

Carrying out runs using mrun

PALM group

Institute of Meteorology and Climatology, Leibniz Universität Hannover

last update: 21st September 2015



What is mrun?

- ▶ **mrun (model run)** is a shell script (using `bash/ksh`-syntax) which can be used to compile and run programs, including the handling of input/output files.

What is mrun?

- ▶ **mrun (model run)** is a shell script (using `bash/ksh`-syntax) which can be used to compile and run programs, including the handling of input/output files.
- ▶ The `mrun`-command has a number of options to control the program execution

```
mrun -d example_cbl -h lcsgih -K parallel -X8 -T2 ...
```

All options including a short description can be displayed by entering

```
mrun ?
```

What is mrun?

- ▶ **mrun (model run)** is a shell script (using `bash/ksh`-syntax) which can be used to compile and run programs, including the handling of input/output files.
- ▶ The `mrun`-command has a number of options to control the program execution

```
mrun -d example_cbl -h lcsgih -K parallel -X8 -T2 ...
```

All options including a short description can be displayed by entering

```
mrun ?
```

- ▶ The shellsript execution is also controlled by a configuration file with default name `.mrun.config`

Carrying Out Runs Using mrun

Carrying out a program run typically needs two steps:

1. compilation

```
f95 ... file1.f90 file2.f90 ...
```

Carrying Out Runs Using mrun

Carrying out a program run typically needs two steps:

1. compilation

```
f95 ... file1.f90 file2.f90 ...
```

2. execution

```
a.out
```

Carrying Out Runs Using mrun

Carrying out a program run typically needs two steps:

1. compilation

```
f95 ... file1.f90 file2.f90 ...
```

2. execution

```
a.out
```

Besides, a program needs input data and creates output data:

Carrying Out Runs Using mrun

Carrying out a program run typically needs two steps:

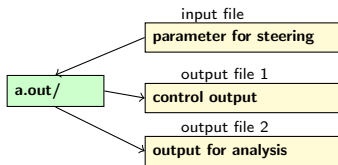
1. compilation

f95 ... file1.f90 file2.f90 ...

2. execution

a.out

Besides, a program needs input data and creates output data:



Carrying Out Runs Using mrun

Carrying out a program run typically needs two steps:

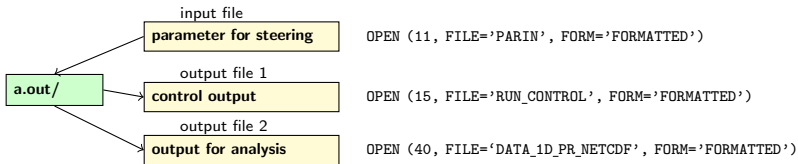
1. compilation

f95 ... file1.f90 file2.f90 ...

2. execution

a.out

Besides, a program needs input data and creates output data:

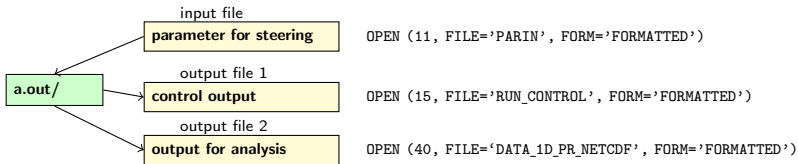


Carrying Out Runs Using mrun

Carrying out a program run typically needs two steps:

1. compilation
f95 ... file1.f90 file2.f90 ...
2. execution
a.out

Besides, a program needs input data and creates output data:



Problems: The user has to copy or rename output files, if he/she wants to run the program more than once and if he concurrently wants to keep the files from the former run(s). If he/she is using different input parameter files for steering, these files also have to be copied into the working directory before the program is executed.

Carrying Out Runs Using mrun

Carrying out a program run typically needs two steps:

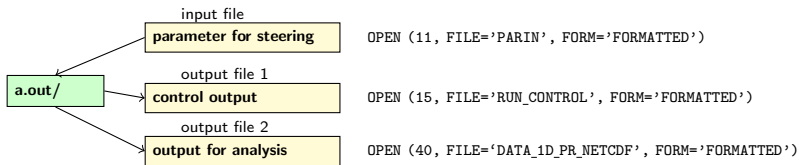
1. compilation

f95 ... file1.f90 file2.f90 ...

2. execution

a.out

Besides, a program needs input data and creates output data:



Problems: The user has to copy or rename output files, if he/she wants to run the program more than once and if he concurrently wants to keep the files from the former run(s). If he/she is using different input parameter files for steering, these files also have to be copied into the working directory before the program is executed.

It is therefore very desirable to automate these tasks!

mrun: Principal Mode of Operation (I)

mrun runs a program (in our case the PALM model) by carrying out the following principle tasks in a sequential order:

mrun: Principal Mode of Operation (I)

mrun runs a program (in our case the PALM model) by carrying out the following principle tasks in a sequential order:

1. Create a temporary working directory and change into this directory:

```
mkdir <tmpdir>/<username>.<randomnumber>  
cd <tmpdir>/<username>.<randomnumber>
```

The path of <tmpdir> is given in the configuration file `.mrun.config` by the environment variable `tmp_user_catalog`, e.g.:

```
%tmp_user_catalog /tmp <hi> parallel
```

mrun: Principal Mode of Operation (I)

mrun runs a program (in our case the PALM model) by carrying out the following principle tasks in a sequential order:

1. Create a temporary working directory and change into this directory:

```
mkdir <tmpdir>/<username>.<randomnumber>
cd <tmpdir>/<username>.<randomnumber>
```

The path of <tmpdir> is given in the configuration file `.mrun.config` by the environment variable `tmp_user_catalog`, e.g.:

```
%tmp_user_catalog /tmp <hi> parallel
```

On Cray-XC30:

```
%tmp_user_catalog /gfs1/work/<username> lcsgh parallel
```

mrun: Principal Mode of Operation (I)

mrun runs a program (in our case the PALM model) by carrying out the following principle tasks in a sequential order:

1. Create a temporary working directory and change into this directory:

```
mkdir <tmpdir>/<username>.<randomnumber>
cd <tmpdir>/<username>.<randomnumber>
```

The path of <tmpdir> is given in the configuration file .mrun.config by the environment variable tmp_user_catalog, e.g.:

```
%tmp_user_catalog /tmp <hi> parallel
```

On Cray-XC30:

```
%tmp_user_catalog /gfs1/work/<username> lcsgh parallel
/<tmpdir>/<username>.<randomnumber>
```

mrun: Principal Mode of Operation (I)

mrun runs a program (in our case the PALM model) by carrying out the following principle tasks in a sequential order:

1. Create a temporary working directory and change into this directory:

```
mkdir <tmpdir>/<username>.<randomnumber>
cd <tmpdir>/<username>.<randomnumber>
```

The path of <tmpdir> is given in the configuration file .mrun.config by the environment variable tmp_user_catalog, e.g.:

```
%tmp_user_catalog /tmp <hi> parallel
```

On Cray-XC30:

```
%tmp_user_catalog /gfs1/work/<username> lcsgh parallel
/<tmpdir>/<username>.<randomnumber>
```

2. Copy the input files from a directory of the user into this working directory:

mrun: Principal Mode of Operation (I)

mrun runs a program (in our case the PALM model) by carrying out the following principle tasks in a sequential order:

1. Create a temporary working directory and change into this directory:

```
mkdir <tmpdir>/<username>.<randomnumber>
cd <tmpdir>/<username>.<randomnumber>
```

The path of <tmpdir> is given in the configuration file .mrun.config by the environment variable tmp_user_catalog, e.g.:

```
%tmp_user_catalog /tmp <hi> parallel
```

On Cray-XC30:

```
%tmp_user_catalog /gfs1/work/<username> lcsgh parallel
/<tmpdir>/<username>.<randomnumber>
```

2. Copy the input files from a directory of the user into this working directory:

```
cp <user_input_file1> /<tmpdir>/<username>.<randomnumber>/<temporary_input_file1>
cp <user_input_file2> /<tmpdir>/<username>.<randomnumber>/<temporary_input_file2>
```

mrun: Principal Mode of Operation (I)

mrun runs a program (in our case the PALM model) by carrying out the following principle tasks in a sequential order:

1. Create a temporary working directory and change into this directory:

```
mkdir <tmpdir>/<username>.<randomnumber>
cd <tmpdir>/<username>.<randomnumber>
```

The path of <tmpdir> is given in the configuration file .mrun.config by the environment variable tmp_user_catalog, e.g.:

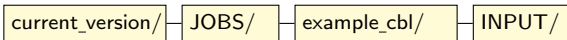
```
%tmp_user_catalog /tmp <hi> parallel
```

On Cray-XC30:

```
%tmp_user_catalog /gfs1/work/<username> lcsgh parallel
/<tmpdir>/<username>.<randomnumber>
```

2. Copy the input files from a directory of the user into this working directory:

```
cp <user_input_file1> /<tmpdir>/<username>.<randomnumber>/<temporary_input_file1>
cp <user_input_file2> /<tmpdir>/<username>.<randomnumber>/<temporary_input_file2>
```



mrun: Principal Mode of Operation (I)

mrun runs a program (in our case the PALM model) by carrying out the following principle tasks in a sequential order:

1. Create a temporary working directory and change into this directory:

```
mkdir <tmpdir>/<username>.<randomnumber>  
cd <tmpdir>/<username>.<randomnumber>
```

The path of <tmpdir> is given in the configuration file .mrun.config by the environment variable tmp_user_catalog, e.g.:

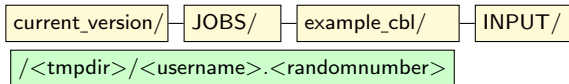
```
%tmp_user_catalog /tmp <hi> parallel
```

On Cray-XC30:

```
%tmp_user_catalog /gfs1/work/<username> lcsgh parallel  
/<tmpdir>/<username>.<randomnumber>
```

2. Copy the input files from a directory of the user into this working directory:

```
cp <user_input_file1> /<tmpdir>/<username>.<randomnumber>/<temporary_input_file1>  
cp <user_input_file2> /<tmpdir>/<username>.<randomnumber>/<temporary_input_file2>
```



mrun: Principal Mode of Operation (I)

mrun runs a program (in our case the PALM model) by carrying out the following principle tasks in a sequential order:

1. Create a temporary working directory and change into this directory:

```
mkdir <tmpdir>/<username>.<randomnumber>
cd <tmpdir>/<username>.<randomnumber>
```

The path of <tmpdir> is given in the configuration file .mrun.config by the environment variable tmp_user_catalog, e.g.:

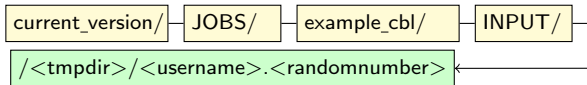
```
%tmp_user_catalog /tmp <hi> parallel
```

On Cray-XC30:

```
%tmp_user_catalog /gfs1/work/<username> lcsgh parallel
/<tmpdir>/<username>.<randomnumber>
```

2. Copy the input files from a directory of the user into this working directory:

```
cp <user_input_file1> /<tmpdir>/<username>.<randomnumber>/<temporary_input_file1>
cp <user_input_file2> /<tmpdir>/<username>.<randomnumber>/<temporary_input_file2>
```

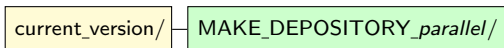


mrun: Principal Mode of Operation (II)

3. Copy the pre-compiled routines to the temporary working directory:

mrun: Principal Mode of Operation (II)

- Copy the pre-compiled routines to the temporary working directory:



mrun: Principal Mode of Operation (II)

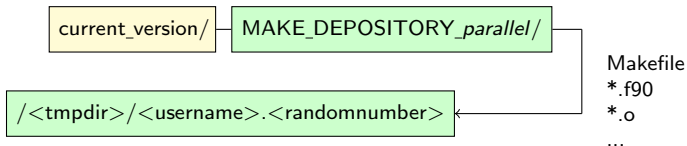
- Copy the pre-compiled routines to the temporary working directory:

current_version/ — MAKE_DEPOSITORY_parallel/

/<tmpdir>/<username>.<randomnumber>

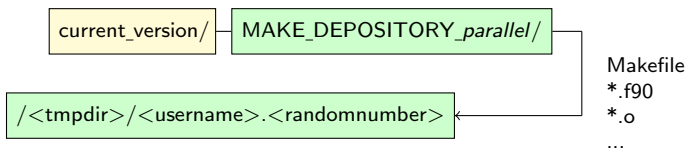
mrun: Principal Mode of Operation (II)

- Copy the pre-compiled routines to the temporary working directory:



mrun: Principal Mode of Operation (II)

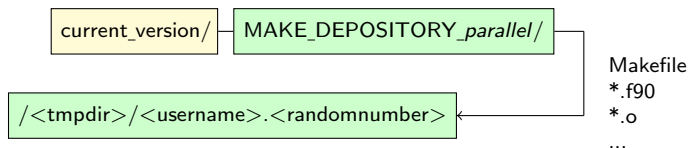
- Copy the pre-compiled routines to the temporary working directory:



- Compile the main program and use pre-compiled object files:
`f95 palm.f90 *.o ... (make Makefile)`

mrun: Principal Mode of Operation (II)

- Copy the pre-compiled routines to the temporary working directory:



- Compile the main program and use pre-compiled object files:
`f95 palm.f90 *.o ... (make Makefile)`
- Execute the program:
`a.out`

mrunc: Principal Mode of Operation (III)

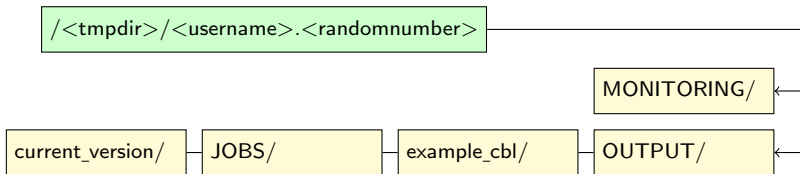
6. Copy the output files from the working directory to a (permanent) directory of the user:

```
cp /<tmpdir>/<username>.<randomnumber>/<temporary_output_file1> <user_output_file1>
cp /<tmpdir>/<username>.<randomnumber>/<temporary_output_file2> <user_output_file2>
```

mruntime: Principal Mode of Operation (III)

- Copy the output files from the working directory to a (permanent) directory of the user:

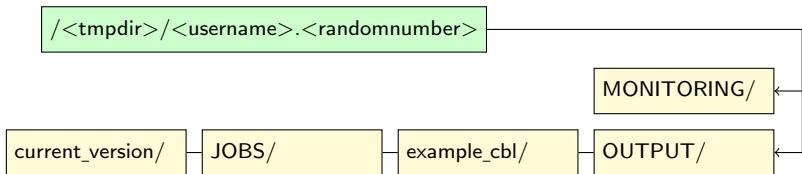
```
cp /<tmpdir>/<username>.<randomnumber>/<temporary_output_file1> <user_output_file1>  
cp /<tmpdir>/<username>.<randomnumber>/<temporary_output_file2> <user_output_file2>
```



mruntime: Principal Mode of Operation (III)

- Copy the output files from the working directory to a (permanent) directory of the user:

```
cp /<tmpdir>/<username>.<randomnumber>/<temporary_output_file1> <user_output_file1>  
cp /<tmpdir>/<username>.<randomnumber>/<temporary_output_file2> <user_output_file2>
```



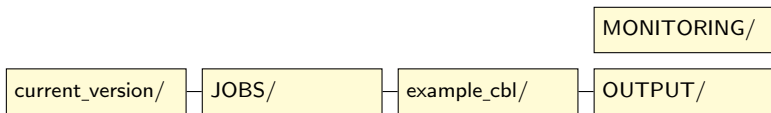
- Delete the temporary working directory

```
rm -rf /<tmpdir>/<username>.<randomnumber>
```

mrun: Principal Mode of Operation (III)

- Copy the output files from the working directory to a (permanent) directory of the user:

```
cp /<tmpdir>/<username>.<randomnumber>/<temporary_output_file1> <user_output_file1>
cp /<tmpdir>/<username>.<randomnumber>/<temporary_output_file2> <user_output_file2>
```



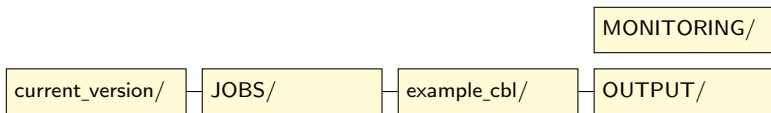
- Delete the temporary working directory

```
rm -rf /<tmpdir>/<username>.<randomnumber>
```

mrun: Principal Mode of Operation (III)

- Copy the output files from the working directory to a (permanent) directory of the user:

```
cp /<tmpdir>/<username>.<randomnumber>/<temporary_output_file1> <user_output_file1>
cp /<tmpdir>/<username>.<randomnumber>/<temporary_output_file2> <user_output_file2>
```



- Delete the temporary working directory

```
rm -rf /<tmpdir>/<username>.<randomnumber>
```

Question: How does `mrun` know which files have to be copied and where from or where to they have to be copied?

Steering File Copy by the Configuration File (I)

Copying of files is controlled by so called *file connection statements*. They connect local files in the temporary working directory with permanent files residing in the directory of the user.

Principle example of a file connection statement (for the PALM parameter file):

```
PARIN      in      d3#      ~/palm/current_version/JOB/INPUT  _p3d      ( nc )
```


Steering File Copy by the Configuration File (I)

Copying of files is controlled by so called *file connection statements*. They connect local files in the temporary working directory with permanent files residing in the directory of the user.

Principle example of a file connection statement (for the PALM parameter file):

```
PARIN      in      d3#      ~/palm/current_version/JOB/INPUT  _p3d      ( nc )
```

↑
local filename in the working directory (must correspond to the filename in the OPEN statement of the program)

Steering File Copy by the Configuration File (I)

Copying of files is controlled by so called *file connection statements*. They connect local files in the temporary working directory with permanent files residing in the directory of the user.

Principle example of a file connection statement (for the PALM parameter file):

```
PARIN      in      d3#      ~/palm/current_version/JOB/INPUT  _p3d      ( nc )
```

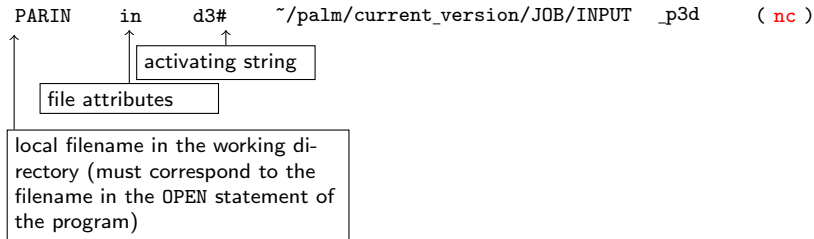
file attributes

local filename in the working directory (must correspond to the filename in the OPEN statement of the program)

Steering File Copy by the Configuration File (I)

Copying of files is controlled by so called *file connection statements*. They connect local files in the temporary working directory with permanent files residing in the directory of the user.

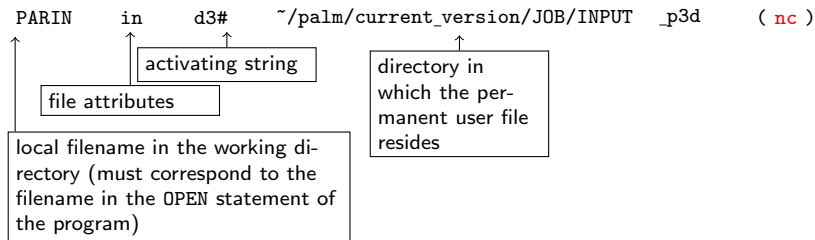
Principle example of a file connection statement (for the PALM parameter file):



Steering File Copy by the Configuration File (I)

Copying of files is controlled by so called *file connection statements*. They connect local files in the temporary working directory with permanent files residing in the directory of the user.

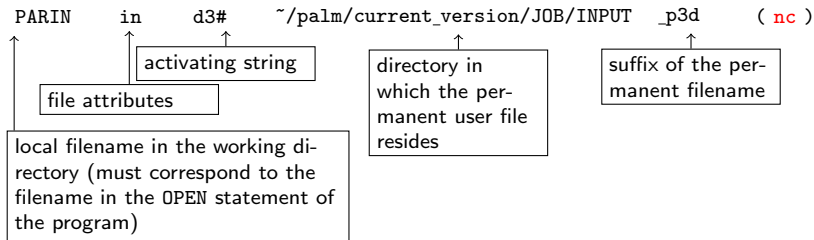
Principle example of a file connection statement (for the PALM parameter file):



Steering File Copy by the Configuration File (I)

Copying of files is controlled by so called *file connection statements*. They connect local files in the temporary working directory with permanent files residing in the directory of the user.

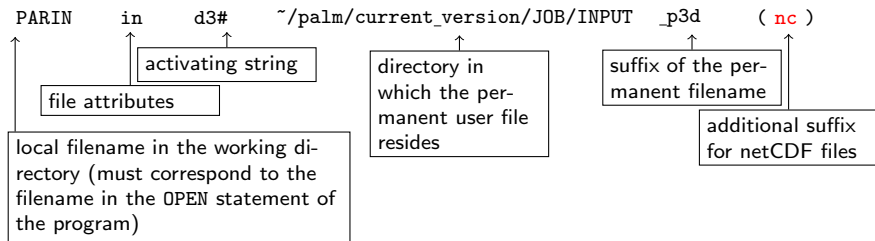
Principle example of a file connection statement (for the PALM parameter file):



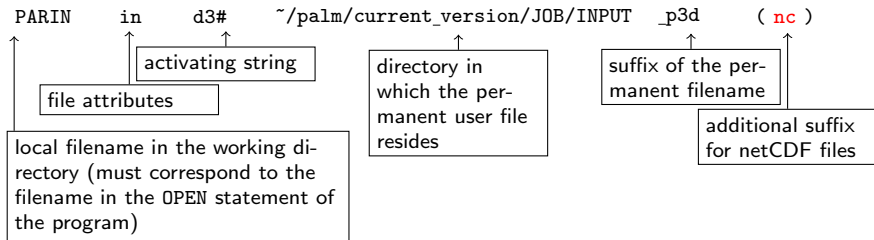
Steering File Copy by the Configuration File (I)

Copying of files is controlled by so called *file connection statements*. They connect local files in the temporary working directory with permanent files residing in the directory of the user.

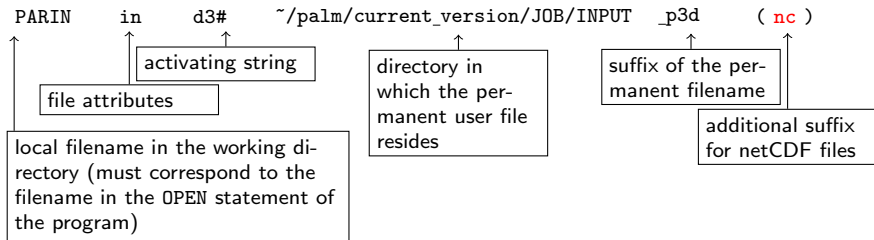
Principle example of a file connection statement (for the PALM parameter file):



Steering File Copy by the Configuration File (II)



Steering File Copy by the Configuration File (II)



The full name of the permanent file results from the directory name, the suffix and the value of mrun-Option `-d`, which defines the so-called **base name** of all files handled by mrun:

```
mrun -d example_cbl ...
```

gives the filename

```
~/palm/current_version/JOB/INPUT/example_cbl_p3d
( ... /example_cbl_p3d.nc )
```


Steering File Copy by the Configuration File (III)

The **base name** can additionally be a part of the directory name by using `$fname` in the directory column of the file connection statement:

```
PARIN in d3# ~/palm/current_version/JOBS/$fname/INPUT _p3d
```

Steering File Copy by the Configuration File (III)

The **base name** can additionally be a part of the directory name by using `$fname` in the directory column of the file connection statement:

```
PARIN in d3# ~/palm/current_version/JOBS/$fname/INPUT _p3d
```

Using the call

```
mrun -d abcde ...
```

the input file will be expected under

```
~/palm/current_version/JOBS/abcde/INPUT/abcde_p3d
```

Steering File Copy by the Configuration File (III)

The **base name** can additionally be a part of the directory name by using `$fname` in the directory column of the file connection statement:

```
PARIN in d3# ~/palm/current_version/JOBS/$fname/INPUT _p3d
```

Using the call

```
mrun -d abcde ...
```

the input file will be expected under

```
~/palm/current_version/JOBS/abcde/INPUT/abcde_p3d
```

In this way, all files handled by the `mrun`-call are stored in the same subdirectory (`abcde/`) and will have the same string (`abcde`) as part of their names, so they can be easily identified as “belonging” to the model run initiated by that `mrun` call.

Steering File Copy by the Configuration File (III)

The **base name** can additionally be a part of the directory name by using `$fname` in the directory column of the file connection statement:

```
PARIN in d3# ~/palm/current_version/JOBS/$fname/INPUT _p3d
```

Using the call

```
mrun -d abcde ...
```

the input file will be expected under

```
~/palm/current_version/JOBS/abcde/INPUT/abcde_p3d
```

In this way, all files handled by the `mrun`-call are stored in the same subdirectory (`abcde/`) and will have the same string (`abcde`) as part of their names, so they can be easily identified as “belonging” to the model run initiated by that `mrun` call.

Instead of always writing the full path name (i.e. `~/palm/current_version/JOBS`), an environment variable can be declared for this at the beginning of the configuration file and be used in the file connection statements:

```
%base_data ~/palm/current_version/JOBS
PARIN in d3# $base_data/$fname/INPUT _p3d
```

Steering File Copy by the Configuration File (III)

The **base name** can additionally be a part of the directory name by using `$fname` in the directory column of the file connection statement:

```
PARIN in d3# ~/palm/current_version/JOBS/$fname/INPUT _p3d
```

Using the call

```
mrun -d abcde ...
```

the input file will be expected under

```
~/palm/current_version/JOBS/abcde/INPUT/abcde_p3d
```

In this way, all files handled by the `mrun`-call are stored in the same subdirectory (`abcde/`) and will have the same string (`abcde`) as part of their names, so they can be easily identified as “belonging” to the model run initiated by that `mrun` call.

Instead of always writing the full path name (i.e. `~/palm/current_version/JOBS`), an environment variable can be declared for this at the beginning of the configuration file and be used in the file connection statements:

```
%base_data ~/palm/current_version/JOBS
PARIN in d3# $base_data/$fname/INPUT _p3d
```

This easily allows to change the directories for all input/output files by just changing the value of `base_data`.

Steering File Copy by the Configuration File (IV)

File connection statements which shall be carried out, have to be activated by giving their activation string in the mrun-option `-r`:

```
PARIN in d3# ~/palm/current_version/JOBS/$fname/INPUT _p3d
```

Steering File Copy by the Configuration File (IV)

File connection statements which shall be carried out, have to be activated by giving their activation string in the mrun-option `-r`:

```
PARIN in d3# ~/palm/current_version/JOBS/$fname/INPUT _p3d
```

The permanent file

```
~/palm/current_version/JOBS/example_cbl/INPUT/example_cbl_p3d
```

will only be copied to the local file PARIN by using the call:

```
mrun -d example_cbl -r "d3#" ...
```

Steering File Copy by the Configuration File (IV)

File connection statements which shall be carried out, have to be activated by giving their activation string in the mrun-option `-r`:

```
PARIN in d3# ~/palm/current_version/JOBS/$fname/INPUT _p3d
```

The permanent file

```
~/palm/current_version/JOBS/example_cbl/INPUT/example_cbl_p3d
```

will only be copied to the local file PARIN by using the call:

```
mrun -d example_cbl -r "d3#" ...
```

Example for an output file:

The file connection statement

```
DATA_1D_PR_NETCDF out:loc pr#
~/palm/current_version/JOBS/$fname/OUTPUT _pr ncs
```


Steering File Copy by the Configuration File (IV)

File connection statements which shall be carried out, have to be activated by giving their activation string in the mrun-option `-r`:

```
PARIN in d3# ~/palm/current_version/JOBS/$fname/INPUT _p3d
```

The permanent file

```
~/palm/current_version/JOBS/example_cbl/INPUT/example_cbl_p3d
```

will only be copied to the local file PARIN by using the call:

```
mrun -d example_cbl -r "d3#" ...
```

Example for an output file:

The file connection statement

```
DATA_1D_PR_NETCDF out:loc pr#
~/palm/current_version/JOBS/$fname/OUTPUT _pr ncs
```

will copy (after program execution) the local file
DATA_1D_PR_NETCDF to the permanent file

```
~/palm/current_version/JOBS/example_cbl/OUTPUT/example_cbl_pr.nc
```

if mrun is called with the options

```
mrun -d example_cbl -r "d3# pr#" ...
```

Steering File Copy by the Configuration File (V)

`mrun` never replaces/overwrites existing files!

Instead, new, so-called file cycles are created.

If an output file, e.g.

```
~/palm/current_version/JOBS/example_cbl/MONITORING/example_cbl_rc
```

has been created from a first call of `mrun` and if the same `mrun` call is submitted again, the second call will not replace the file `example_cbl_rc`, but will create a new file with name:

```
~/palm/current_version/JOBS/example_cbl/MONITORING/example_cbl_rc.1
```

Steering File Copy by the Configuration File (V)

`mrun` never replaces/overwrites existing files!

Instead, new, so-called file cycles are created.

If an output file, e.g.

```
~/palm/current_version/JOBS/example_cbl/MONITORING/example_cbl_rc
```

has been created from a first call of `mrun` and if the same `mrun` call is submitted again, the second call will not replace the file `example_cbl_rc`, but will create a new file with name:

```
~/palm/current_version/JOBS/example_cbl/MONITORING/example_cbl_rc.1
```

In case of netCDF-files, the file cycle number is inserted before the netCDF-suffix `.nc`, e.g.

```
example_cbl_pr.1.nc
```

Steering File Copy by the Configuration File (V)

`mrun` never replaces/overwrites existing files!

Instead, new, so-called file cycles are created.

If an output file, e.g.

```
~/palm/current_version/JOBS/example_cbl/MONITORING/example_cbl_rc
```

has been created from a first call of `mrun` and if the same `mrun` call is submitted again, the second call will not replace the file `example_cbl_rc`, but will create a new file with name:

```
~/palm/current_version/JOBS/example_cbl/MONITORING/example_cbl_rc.1
```

In case of netCDF-files, the file cycle number is inserted before the netCDF-suffix `.nc`, e.g.

```
example_cbl_pr.1.nc
```

The implemented file cycle mechanism does not allow to use any other dots "." in the path or filename:

```
~/palm/version_3.6/JOBS/example.sbl/MONITORING/example.sbl_rc
```

File Connection Statements From the Default .mrun.config file

```

#-----
# List of input-files
#-----
PARIN                in:job          d3#      $base_data/$fname/INPUT    _p3d
PARIN                in:job          d3f      $base_data/$fname/INPUT    _p3df
TOPOGRAPHY_DATA     in:locopt       d3#:d3f  $base_data/$fname/INPUT    _topo
NUDGING_DATA        in:locopt       d3#:d3f  $base_data/$fname/INPUT    _nudge
LSF_DATA            in:locopt       d3#:d3f  $base_data/$fname/INPUT    _lsf
BININ               in:loc:flpe     d3f      $base_data/$fname/RESTART  _d3d
PARTICLE_RESTART_DATA_IN in:loc:flpe    prtff    $base_data/$fname/RESTART  _rprt
DATA_1D_PR_NETCDF   in:locopt       prf      $base_data/$fname/OUTPUT    _pr      nc
DATA_1D_SP_NETCDF   in:locopt       spf      $base_data/$fname/OUTPUT    _sp      nc
DATA_1D_TS_NETCDF   in:locopt       tsf      $base_data/$fname/OUTPUT    _ts      nc
DATA_1D_PTS_NETCDF  in:locopt       ptsf     $base_data/$fname/OUTPUT    _pts     nc
DATA_2D_XY_NETCDF   in:locopt       xyf      $base_data/$fname/OUTPUT    _xy      nc
DATA_2D_XY_AV_NETCDF in:locopt       xyf      $base_data/$fname/OUTPUT    _xy_av   nc
DATA_2D_XZ_NETCDF   in:locopt       xzf      $base_data/$fname/OUTPUT    _xz      nc
DATA_2D_YZ_NETCDF   in:locopt       yzf      $base_data/$fname/OUTPUT    _yz      nc
DATA_3D_NETCDF      in:locopt       3df      $base_data/$fname/OUTPUT    _3d      nc
DATA_PRT_NETCDF     in:locopt:pe    prtff    $base_data/$fname/OUTPUT    _prt

#
#-----
# List of output-files
#-----
BINOUT              out:loc:flpe    restart  $base_data/$fname/RESTART  _d3d
PARTICLE_RESTART_DATA_OUT out:loc:flpe    prt#:prtff $base_data/$fname/RESTART  _rprt

#
RUN_CONTROL         out:loc:tr      d3#      $base_data/$fname/MONITORING _rc
IN_CONTROL         out:loc:tra     d3f      $base_data/$fname/MONITORING _rc
HEADER             out:loc:tr      d3#      $base_data/$fname/MONITORING _header

```

Additional Features of mrun (I)

- ▶ Generating batch jobs on local **and** remote host.

Additional Features of mrun (I)

- ▶ Generating batch jobs on local **and** remote host.
- ▶ Setting of unix environment variables for job and model control (e.g. for determining compiler options, etc.).

Additional Features of mrun (I)

- ▶ Generating batch jobs on local **and** remote host.
- ▶ Setting of unix environment variables for job and model control (e.g. for determining compiler options, etc.).
- ▶ Values have to be set in the mrun configuration file `.mrun.config`:

Additional Features of mrun (I)

- ▶ Generating batch jobs on local **and** remote host.
- ▶ Setting of unix environment variables for job and model control (e.g. for determining compiler options, etc.).
- ▶ Values have to be set in the mrun configuration file `.mrun.config`:

```
%tmp_user_catalog      /gfs1/work/<replace by your HLRN-III username>      lccrayh parallel
%tmp_data_catalog      /gfs1/work/<replace by your HLRN-III username>      lccrayh parallel
%compiler_name         ftn                             lccrayh parallel
%compiler_name_ser     ftn                             lccrayh parallel
%cpp_options           -e:Z:-DMPI_REAL=MPI_DOUBLE_PRECISION:-DMPI_2REAL=MPI_2DOUBLE_PRECISION:...
%mopts                -j:4                             lccrayh parallel
%fopts                -em:-O3:-hnoomp:-hfp3:-hdynamic                    lccrayh parallel
%lopts                -em:-O3:-hnoomp:-hfp3:-hdynamic                    lccrayh parallel
%remote_username       <replace by your HLRN-III username>                  lccrayh parallel
%memory                2300                                       lccrayh parallel
%modules               fftw:cray-hdf5-parallel:cray-netcdf-hdf5parallel lccrayh parallel
```

Additional Features of mrun (II)

- ▶ User-defined unix commands are carried out before or after execution of the model (input/output commands) or in case of errors during the compile- or run-step (error commands). These commands can also be defined in the `mrun` configuration file:

Additional Features of mrun (II)

- ▶ User-defined unix commands are carried out before or after execution of the model (input/output commands) or in case of errors during the compile- or run-step (error commands). These commands can also be defined in the `mrun` configuration file:

```
IC:echo 'PALM will soon start to execute in \${PWD}' > /home/h/niksiraa/mrun_messages
EC:ls -al
OC:tar -cf DEBUG.tar DEBUG_*
```

Additional Features of mrun (II)

- ▶ User-defined unix commands are carried out before or after execution of the model (input/output commands) or in case of errors during the compile- or run-step (error commands). These commands can also be defined in the `mrun` configuration file:

```
IC:echo 'PALM will soon start to execute in \${PWD}' > /home/h/niksiraa/mrun_messages
EC:ls -al
OC:tar -cf DEBUG.tar DEBUG_*
```

- ▶ Automatic generation of restart jobs.

PALM *Interactive* Example Run Using mrun: Tracing the Run by the User

start run on local machine:


```
mrun -d example_cbl -h lcmuknb ...
```

PALM *Interactive* Example Run Using mrun: Tracing the Run by the User

start run on local machine:

```
mrun -d example_cbl -h lcmuknb ...
```

temporary working directory
is created, all required files
are copied there



PALM *Interactive* Example Run Using mrun: Tracing the Run by the User

start run on local machine:

```
mrun -d example_cbl -h lcmuknb ...
```

temporary working directory
is created, all required files
are copied there

follow run messages on terminal

PALM *Interactive* Example Run Using mrun: Tracing the Run by the User

start run on local machine:

```
mrun -d example_cbl -h lcmuknb ...
```

temporary working directory
is created, all required files
are copied there

follow run messages on terminal

as soon as message

```
*** execution starts in directory
    "<tmpdir>"
```

you can change to this directory (in a
new terminal) and watch the progress
of timesteps

```
cd <tmpdir>
tail -f RUN_CONTROL
```

However, this might be difficult in case
of short run times, because the run may
have finished before you have entered
the commands!

PALM *Batch* Example Run (on HLRN, remote) Using mrun: Tracing the Run by the User

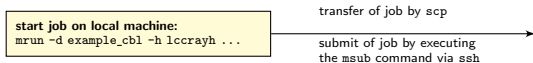
- ▶ Reminder: Running batch jobs requires a directory `/job_queue` for the job protocol files on the local and remote host.

start job on local machine:

```
mrun -d example_cbl -h lccrayh ...
```

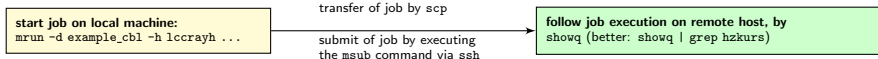
PALM *Batch* Example Run (on HLRN, remote) Using mrun: Tracing the Run by the User

- ▶ Reminder: Running batch jobs requires a directory `/job_queue` for the job protocol files on the local and remote host.



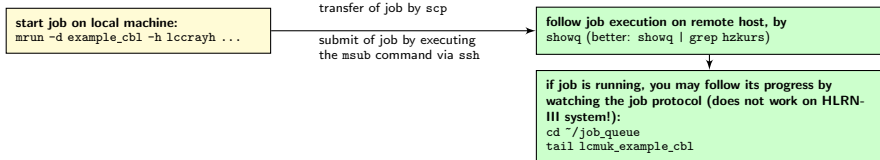
PALM *Batch* Example Run (on HLRN, remote) Using mrun: Tracing the Run by the User

- ▶ Reminder: Running batch jobs requires a directory `/job_queue` for the job protocol files on the local and remote host.



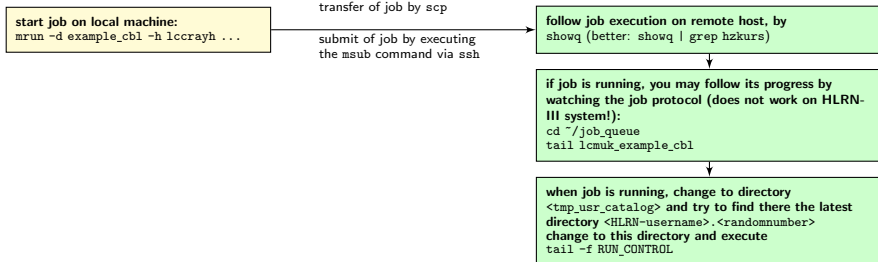
PALM *Batch* Example Run (on HLRN, remote) Using mrun: Tracing the Run by the User

- ▶ Reminder: Running batch jobs requires a directory `/job_queue` for the job protocol files on the local and remote host.



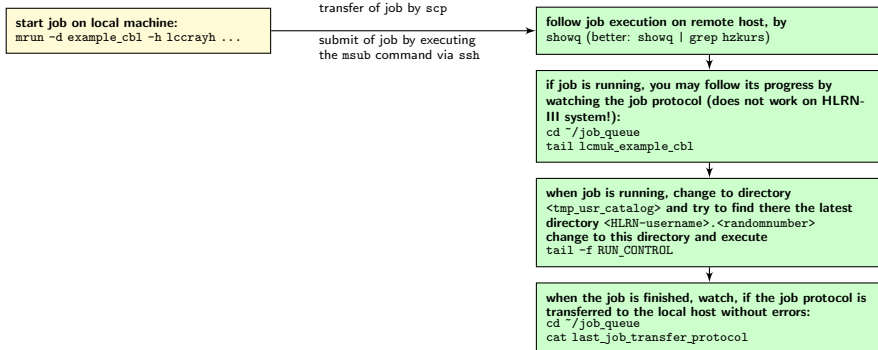
PALM *Batch* Example Run (on HLRN, remote) Using mrun: Tracing the Run by the User

- ▶ Reminder: Running batch jobs requires a directory `/job_queue` for the job protocol files on the local and remote host.



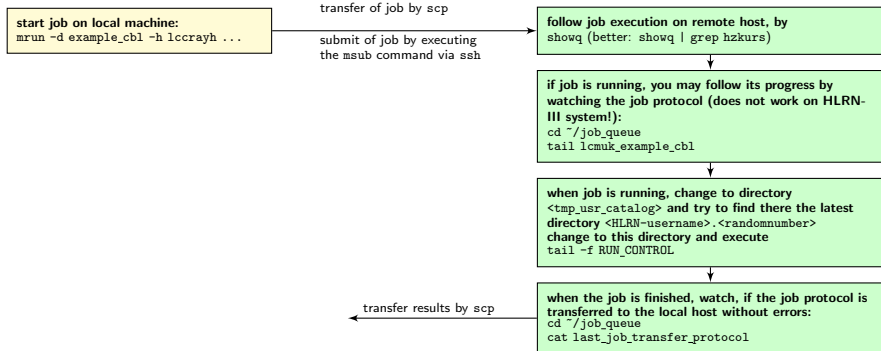
PALM *Batch* Example Run (on HLRN, remote) Using mrun: Tracing the Run by the User

- ▶ Reminder: Running batch jobs requires a directory `/job_queue` for the job protocol files on the local and remote host.



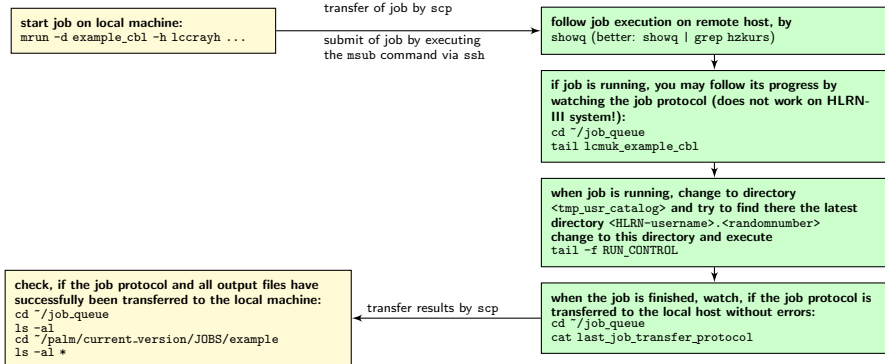
PALM *Batch* Example Run (on HLRN, remote) Using mrun: Tracing the Run by the User

- ▶ Reminder: Running batch jobs requires a directory `/job_queue` for the job protocol files on the local and remote host.



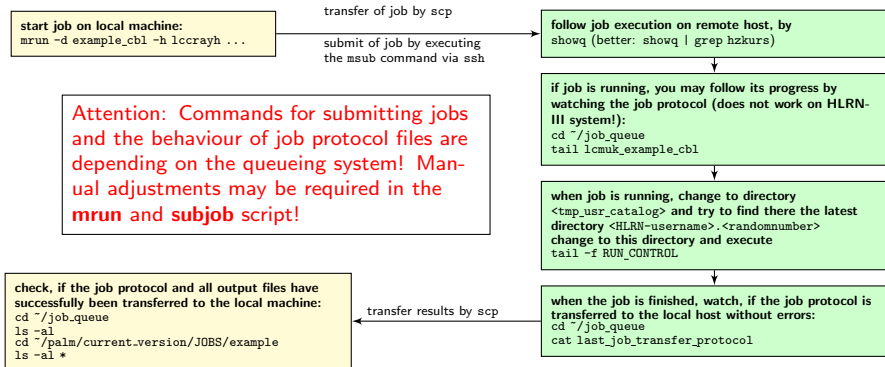
PALM *Batch* Example Run (on HLRN, remote) Using mrun: Tracing the Run by the User

- ▶ Reminder: Running batch jobs requires a directory `/job_queue` for the job protocol files on the local and remote host.



PALM Batch Example Run (on HLRN, remote) Using mrun: Tracing the Run by the User

- ▶ Reminder: Running batch jobs requires a directory `/job_queue` for the job protocol files on the local and remote host.



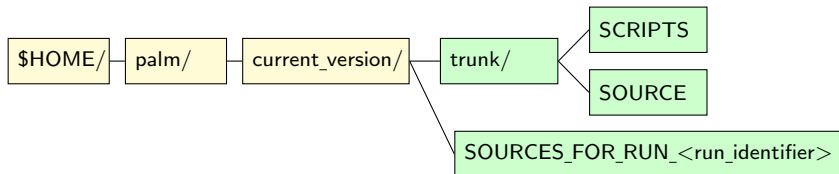
PALM Runs Using mrun: Further Details

PALM Runs Using mrun: Further Details

- ▶ For every (initial) run, `mr`un creates a source directory which contains copies of

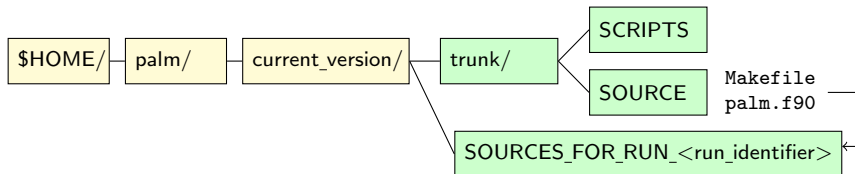
PALM Runs Using mrun: Further Details

- ▶ For every (initial) run, `mrun` creates a source directory which contains copies of



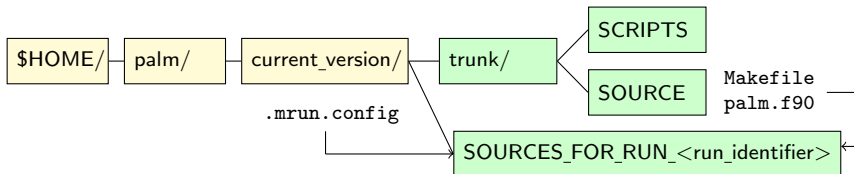
PALM Runs Using mrun: Further Details

- ▶ For every (initial) run, `mrun` creates a source directory which contains copies of
 - ▶ those source code files to be translated (always the main program, `palm.f90`, plus user interface files)
 - ▶ the Makefile



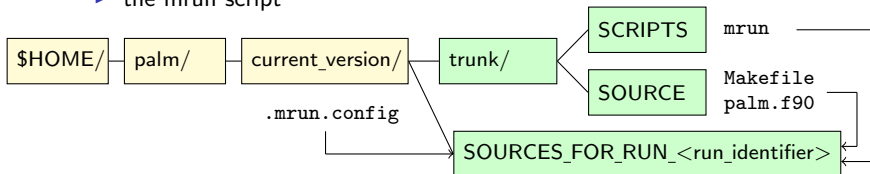
PALM Runs Using mrun: Further Details

- ▶ For every (initial) run, `mrun` creates a source directory which contains copies of
 - ▶ those source code files to be translated (always the main program, `palm.f90`, plus user interface files)
 - ▶ the Makefile
 - ▶ the configuration file



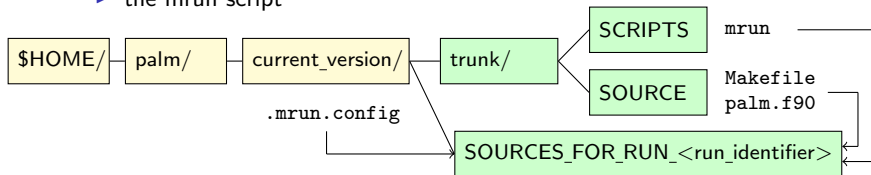
PALM Runs Using mrun: Further Details

- ▶ For every (initial) run, mrun creates a source directory which contains copies of
 - ▶ those source code files to be translated (always the main program, palm.f90, plus user interface files)
 - ▶ the Makefile
 - ▶ the configuration file
 - ▶ the mrun script



PALM Runs Using mrun: Further Details

- ▶ For every (initial) run, mrun creates a source directory which contains copies of
 - ▶ those source code files to be translated (always the main program, palm.f90, plus user interface files)
 - ▶ the Makefile
 - ▶ the configuration file
 - ▶ the mrun script



- ▶ These files are used in the run/job. They are also used by restart jobs, which guarantees, that all jobs in a job chain are using the same information. Please never modify these directories, unless you exactly know, what you are doing.

PALM Runs Using mrun: most important mrun options

`mr`un

PALM Runs Using mrun: most important mrun options

mrun

- ▶ `-d example_cbl`

run identifier, basefile/directory name of all data files of this run

PALM Runs Using mrun: most important mrun options

mrun

- ▶ **-d example_cbl**
run identifier, basefile/directory name of all data files of this run
- ▶ **-h host identifier**
host on which the run shall be performed (reminder: this is not the UNIX hostname):
lcmuknb, lcmuk, lccrayh...

PALM Runs Using mrun: most important mrun options

mrun

- ▶ **-d example_cbl**
run identifier, basefile/directory name of all data files of this run
- ▶ **-h host identifier**
host on which the run shall be performed (reminder: this is not the UNIX hostname):
lcmuknb, lcmuk, lccrayh...
- ▶ **-K parallel**
run in parallel on several processor cores

PALM Runs Using mrun: most important mrun options

mrun

- ▶ **-d example_cbl**
run identifier, basefile/directory name of all data files of this run
- ▶ **-h host identifier**
host on which the run shall be performed (reminder: this is not the UNIX hostname):
lcmuknb, lcmuk, lccrayh...
- ▶ **-K parallel**
run in parallel on several processor cores
- ▶ **-X 4**
number of processors to be used

PALM Runs Using mrun: most important mrun options

mrun

- ▶ **-d example_cbl**
run identifier, basefile/directory name of all data files of this run
- ▶ **-h host identifier**
host on which the run shall be performed (reminder: this is not the UNIX hostname):
lcmuknb, lcmuk, lccrayh...
- ▶ **-K parallel**
run in parallel on several processor cores
- ▶ **-X 4**
number of processors to be used
- ▶ **-T 4**
number of tasks to be started on one node of the computer

PALM Runs Using mrun: most important mrun options

mrun

- ▶ **-d example_cbl**
run identifier, basefile/directory name of all data files of this run
- ▶ **-h host identifier**
host on which the run shall be performed (reminder: this is not the UNIX hostname):
lcmuknb, lcmuk, lccrayh...
- ▶ **-K parallel**
run in parallel on several processor cores
- ▶ **-X 4**
number of processors to be used
- ▶ **-T 4**
number of tasks to be started on one node of the computer
- ▶ **-t 100**
CPU time allowed to be used for the batch jobs

PALM Runs Using mrun: most important mrun options

mrun

- ▶ **-d example_cbl**
run identifier, basefile/directory name of all data files of this run
- ▶ **-h host identifier**
host on which the run shall be performed (reminder: this is not the UNIX hostname):
lcmuknb, lcmuk, lccrayh...
- ▶ **-K parallel**
run in parallel on several processor cores
- ▶ **-X 4**
number of processors to be used
- ▶ **-T 4**
number of tasks to be started on one node of the computer
- ▶ **-t 100**
CPU time allowed to be used for the batch jobs
- ▶ **-b**
run in batch mode

PALM Runs Using mrun: most important mrun options

mrun

- ▶ **-d example_cbl**
run identifier, basefile/directory name of all data files of this run
- ▶ **-h host identifier**
host on which the run shall be performed (reminder: this is not the UNIX hostname):
lcmuknb, lcmuk, lccrayh...
- ▶ **-K parallel**
run in parallel on several processor cores
- ▶ **-X 4**
number of processors to be used
- ▶ **-T 4**
number of tasks to be started on one node of the computer
- ▶ **-t 100**
CPU time allowed to be used for the batch jobs
- ▶ **-b**
run in batch mode
- ▶ **-q mpp1q**
batch job queue

PALM Runs Using mrun: most important mrun options

mrun

- ▶ **-d example_cbl**
run identifier, basefile/directory name of all data files of this run
- ▶ **-h host identifier**
host on which the run shall be performed (reminder: this is not the UNIX hostname):
lcmuknb, lcmuk, lccrayh...
- ▶ **-K parallel**
run in parallel on several processor cores
- ▶ **-X 4**
number of processors to be used
- ▶ **-T 4**
number of tasks to be started on one node of the computer
- ▶ **-t 100**
CPU time allowed to be used for the batch jobs
- ▶ **-b**
run in batch mode
- ▶ **-q mpp1q**
batch job queue
- ▶ **-r "..."**
activation strings