Source code overhaul

Jacob Weismann Poulsen, DMI

http://benchmark.dmi.dk/

Type <Space> to get on with the show!

Outline for the presentation

- Preprocessing
- Libraries and interfacing
- Subroutine/function
- Bug hunting
- Other overhaul considerations
- My local framework (*only* if time permits it)

Preprocessing

- Overhaul ideas
  - Identify and divide the current CPPFLAGS into 3 classes
  - Get rid of obsolete CPPFLAGS and corr. dead code
  - Collapse dual flags into one
  - Fix inconsistent meanings, e.g. PREC32
  - Handle environment flags automatically
- Proof_of_concept results (for hlprog)

  (please do not take these numbers too serious)
  - 85% (and corr. code) can go (8/7 in total, 6/5 of them are optional)
  - Positive side effect: got rid of GC

Libraries

- Overhaul ideas
  - Get rid of files with unused functions/subroutines
  - Get rid of unused include files
  - Get rid of multiple defined include and source files
  - Get rid of multiple defined functions/subroutines
  - BEWARE: Fix inconsistencies in multiple definitions
  - Review the structural design of the libraries (draw, please)
  - Review the interfacing (use compilers)
    - Look for parameter inconsistencies (numbers, types)

Libraries II

- Examples (not complete)

symbol	source code 1	source code 2
cheby_	grdy/cheby.F	grdy/dfcoef.F
dolph_	grdy/dolph.F	grdy/dfcoef.F
char2int_var_	obssupport/char2int.F	hlcmaio/char2int.F
cleanup_	strgen4/cleanup.F	librt7/cleanup.F90
dum_dsteqr_	common/lapack/dsteqr.F	hlhelp/dsteqr.F
dum_ssyev_	common/lapack/ssyev.F	hlhelp/ssyev.F
gbyte_	common/pbio/gbyte.c	port/gbyte.F
gbytes_	common/pbio/gbyte.c	port/gbytes.F
int2char_	port/int2char.F	obssupport/int2char.F
modiswindtocma_	cmatoobs/modiswindtocma.F	cmatoobs/modistocma.F
regrot_	util/regrot.F	hlhelp/regrot.F
sbyte_	common/pbio/gbyte.c	port/sbyte.F
setpar_	common/pbio/setpar.c	gcod/setpar.F
svp_	obop/chkprf.F	librt7/supsat.F90
swab_	port/swab.F	hlhelp/swab.F
turnwi_	util/turnwi.F	hlhelp/turnwi.F
yomslphy_	mfphys/modules/yomslphy.F	mfphys/modules/mslphy.F

Libraries III

Before (circular references):

```
MAIN
    grdy
        util
        phys
            grdy
                ...
        prpo
            phys
                ...
            grdy
                ...
            vari
            tsfs
                grwl
                    port
                        gcod
```

After (no circular references):

```
MAIN
    grdy
        phys
            util
        mpp
        prpo
            util
            mpp
            vari
            tsfs
            grwl
                port
                    gcod
```

Subroutine/function I

- Add implicit none and fix type issues
- Get rid of compiler warnings
- Get rid of declared but unused variables
- Get rid of assigned but unused variables
- Get rid of unused parameters

Subroutine/function II

- Look for obsolete constructs, e.g. equivalence, computed GOTO, arithmetic IF
- Look for dangerous constructs, e.g. mixing of types in common blocks
- Look for unintentional (?) handling of parameters
  - Add intent on parameters
- Comments
  - Get rid of HTML tags
  - Get rid of obsolete/debug comments (code)
  - Anonymize code, i.e. get rid of author and signatures, e.g. cfoo<date>, ...
- Consistent Fortran90 style
  - THIS MUST BE DONE AUTOMATICALLY, i.e. we must state precisely what we want iff we want a more consistent code style, cf. backup slides

Bug hunting

- Look for uninitialized variables on the stack
- Look for uninitialized variables on the heap
- Look for array bound issues (type-castings OK, but...)
- Compare the results of the optional cases: serial, MPI, MPI+HGS,...
- Look for IEEE versus brain-dead tuning (On) differences

Other overhaul ideas I

Support 32/32 and 64/32 (and less interesting 64/64) precision in computations (not trivial)

Rewrite halo swap subroutines using Fortran instead of CPP :)

Rewrite **simple** C stuff to Fortran to avoid unnecessary complexity, e.g. no need to implement *second()* in C

Collect global definitions in structs (fortran type) or modules, e.g. for global communication and for global dimensions:

```
mype, nproc, nxproc, nyproc
idatastart, jdatastart
atbase, attop, atright, atleft


klon_global, klat_global, klev_global
nhalo, nslon, nelon, nslat, nelat
nacdg, nacdg2, ndiffx, ndiffy
ntask, nhorph, lserial
```

Other overhaul ideas II

Standardized dimensions, e.g.:

```
nlon, klon, nx, nx_local --> nlon
nlat, klat, ny, ny_local --> nlat
nlev, klev               --> nlev
ntyp,  ktyp     --> ntyp
nsvar, ksvar    --> nsvar
nhalo, khalo    --> nhalo
kacdg, nsacdg   --> nacdg
nhorph, khorph --> nhorph
itask, ktast    --> ntask
ndiffx, nrec_fft --> ndiffx
ndiffy, ntri_fft --> ndiffy
```

Many other ideas, but maybe we should do it *upstream* and **together** :)

Local work framework I

- Autoconf/automake setup allowing to setup and build consistently across platforms:
  - tar -zxvf hirlam-7.0-bench-hlprog-vx.y.tar.gz
  - cd hirlam-7.0-bench-hlprog-vx.y
  - F77=ftn ./configure --enable-mpi --enable-hgs && make
  - FFLAGS="-tp k8-64 -fastsse -Mipa=fast" F77=pgf90 ./configure && make
- Build and run wrapper scripts allowing to build, run and profile consistently across platforms:
  - ./build_run_ mpi, mpi_hgs, openmp, 64, ansi, ieee, stack, profile, valgrind,...

Local work framework II

- Poor-mans verification (manual inspections too complex)
  - Domain average values
  - Gridpoint values
  - YAA, probably less poor (please consult Henrik Feddersen)

Local work framework III

- Most of the ideas mentioned above has been implemented as a proof_of_concept on a forked version of hirlam 7.0
- Code: Standard Fortran90 compliant with consistent layout, implicit none, intent (incomplete), no compiler warnings
- Feature enhancments: 32/32, 64/32
- Bugs found (and most of fixed) (categories: interface, uninitialized, parallelized)
- Cleanup statistics *(do not take these numbers too serious)*

Current number of source code lines: 76522 (clean-score: 40.1 %)
Current number of subroutines and functions: 669 (clean-score: 34.7 %)
Current number of CPPFLAGS: 9 (clean-score: 85.9 %)
Current number of source files: 293 (clean-score: 31.2 %)
Current number of include files: 43 (clean-score: 50.0 %)
Current number of include lines:: 11116 (clean-score: 19.5 %)

The end

- Thanks to ...
  - The DMI project group (Maryanne Kmit, Henrik Feddersen)
  - The Hirlam system group (you all know the dream-team)
  - FMI (Niko Sokka), INM (Jose A. Garcia-Moya)
  - Roland Richter, SGI

Code conventions (proposal) I

File structure proposal

```
SUBROUTINE foo ()

   !<subroutine description>

   IMPLICIT NONE

   #<includes files>

   ! Input/output parameters

   <i/o variables>

   ! Local parameters

   <local varables>

   ! -------------------------

   <The code>

END SUBROUTINE foo
```

Code conventions (proposal) II

- FORTRAN keywords are in upper-case (e.g DO, IF, .TRUE., .AND.)
- everything else (but comments) are in lower-case
- comments uses upper-case and lower-case like in normal text
- comments start with "! " in the first two columns
- comments in the code have NO attached section numbers
- continuation lines of argument lists are indented so that they align behind the opening paranthesis
- continuation lines of equations or similar are indented so that they align one char behind the equality sign

Code conventions (proposal) III

- logical operators are surrounded by single spaces
- mathematical operators (+, -, * etc.) and the "=" are surrounded by single spaces
- equality signs are aligned blockwise
- continuation lines are marked by a "."
- blocks in conditions or loops are indented by 2 spaces
- there is never a space directly behind an opening paranthesis and directly before a closing paranthesis

Code conventions (proposal) IV

- there is one single space before the opening paranthesis in a subroutine call or in IF statements, or in similar FORTRAN constructs
- there is one single space behind the closing paranthesis in an if block and similar FORTRAN constructs
- there is no space before the opening paranthesis of arrays and similar constructs
- the argument list in subroutine calls is separated by comma plus a single space
- the dimension list of an array is separated by comma only, without any space in between