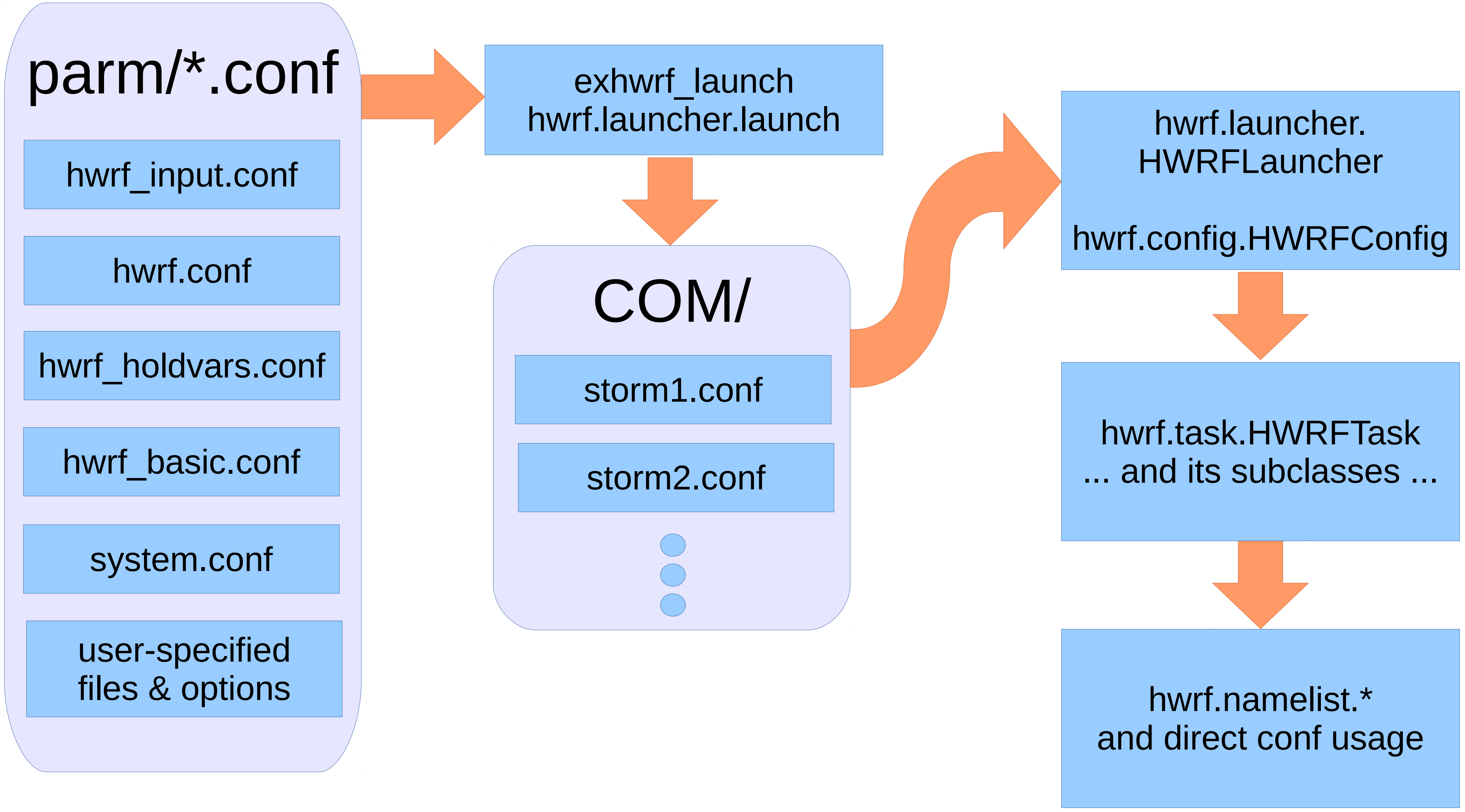
Configuring the HWRF

Overview


parm/*.conf
hwrf_input.conf
hwrf.conf
hwrf_holdvars.conf
hwrf_basic.conf
system.conf
user-specified files & options
exhwrf_launch hwrf.launcher.launch
COM/
storm1.conf
storm2.conf
hwrf.launcher. HWRFLauncher
hwrf.config.HWRFConfig
hwrf.task.HWRFTask ... and its subclasses ...
hwrf.namelist.* and direct conf usage

Unix .conf Files

Format

parm/*.conf

- hwrf_input.conf
- hwrf.conf
- hwrf_holdvars.conf
- hwrf_basic.conf
- system.conf
- user-specified files & options

- Simple format:

```
# This is a comment
[section]
key=value ; This is also a comment
key2=value2
```

Unix .conf Files

Doxygen Documentation

parm/*.conf

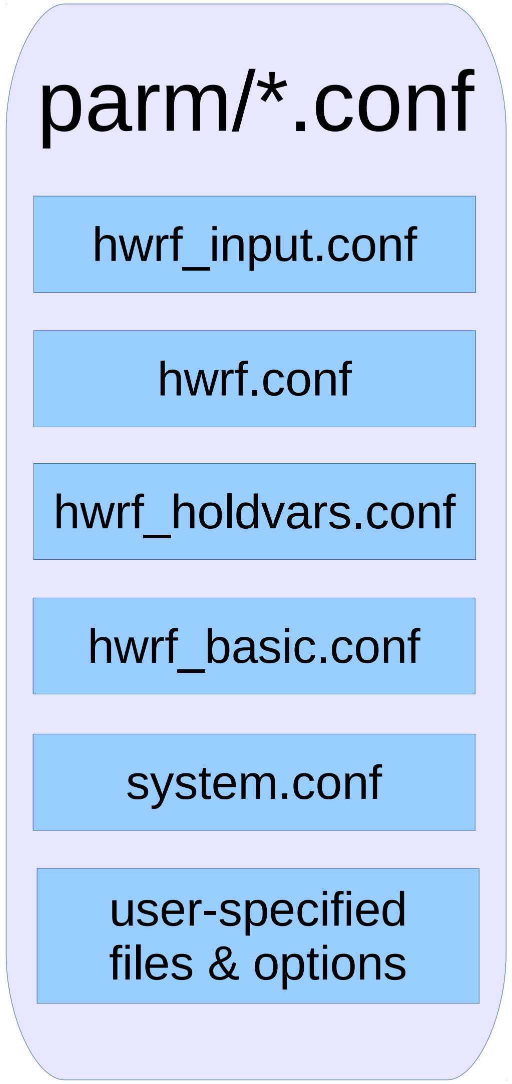
- hwrf_input.conf
- hwrf.conf
- hwrf_holdvars.conf
- hwrf_basic.conf
- system.conf
- user-specified files & options

- Documentation format:

```
## Short description of section
#
# Long description of section
# @note Doxygen+markdown syntax
[section]
key=value ;; short description
## Short description of key2
#
# long description of key2
key2=value2
```

Unix .conf Files

Python String Substitution



- String substitution:

```
[myprog]
basedir = /path/to/basedir
exename = myprog
exepath = {basedir}/exec/{exename}
```

- Key exepath in **[myprog]** expands to this string:
  - /path/to/basedir/exec/myprog

Unix .conf Files

Python String Substitution

parm/*.conf

- hwrf_input.conf
- hwrf.conf
- hwrf_holdvars.conf
- hwrf_basic.conf
- system.conf
- user-specified files & options

- String substitution with formatting:

```
[myprog]
basedir = /path/to/basedir
gridnum = 5
exename = myprog_grid_{gridnum:02d}
exepath = {basedir}/exec/{exename}
```

- Key exepath in **[myprog]** is:
  - /path/to/basedir/exec/myprog_grid_05
- C-style (printf) formatting codes

Unix .conf Files

HWRF Extensions

parm/*.conf

- hwrf_input.conf
- hwrf.conf
- hwrf_holdvars.conf
- hwrf_basic.conf
- system.conf
- user-specified files & options

- Substitute from specified section:

```
[grid]
num = 5

[myprog]
basedir = /path/to/basedir
exename = myprog_grid_{grid/num:02d}
exepath = {basedir}/exec/{exename}
```

- Key exepath in **[myprog]** is:
  - /path/to/basedir/exec/myprog_grid_05
  - **grid**/num = **[grid]** section num key

Unix .conf Files
HWRF Extensions

parm/*.conf

hwrf_input.conf

hwrf.conf

hwrf_holdvars.conf

hwrf_basic.conf

system.conf

user-specified files & options

- Substitute from **[config]** and **[dir]** if not in local section.

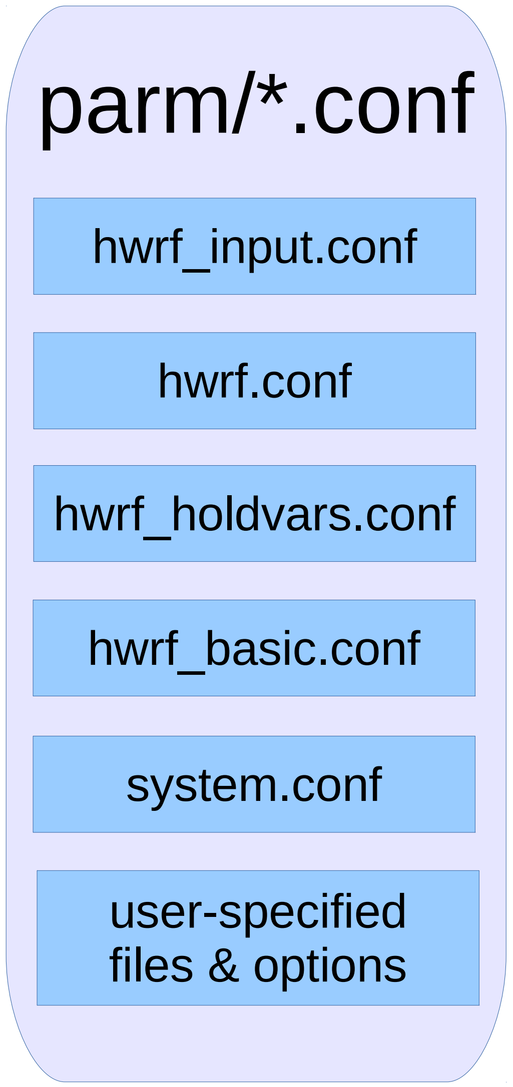
```
[dir]
basedir = /path/to/basedir

[config]
gridnum = 5

[myprog]
exepath = {basedir}/exec/{exename}
exename = myprog_grid_{gridnum:02d}
```

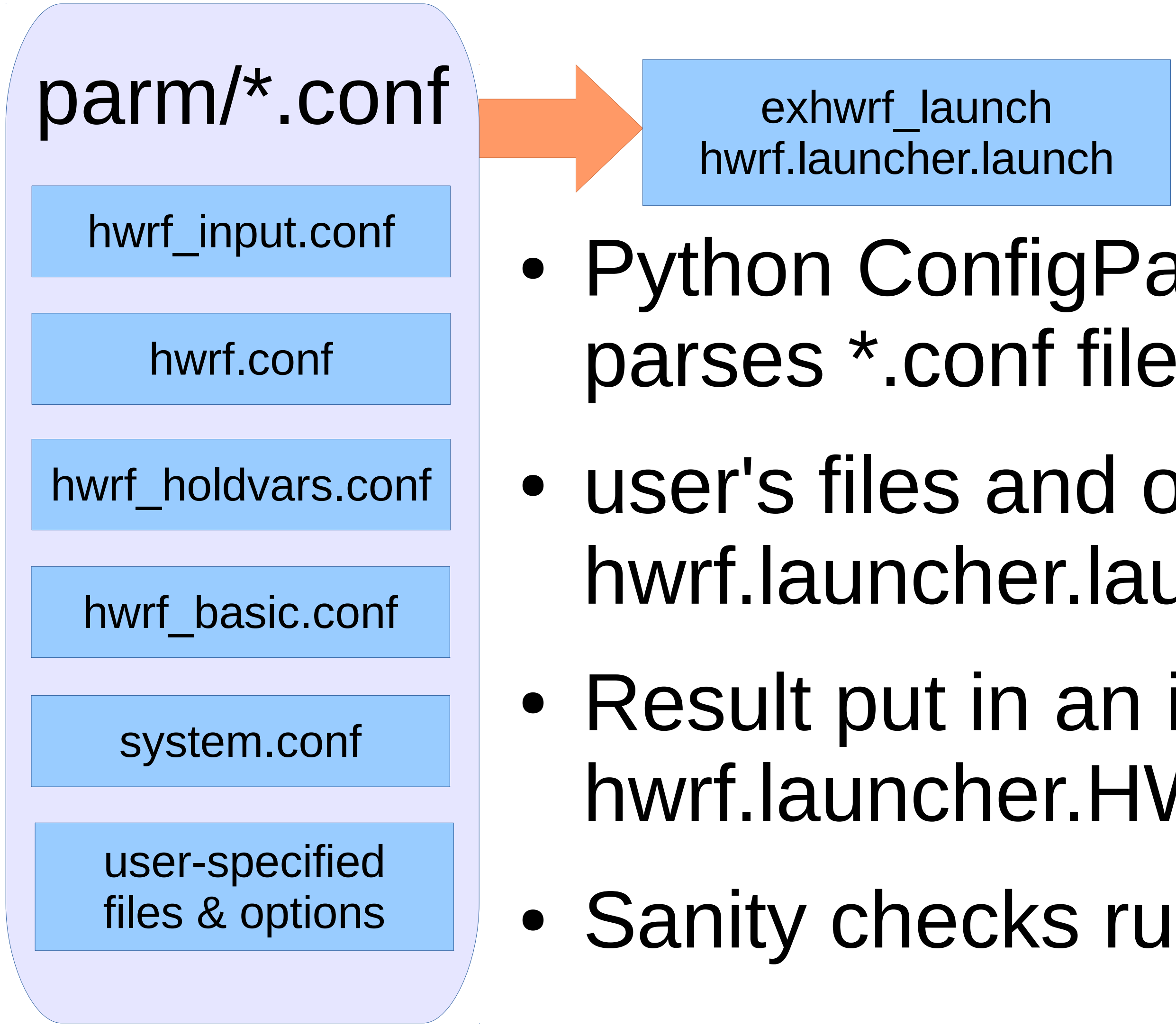
- Key exepath in **[myprog]** is:
  - /path/to/basedir/exec/myprog_grid_05
  - basedir from **[dir]**, gridnum from **[config]**

File Ordering



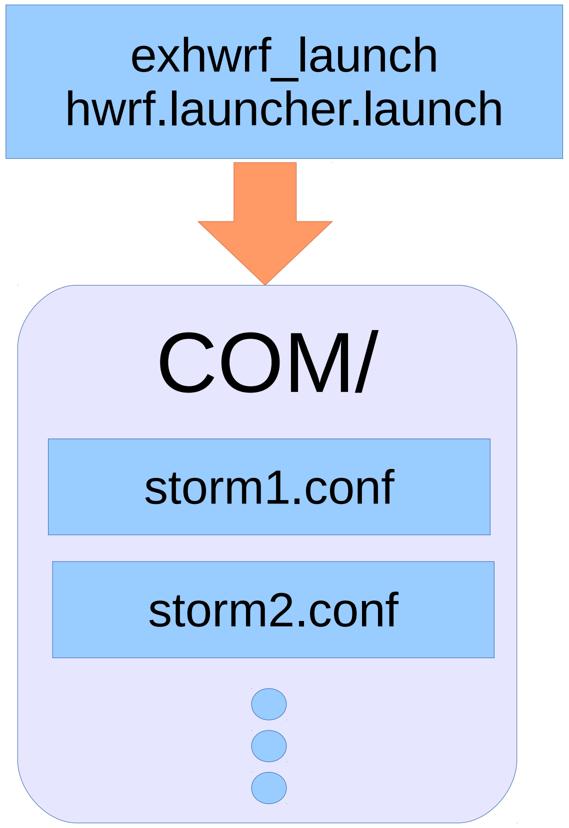
- Files read in order
- Options in later files overrides earlier.
- system.conf: per-system (Jet, WCOSS, S4, etc.) overrides
- user-specified-options: hourly output, alternate microphysics, etc.
  - user-specified files read before options
  - ../path/to/my.conf
  - **section**.key=value

Config Processing



- Python ConfigParser.ConfigParser parses *.conf files in order, then
- user's files and options sent to hwrf.launcher.launch() added.
- Result put in an in-memory hwrf.launcher.HWRFLauncher object
- Sanity checks run on configuration.

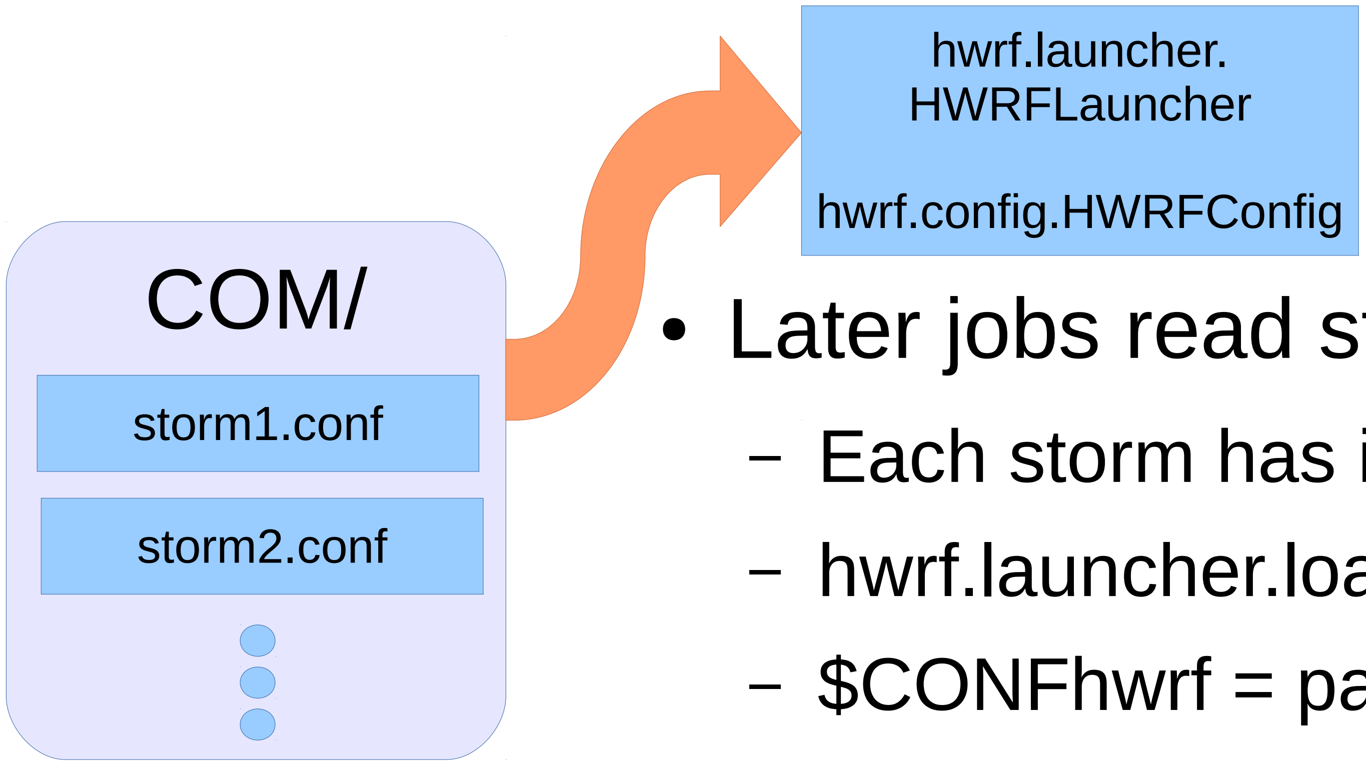
Make storm*.conf
hwrf.launcher.launch()



- exhwrf_launch writes storm*.conf
- Contains processed config data
  - Later jobs only read the storm*.conf and never process other conf info
- Edit storm*.conf to make per-cycle changes, such as emergency
- Operational example:
  - GFS ENKF failed in operations.
  - Add **[config]** run_gsi=no for one cycle
  - HWRF forecast run without data assim

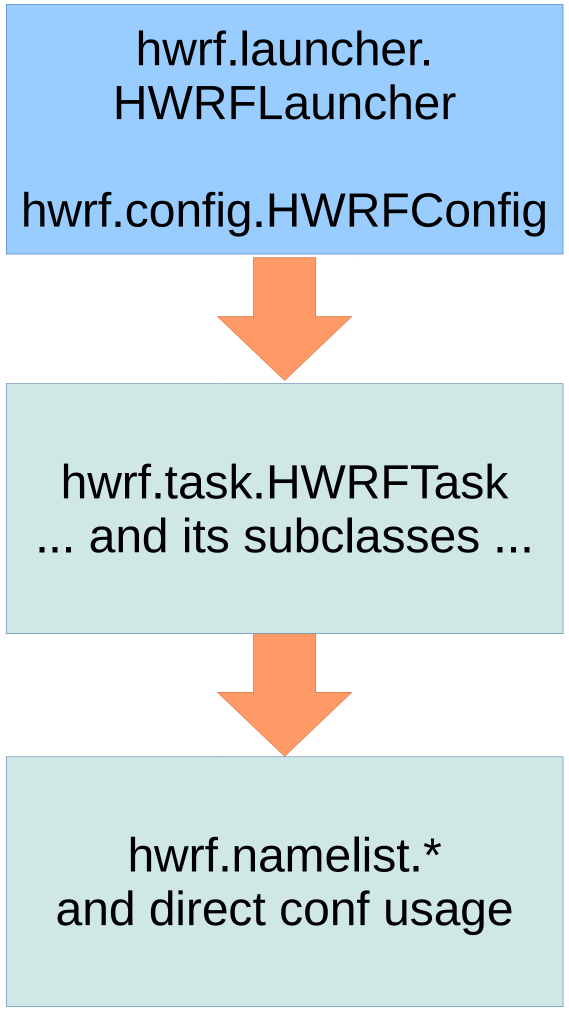
Load storm*.conf

hwrf.launcher.load()



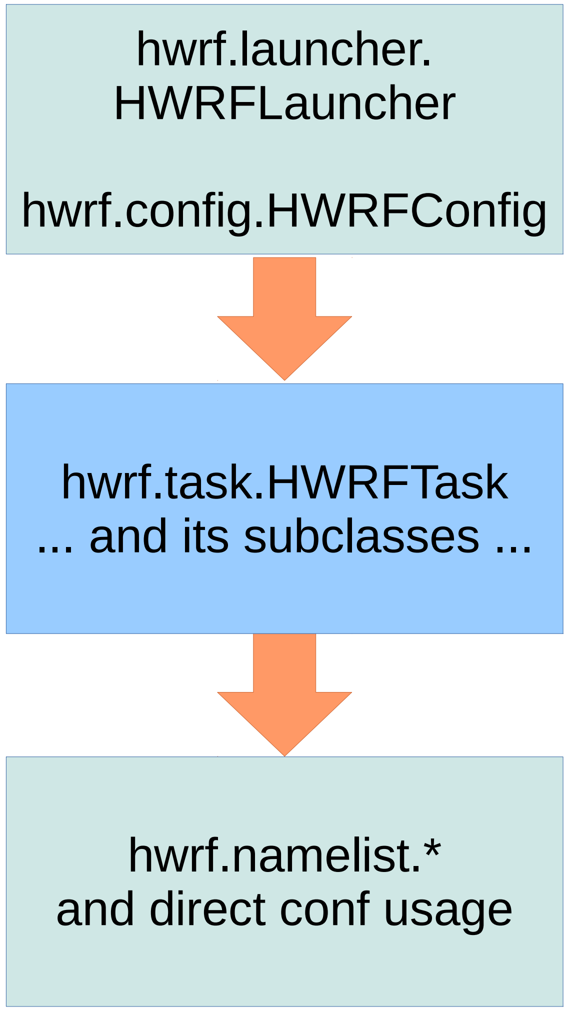
- Later jobs read storm*.conf
  - Each storm has its own *.conf file
  - hwrf.launcher.load()
  - $CONFhwrf = path to storm*.conf
- hwrf.launcher.HWRFLauncher
  - subclass of hwrf.config.HWRFConfig
  - Contains many convenience functions for accessing conf info

HWRFLauncher/HWRFConfig



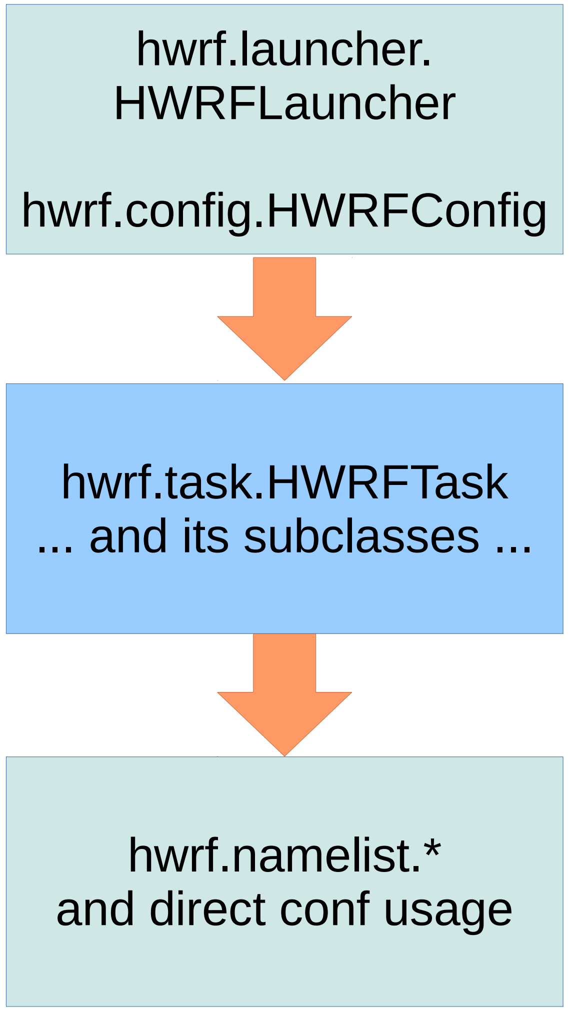
- HWRFLauncher/HWRFConfig.
  - Classes that access conf data.
  - getstr(section,key) => value
    - Raise exception if key unspecified
  - getstr(section,key,default)
    - Return default if key is unspecified
  - getint, getfloat, etc.
    - Specified return types
  - cycle - forecast cycle (property)
  - (see docs or *.py for full list)
- Most conf access is through **HWRFTask** (next slide)

HWRFTask



- Represents one task to be performed
  - GeogridTask, WRFAtmos, etc.
- Has a database taskname
  - self.taskname
  - More on database in a later presentation.
- Has a conf section (self.section)
- Has an HWRFConfig (self.conf)
- Default: task name = section name

HWRFTask

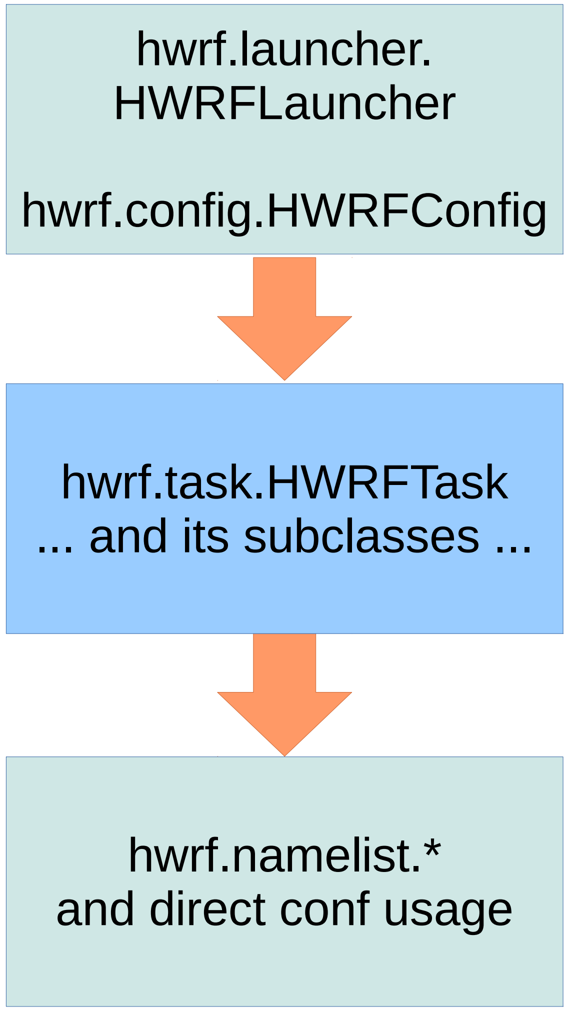


- Aliases:
  - confstr(key)

    = task.conf.getstr(task.section,key)

    - String value for key in my conf section.
    - Raise exception if missing.
  - confstr(key,default)
    - Same, but return default if missing
  - confbool, conffloat, etc.
    - Alternate datatypes

HWRFTask



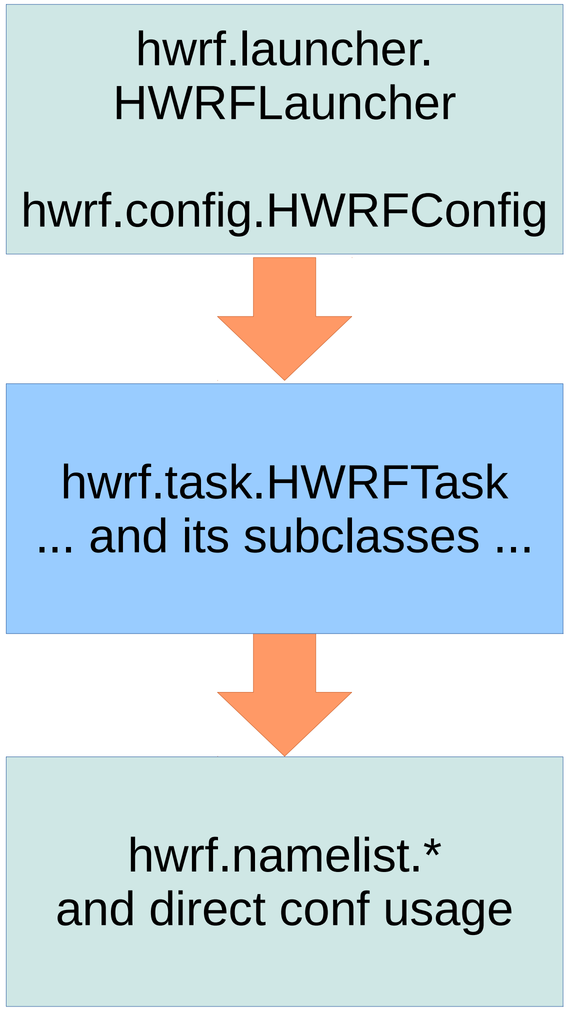
- task.icstr(“{FIXhwrf}/grid{gridnum}”
  - Expand string with substitution

```
[dir]
FIXhwrf = {HOMEhwrf}/fix/
HOMEhwrf = /path/to/install/dir

[mytask]
gridnum = 5
```

  - Returns “/path/to/install/dir/fix/grid5”

HWRFTask

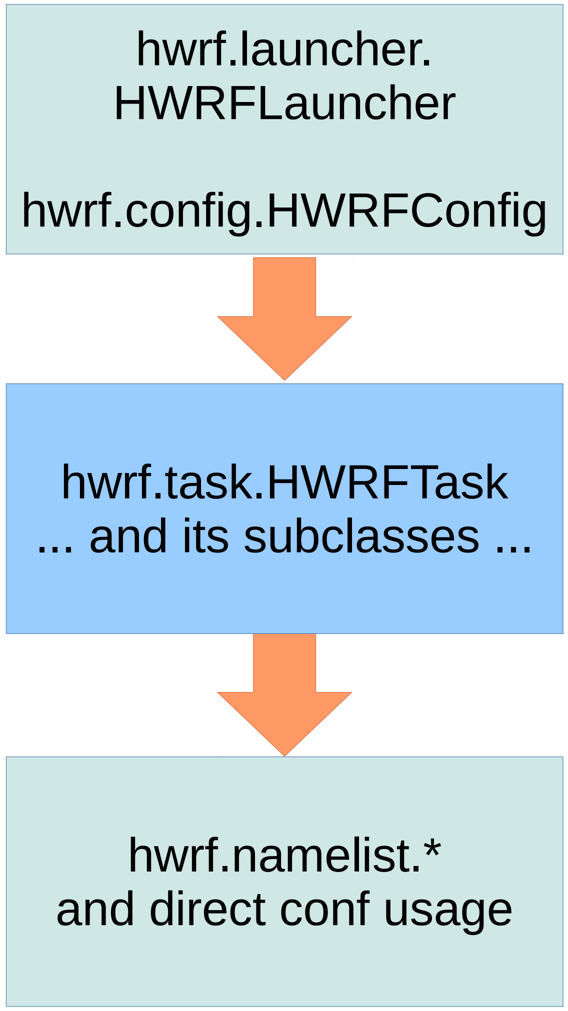


- task.timestr("{FIXhwrf}/month{aMM}.dat",
  ftime,atime,...)
  - String substitution with fcst/analysis time info
    - atime is optional, defaults to self.conf.cycle

```
[dir]
FIXhwrf = {HOMEhwrf}/fix/
```

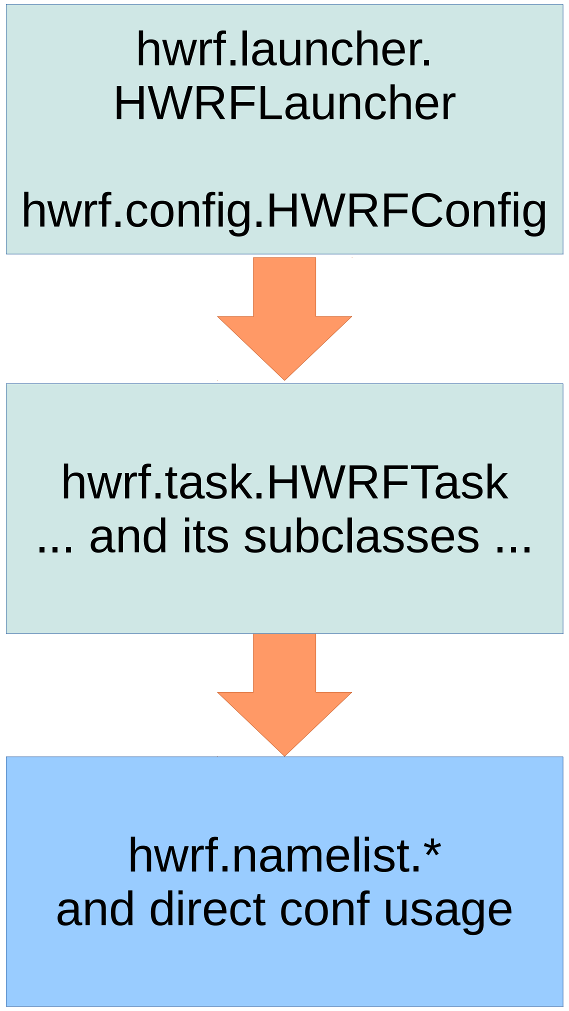
  - Returns "/path/to/install/dir/fix/month08.dat" for a cycle in August.
  - See hwrf.config.HWRFConfig.set_time_vars for list.

HWRFTask



- Shortcuts:
  - getexe(“wrf”) - **[exe]** section wrf key
  - getdir(“FIXgsi”) - **[dir]** FIXgsi key
  - getloc('syndat') - syndat in **[exe]** or **[dir]**
- Taskvars:
  - Object-local key=value for string expansion
  - task.tvset(key,value) - set object-local value
  - task.tvget(key) - get object-local value
  - task.tvdel(key) - delete object-local value
  - task.taskvars - dict of object-local values
  - task.tvhave(key) - check for object-local value

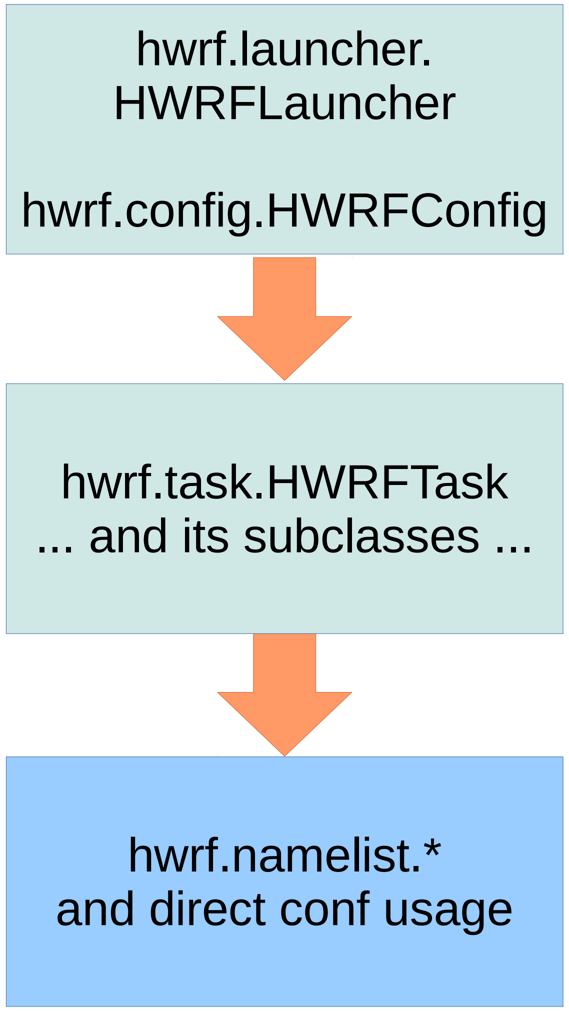
hwrf.namelist.NamelistInserter



- Insert config information into a file
  - Intended for Fortran namelists, but can work with other text files
  - Converts Python datatypes to Fortran Namelist datatypes

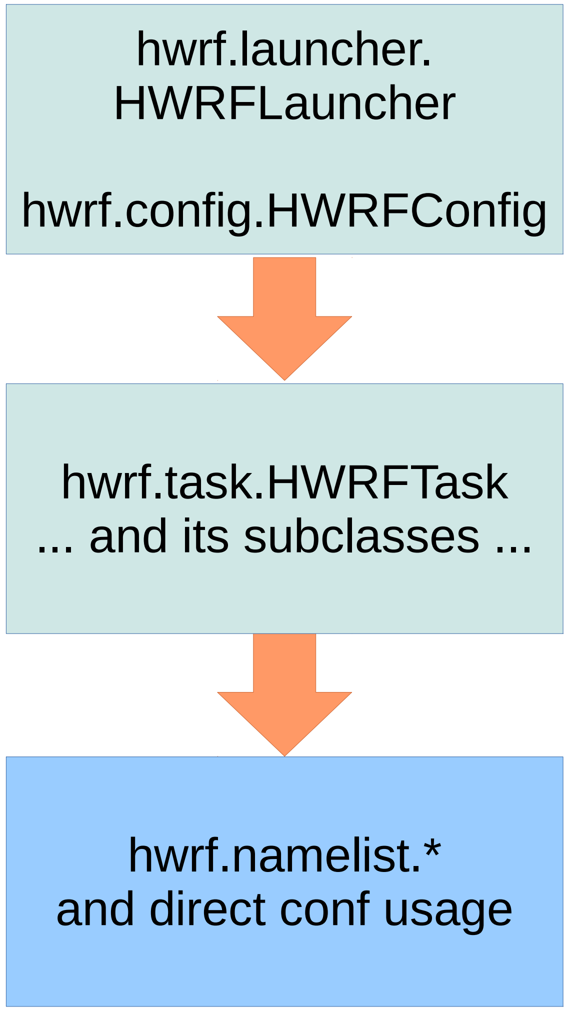
```
hwrf.conf:  [conf]
hwrf.conf:  var=T,F,T
myfile.nl:  &nl
myfile.nl:  myvar=<var>/
output:     &nl
output:     myvar=.true.,.false.,.true./
```

hwrf.namelist.NamelistInserter



- Specify datatype for conversion:
  - float <f:var> or <r:var>
  - integer <i:var>
  - String <s:var>
  - Bool <b:var> or <l:var>
  - Date/time <d:var>
    - YYYY-MM-DD_HH:MM:SS
- Insert without type conversion:
  - <u:var>
- Guess datatype:
  - <var>

hwrf.namelist.Conf2Namelist



- Make Fortran namelist from conf data.
- No input text file; only needs conf data.
- Can merge multiple Conf2Namelist together to make Fortran arrays
  - used to merge multiple domains' data when making the WRF namelist.

hwrf.namelist.Conf2Namelist

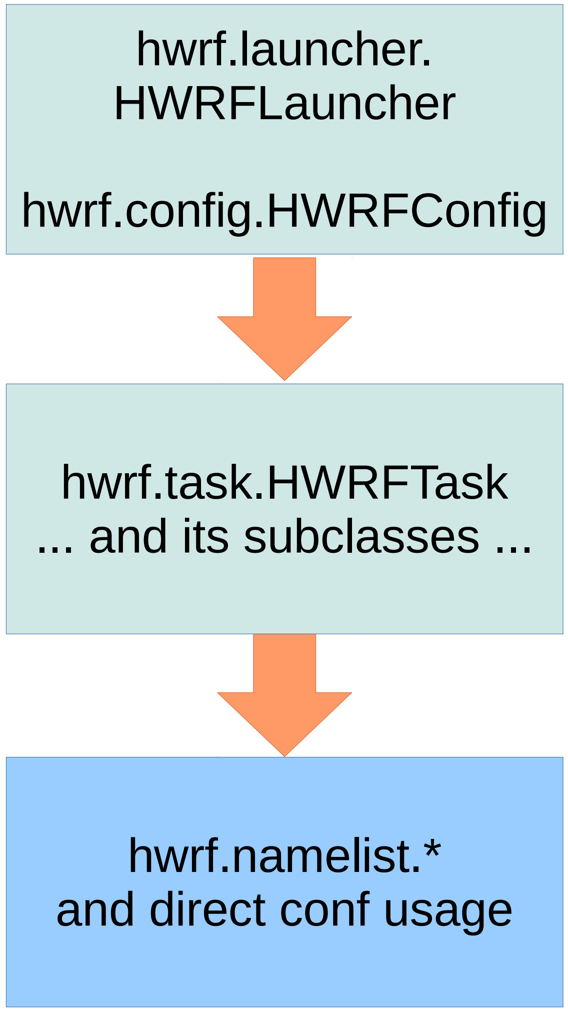
Single Domain Example

```
conf=RawConfigParser()
conf.readfp(StringIO('''
[sec1]
physics.mp_physics=85
physics.cu_physics=84
namelist=sec2,sec3
[sec2]
physics.cu_physics=4
physics.bl_pbl_physics=93
[sec3]
physics.bl_pbl_physics=3
    ''')
str(Conf2Namelist(
    conf,'sec1'))
```

hwrf.namelist.*
and direct conf usage

```
&physics
  bl_pbl_physics = 3
  cu_physics = 84
  mp_physics = 85
/
```

HWRF Input Data
DataCatalog

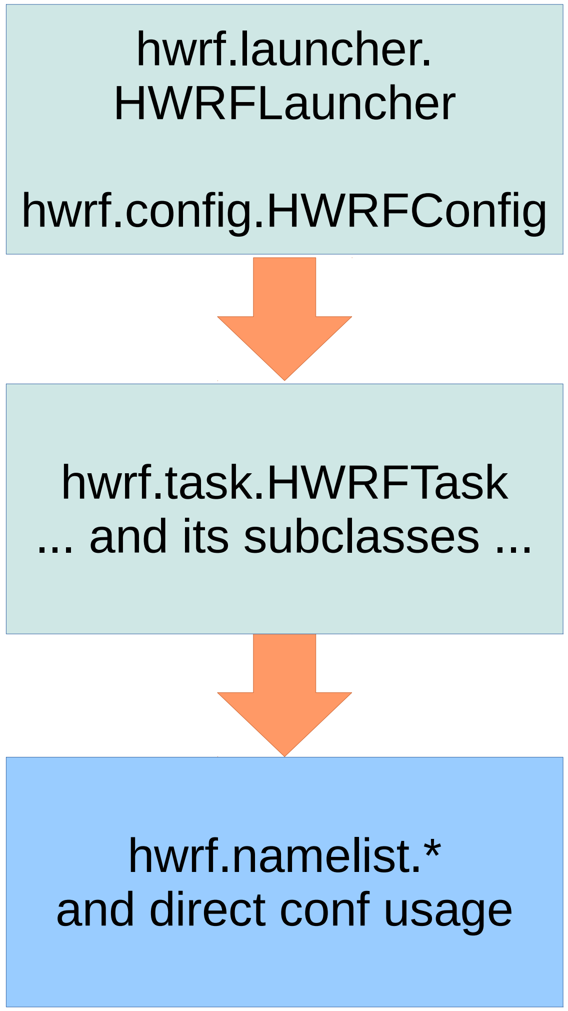


- Specifies location of inputs to HWRF
  - GDAS, GFS, GEFS, GFS ENKF, etc.
- Example: GFS spectral forecast file in [**hwrfdata**] section:

```
[hwrfdata]
inputroot={WORKhwrf}/hwrfdata
gfs={inputroot}/gfs.{aYMDH}
gfs_sf={gfs}/gfs.t{aHH}z.sf{fahr:02d}
```

HWRF Input Data

InputSource

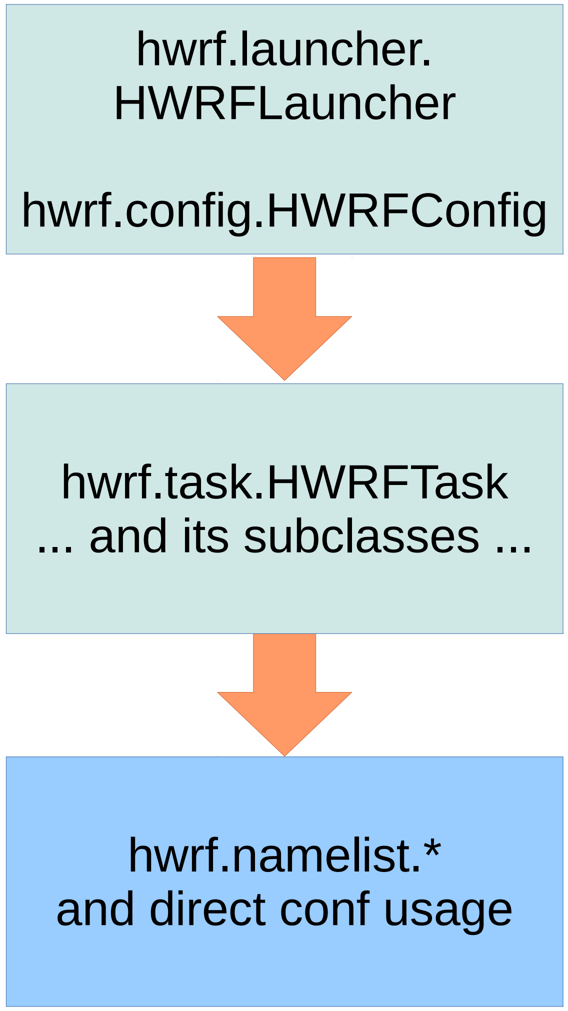


- Lists available input data (DataCatalogs), order in which they can be attempted, and valid cycles.
- Jet example:

```
[jet_sources]
jet_hist%location = file:///
jet_hist%histprio = 90
```

- exhwrf_input job will look at the **[jet_hist]** section to find input data.
  - ...histprio=90 used to decide which other input sources are tried first.
  - Higher priority sources are tried first.

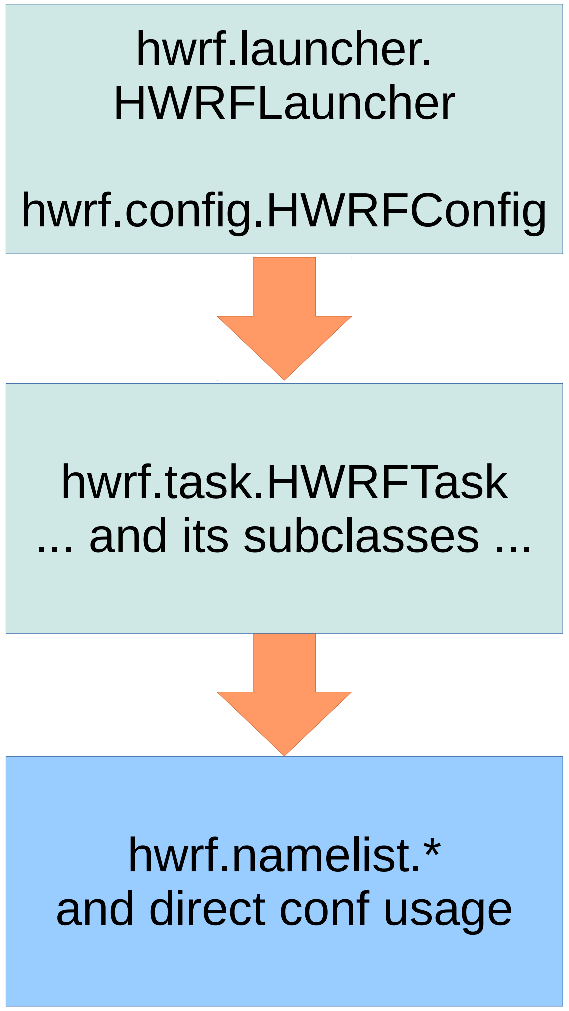
WRF Configuration



- Driven by Conf2Namelist
  - bold are conf sections

```
storm1outer=WRFDomain(conf,'storm1outer')
storm1inner=WRFDomain(conf,'storm1inner')
moad=WRFDomain(conf,'moad')
wrf=WRFSimulation(
   conf,'wrf',moad,conf.cycle,
   to_datetime_rel(126*3600,conf.cycle))
wrf.add(storm1outer,moad)
wrf.add(storm1inner,storm1outer)
```

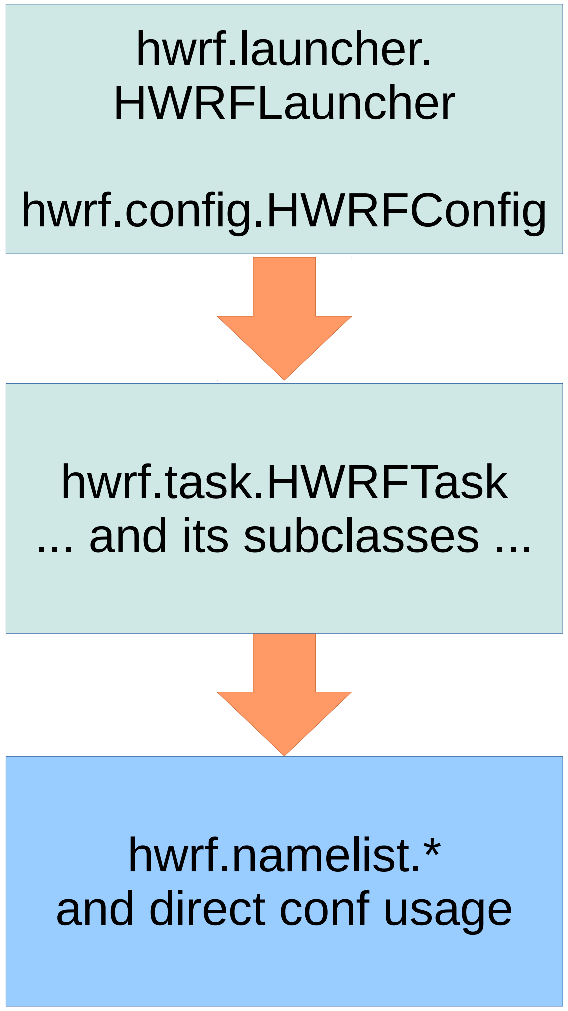
WRF Configuration



- Set information that is not domain-specific:

```
wrf=WRFSimulation(
   conf,'wrf',moad,conf.cycle,
   to_datetime_rel(126*3600,conf.cycle))
[wrf]
dt = 38+4/7
bdystep = 21600
ptsgm = 15000
ptop = 200
prep_hybrid = .true.
io_form = 11
namelist = wrf_namelist
```

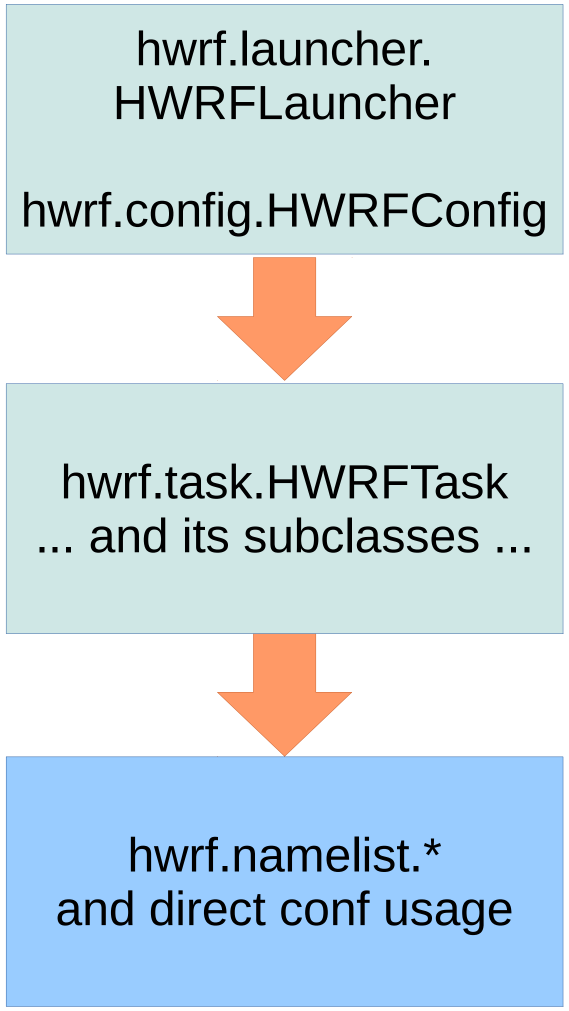
WRF Configuration



- Namelist settings that are not domain-specific:

```
wrf=WRFSimulation(
   conf,'wrf',moad,conf.cycle,
   to_datetime_rel(126*3600,conf.cycle))
[wrf_namelist]
physics.var_ric = 1.0
physics.num_soil_layers = 4
dynamics.euler_adv = .False.
bdy_control.spec_bdy_width = 1
...
```

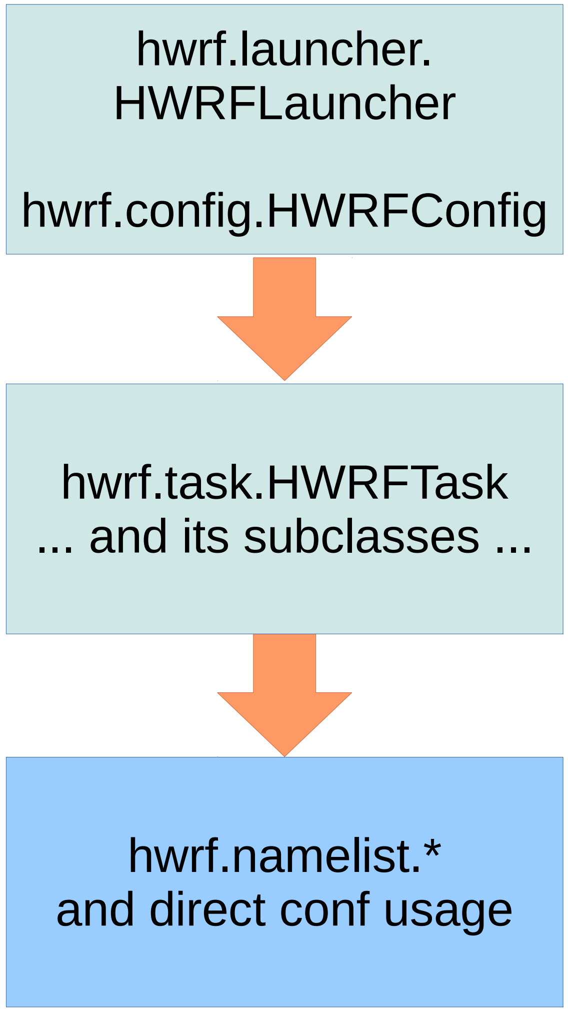
WRF Configuration



- Domain-specific information
  - Namelist info in separate section.

```
moad=WRFDomain(conf,'moad')
[moad]
nx = 288
ny = 576
parent_grid_ratio = 1
dx = 0.135
dy = 0.135
start = moad
namelist = moad_namelist
```

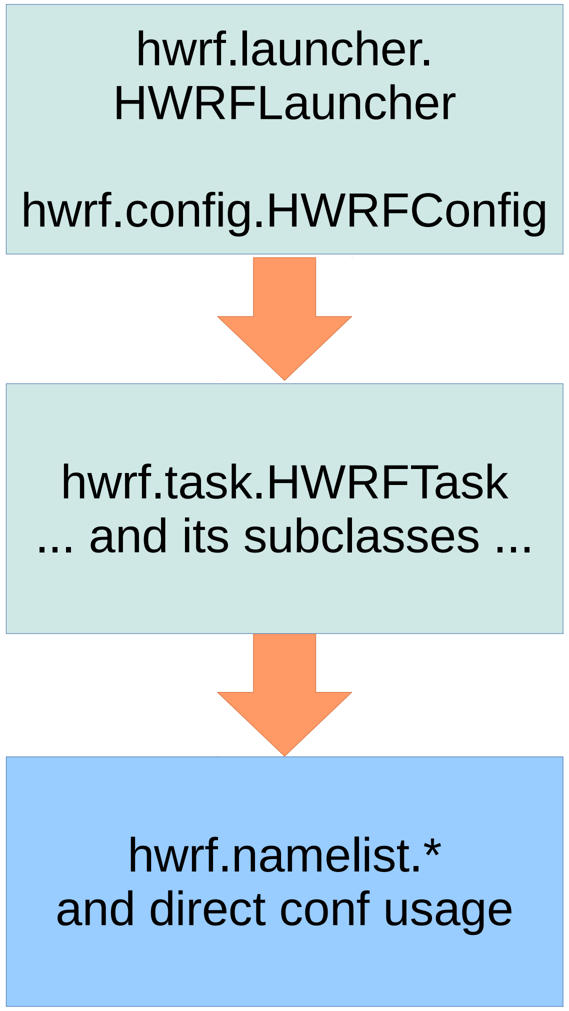
WRF Configuration



- Namelist information for one domain.
  - If settings are in parent domain but not child, they are copied from parent.

```
[moad_namelist]
physics.mp_physics = 5
physics.ra_lw_physics = 4
physics.ra_sw_physics = 4
physics.sf_sfclay_physics = 88
physics.sf_surface_physics = 2
physics.bl_pbl_physics = 3
```

WRF Configuration



- Auto-centering

```
storm1outer=WRFDomain(conf,'storm1outer')

storm1inner=WRFDomain(conf,'storm1inner')

[storm1outer]
nx = 142
ny = 274
...
start = auto ;; Center on child or wrfanl

[storm1inner]
nx = 265
ny = 472
...
start = centered ;; Center on storm or wrfanl
```

?