

***m\_eval [ option ... ] expression [ expression ...***

**Examples:**

This command subtracts 10 days from the current date and represents the result in a specific date format:

$ m\_eval **'**(date("YYYYMMDD")) (today() - 10)**'**=>   "20041130"

This casts an integer to a decimal:

$ m\_eval '((decimal(5)) 42)'  =>  "ooo42"

**m\_wc:**

**Purpose:**

This command is used to count number of records from one or more data files/multi files.

**Syntax:**

***m\_wc [ -no-commas ] metadata data***

**Examples:**

$ m\_wc myrecfmt.dml mfs/mydata.dat

209,7152 4,297,064,448 mfs/mydata.dat

**m\_env:**

**Purpose:**

This command is used to obtain all the Ab Initio configuration variable settings in an environment. This command will be useful to get Abi environment variable values.

**Syntax:**

***m\_env [-describe]***

**Examples:**

m\_env

Current Configuration Variable Settings for Ab Initio:

Variable Set Value or Resulting Action

-------------------- --- --------------------------------------------AB\_AIR\_BRANCH <unset>

AB\_WORK\_DIR \* /var (from environment)

**m\_kill:**

**Purpose:**

This command is used to kill a running job. The command blocks until it can verify that the job is killed or after AB\_TIMEOUT seconds, whichever comes first. This command will be useful incases when jobs were started accidentally.

**Syntax:**

***m\_kill [ signal] {jobname.rec | jobname }***

signal – can have three values –TERM –KILL and –QUIT

-TERM Kills the job and triggers a rollback. This is the default.

-KILL Kills the job immediately without triggering a rollback. You must issue the command m\_rollback later.

-QUIT Kills the job immediately without triggering a rollback and forces the Xmp run process to dump core (system limits permitting).

**Examples:**

m\_kill -TERM f\_so\_line\_work\_F2CMART.rec

Output:

Failed opening recovery file /a1004/home/test/LVLTdw/f2c/bin/f\_so\_line\_work\_F2CMART.rec: No such file or directory

[1] + Done(4) nohup f\_so\_line\_work\_F2CMART.ksh &

**m\_rollback:**

**Purpose:**

This command is used perform a manual rollback incase of any Abinito job failure. It may not always be possible for the Co>Operating System to restore the system to an earlier state. For example, a failure could occur because a host or its native operating system crashed. In this case, it is not possible to cleanly shut down flow or file operations, or to roll back file operations performed in the current phase. In fact, it is likely that stray files (intermediate temporaries) will be left lying around.

**Syntax:**

***m\_rollback [-d] [-i] [-h] [-kill] recoveryfile***

**Options:**

-d Delete the job along with its recovery file and any log files it created.

-i Display the state of the job and prompt the user whether the job should be deleted.

-kill causes m\_rollback to attempt to kill the job.

If the -i option is not used, jobs that have reached their first checkpoint will be rolled back to the checkpoint. Jobs that do not include checkpoints or that did not reach their first checkpoint will be deleted. The default behavior of m\_rollback when called on a running job is to display a warning message then exit.

**Examples:**

m\_rollback -d my-job.rec

ABINITIO: Recovery of job from recovery file "/l8b/ user\_name /my-job.rec" in progress

ABINITIO: Job closed and recovery file deleted

**Working with Multifiles:**

Ab Initio multifiles are [parallel files](mk:@MSITStore:C:%5CProgram%20Files%5CAb%20Initio%5CAb%20Initio%20GDE%5CProgram%20Files%5CAb_Initio_Help.chm::/Parallelfiles.html#1077404) composed of individual files, typically located on different disks and usually, but not necessarily, on different systems. Multifile is partition of single individual file. The multifile organizes the partitions into a single entity, and enables you to manage them as you would a serial file. An Ab Initio multifile is a single, virtual file; you can reference all the partitions of the multifile at once by referencing its [control partition](mk:@MSITStore:C:%5CProgram%20Files%5CAb%20Initio%5CAb%20Initio%20GDE%5CProgram%20Files%5CAb_Initio_Help.chm::/Multifilesystems.html#1002465).

Following are the steps to create a multifile directory for an application.

* Create the data and partitions directory to keep the data and control files.
* Execute the m\_mkfs command from the data directory to create the multifile system.

**[m\_mkfs](http://softdocs.blogspot.in/2011/03/abinito-useful-commands.html)**

**Purpose:**

This command is used to create a multifile system. Multifile system consists of Control file directory along with the partitions directory. Control file resides in the $APPL\_DATA/mdata directory and data files in partitions directories. Depending upon number of partitions multifile system will be created.

**Syntax:**

***m\_mkfs [ options ] control\_url partition\_url [ partition\_url ... ]***

**Following are the m\_mkfs options:**

|  |  |
| --- | --- |
| -m mode or  -mode mode | Set the protection for the created multifile system. The mode option is one of the symbols accepted by the Unix chmod command. |
| -mvfile | Specify that all multifiles in a multi-directory (and in all its multi-subdirectories) will have vfiles (segmented files) as their partitions. (By default, the multifiles in a multi-directory have standard files as their partitions.) |
| -max bytes or  -max-segment-size  bytes | Specify the maximum segment size for vfile partitions. (The default is a system-specific value.) Note that the value supplied serves only as an approximation of the maximum segment size. It is advisable to use a value somewhat smaller than the actual desired value. This option is only meaningful when -mvfile is also specified. |

**Examples:**

This command will create a multifile system with 5 way partitions.

m\_mkfs $/EXPORT/HOME/TEST/DATA/mdata \

$/EXPORT/HOME/TEST/DATA/partitions/0 \

$/EXPORT/HOME/TEST/DATA/partitions/1 \

$/EXPORT/HOME/TEST/DATA/partitions/2 \

$/EXPORT/HOME/TEST/DATA/partitions/3 \

$/EXPORT/HOME/TEST/DATA/partitions/4

Note: Before executing the above command create mkdir $/EXPORT/HOME/TEST/DATA/partitions

**Multifile Commands:**

Working with multifiles will be similar to regular files, by adding a prefix of m\_<command> most of the regular operations can be carried out. Following are the frequently used commands used with the multifiles.

|  |  |  |
| --- | --- | --- |
| **Command** | **Syntax** | **Purpose** |
| m\_mv | m\_mv *source\_path1.. dest\_path* | Similar to unix move commad to move files  from one location to other |
| m\_rmfs | m\_rmfs *path* | To remove the multifile system |
| m\_rm | m\_rm *url* [*url* ...] | Similar to Unix rm command to removes  multifiles or files. |
| m\_chmod | m\_chmod *mode url* [*url* ...] | Similar to Unix chmod to provide permissions to  multifiles |
| m\_touch | m\_touch *url* [*url* ...] | Similar to Unix touch command to create a empty  multifiles |
| m\_cp | m\_cp *source\_file\_url1..* *dest\_url* | Similar to Unix Copy command to create copies of multifiles |

**m\_db**

This command is used for performing database operation from the command prompt. Following are the various m\_db commands that are used frequently.

|  |  |  |
| --- | --- | --- |
| **Command** | **Syntax** | **Purpose** |
| m\_db unload | m\_db unload dbc\_file -table tablename | To Unloads data from database table, select or expression to stdout.  *Eg: m\_db unload mydb.dbc -table 'fred.mytable'* |
| m\_db truncate | m\_db truncate dbc\_file -table tablename | To truncate a table. |
| m\_db gendml | m\_db gendml dbc\_file -table tablename | Used to generate appropriate metadata(dml) for the table/insert/select or expression. This is very useful command for generating dynamic dml. *Eg: m\_db gendml mydb.dbc -table 'fred.mytable'* |
| m\_db genctl | m\_db genctl dbc-name\  -dml 'metadata-string' -table tablename \  -component name | Used to generate a load control file based on the DML supplied |
| m\_db test | m\_db test dbc\_file | Runs diagnostic tests against your database for the dbc file specified. This command can be used to check db connections from wrapper scripts |
| m\_db load | m\_db load dbc\_file -dml 'metadata-string'  -table tablename | Loads data to a database table or insert statement from st |