Hacking Gromacs: Getting Your Feet Wet With CVS Code

Erik Lindahl

lindahl@cbr.su.se
Outline: Hacking Gromacs

- Release vs. development codebase
- Building the CVS version(s) of Gromacs
- Configuration with automake & autoconf
- Architecture
- Organization of the source code
- Examples of modifications
- Good bug reports & patches
- Creating “distribution tarballs”
CVS repository

- Tracks all changes in Gromacs files
- Real-time update (whenever we check in)
- “How did stat.c change from 3.1.3 to 3.1.4?”
- Comments to all changes
- Immediate access to bug fixes
- Access to latest development version
- Test sets, manual source, etc.
- Completely public read-only access!
CVS branches

Always two ‘active’ branches: HEAD & patches

3.2
release-3-2-patches
3.2.1

3.3
release-3-3-patches
3.3.1

HEAD

3.3.99_development_20070215

3.3.2
Using CVS

- You probably have it installed ('which cvs')
- Instructions available on Gromacs site
- First log in (anonymously)

```bash
$>cvs -z3 -d :pserver:anoncvs@cvs.gromacs.org:/home/gmx/cvs login
Password: [leave blank, hit return]
```

- “Check out” latest HEAD version:

```bash
$>cvs -z3 -d :pserver:anoncvs@cvs.gromacs.org:/home/gmx/cvs co gmx
```

- Check out the release branch instead: (1 line)

```bash
$>cvs -z3 -d :pserver:anoncvs@cvs.gromacs.org:/home/gmx/cvs co -r release-3-3-patches gmx
```
Working with CVS

- The CVS ‘co’ command checks out a Gromacs version in a gmx subdirectory
- You cannot built yet - no configure script!
- Create it with command “./bootstrap”
- What is the configure script, really?
Configuration

- “Where is the X11 library?”
- “What version is the FFTW library?”
- “Is the Intel Math Kernel Library installed?”
- “Do we have that buggy gcc version?”
- “Does the compiler understand assembly?”
- “Which flags should be used for this compiler?”
- “Is this a big or small endian system?”
- “Is a long integer 4 or 8 bytes?”
- “How do we build a shared library here?”
GNU autotools

- Automatic system configuration
- "Ugly tools for an ugly world"
- Avoid editing complicated makefiles
# Note: Makefile is automatically generated from Makefile.in by the configure script, and Makefile.in is generated from Makefile.am by automake.

```
AM_CPPFLAGS = -I$(top_srcdir)/include -DGMXLIBDIR="$(datadir)/top"
LDADD = ../mdlib/libmd@LIBSUFFIX@.la ../gmxlib/libgmx@LIBSUFFIX@.la
```

```
bin_PROGRAMS = \
  grompp          mdrun           tpbconv         pdb2gmx          \
  protonate       luck            gmxdump          \
  gmxcheck        x2top           ffscan

mdrun_SOURCES = \
  glaasje.c       glaasje.h       gctio.c         init_sh.c       \
  ionize.c        ionize.h        xmdrun.h        \
  do_gct.c        relax_sh.c      repl_ex.c       repl_ex.h       \
  xutils.c        compute_io.h    compute_io.c  \
  md.c            mdrun.c         genalg.c        genalg.h
```

Note: No system-specific stuff or conditional libraries!
(We never look in the autogenerated complex Makefile.in)
Autoconf

- How do we do the system configuration?
  - Cannot count on anything being present
  - Use an extremely generic shell script!
  - This script (configure) is generated automatically for us by autoconf, from an input specification in the file configure.ac (written in the M4 macrolanguage)
configure.ac example

# Process this file with autoconf to produce a configure script.

AC_PREREQ(2.50)
AC_INIT(GROMACS, 3.3.1, gmx-users@gromacs.org)
AC_CONFIG_SRCDIR(src/gmxlib/3dview.c)
AC_CONFIG_AUX_DIR(config)
AC_CANONICAL_HOST

### Single/Double
AC_ARG_ENABLE(float,
[  --disable-float               use double instead of single precision],
[  enable_float=yes]
)

AC_CHECK_SIZEOF(int)
AC_CHECK_SIZEOF(long int)

#####
# Checks for additional and/or optional functions or libraries.

AC_FUNC_MALLOC
AC_FUNC_MEMCMP
AC_TYPE_SIGNAL

Many pre-existing macros (our own are inacinclude.m4)
autoconf uses libtool to create shared libraries portably
Building Gromacs

- After running `./bootstrap`, you have the same setup as a Gromacs distribution

- Building Gromacs:
  - `./configure --prefix=/some/install/path`
    (errors? read the end of config.log)
  - `make -j #` (#=number of CPUs you have)
  - `make install`
Source tree organization

gmx
  admin
  config
  scripts
  man
  share
    html
    tutor
    template
    top
  include
    types
  src
    gmxlib
    mdlib
    kernel
    tools
    ngmx
    contrib
    nonbonded

low-level routines

assembly kernels

data structure definitions

high-level routines

core programs
Gromacs flowcharts

- High level path through the source code during a simulation
- Calculating forces
- Data structures, neighbor lists
- Original charts by Gerrit Groenhof
mdrun -v -s topol.tpr

**main()**
- kernel/mdrun.c

**mdrunner()**
- kernel/md.c

**do_md()**
- kernel/md.c

**Normal modes**

**do_nm()**
- mdlib/minimize.c

**do_lbfgs()**
- mdlib/minimize.c

**do_cg()**
- mdlib/minimize.c

**do_steep()**
- mdlib/minimize.c

**Minimization**

**do_force()**
- mdlib/sim_util.c

**write_traj()**
- mdlib/stat.c

**update()**
- mdlib/update.c

**Details on next slide**

**do_update_md()**
- mdlib/update.c

**constrain()**
- mdlib/constr.c

**print statistics and quit**
- kernel/md.c
do_force() in mdlib/sim_util.c

ns() in mdlib/force.c

force() in mdlib/force.c

do_nonbonded() in gmxlib/nonbonded/nonbonded.c

gmx_pme_do() in mdlib/pme.c

calc_bonds() in gmxlib/bondfree.c

search_neighbors() in mdlib/ns.c

nb_kernel312() in ../nonbonded/nb_kernel/nb_kernel312.c

nb_kernel312_x86_64_sse() in ../nb_kernel312_x86_64_sse.s

gmx_nb_free_energy_kernel() in gmxlib/nonbonded/nb_free_energy.c

setup_kernels() called on first execution of do_nonbonded()

spread_q_bsplines() in mdlib/pme.c

solve_pme() in mdlib/pme.c

+3D FFT

+3D iFFT

gather_f_bsplines() in mdlib/pme.c

bonds(), angles(), pdihs(), etc in gmxlib/bondfree.c

do_nonbonded14() in gmxlib/nonbonded/nonbonded.c
nb_kernel100_c.c:

```
for (k=nj0; (k<nj1); k++)
{
    jnr           = jjnr[k];
    j3            = 3*jnr;
    jx1           = pos[j3+0];
    jy1           = pos[j3+1];
    jz1           = pos[j3+2];
    dx11          = ix1 - jx1;
    dy11          = iy1 - jy1;
    dz11          = iz1 - jz1;
    rsq11         = dx11*dx11+dy11*dy11+dz11*dz11;
    rinv11        = 1.0/sqrt(rsq11);
    qq            = iq*charge[jnr];
    rinvvsq       = rinv11*rinv11;
    vcoul         = qq*rinv11;
    vctot         = vctot+vcoul;
    fscal         = (vcoul)*rinvvsq;
    tx            = fscal*dx11;
    ty            = fscal*dy11;
    tz            = fscal*dz11;
    fix1          = fix1 + tx;
    fiy1          = fiy1 + ty;
    fiz1          = fiz1 + tz;
    faction[j3+0] = faction[j3+0] - tx;
    faction[j3+1] = faction[j3+1] - ty;
    faction[j3+2] = faction[j3+2] - tz;
}
```

nb_kernel100_x86_64_sse.s:

```
;# calculate rinv=1/sqrt(rsq)
rsqrtps xmm5, xmm1
mulps xmm2, xmm5
mulps xmm5, xmm5
unpcklps xmm0, xmm7 ;# jqa jqb jqc jqd
mulps xmm4, [rsp + nb100_three]
mulps xmm5, xmm1 ;# rsq*lu*lu
subps xmm4, xmm5 ;# 30-30*lu*lu
mulps xmm4, xmm2
mulps xmm0, [rsp + nb100_iq]
mulps xmm4, [rsp + nb100_half]
movaps xmm1, xmm4
mulps xmm4, xmm4
;# xmm1=rinv
;# xmm4=rinvvsq

;# calculate coulomb interaction, xmm0=qq
mulps xmm0, xmm1 ;# xmm0=vcoul
mulps xmm4, xmm0 ;# xmm4=fscal

;# add potential to vctot (sum in xmm12)
addps xmm12, xmm0

mov rsi, [rbp + nb100_faction]
;# the fj's - accumulate x & y forces from memory
movlps xmm0, [rsi + r8*4] ;# x1 y1 -
movlps xmm1, [rsi + r10*4] ;# x3 y3 -
movhps xmm0, [rsi + r9*4] ;# x1 y1 x2 y2
movhps xmm1, [rsi + r11*4] ;# x3 y3 x4 y4

;# calculate scalar force by multiplying dx with fscal
mulps xmm9, xmm4
mulps xmm10, xmm4
mulps xmm11, xmm4

;# xmm0-xmm2 contains tx-tz (partial force)
;# accumulate i forces
addps xmm13, xmm9
addps xmm14, xmm10
addps xmm15, xmm11
```
Common data structures

**topology**
- name
- guess!
- idef
- bonded interactions
- atoms
- mass, charge, etc.
- block[]
- block definitions
- symtab
- name references

**inputrec**
- all the mdp options
- nsteps
- ns_type
- nstlist
- nstxout
- pme_order
- ... userint1-4
- userreal1-4

**forcerec**
- derived options
- rlist
- neighborlist cutoff
- epsilon_r
- reeltypes
- cutoff_RF/PME
- vdwtype
- cutoff/bham/tables
- nnblists
- nblists[]
- neighbors
- nbfp
- nonbonded params

**mdatoms**
- massA/B
- particle mass
- chargeA/B
- particle charge
- typeA/B
- particle LJ type
- cTC[]
- T-coupling groups
- cENER[]
- Energy groups
- cFREEZE[]
- Freeze groups

**block**
- nr
- #blocks
- index[]
- array with indices in a[]
- nra
- number of atoms
- a[]
- array with atom numbers in each group
Gromacs block definition

index[0,..,nr-1]

index[0,..,nra-1]
Gromacs neighborlists

- **il_code**: index to nb kernel
- **icoul / ivdw**: Coul & vdw type
- **nri**: number of lists
- **nrj**: # neighbors
- **iinr**: index of list owners
- **jindex**: list limits
- **jjnr**: neighbor indices

```
ii[nr][0,...,nri-1]
jin[nr][0,...,nri]
njn[nr][0,...,nri-1]
```

**topology**
Bugzilla

- [http://bugzilla.gromacs.org](http://bugzilla.gromacs.org)
- Report, track, and find patches for bugs
- NOT for support, though - we often don’t comment at all on a bugzilla entry until we have had time to test/confirm it
- Mailing list is better if you are not sure
- Good bug reports have lots of information
- Always try to provide a (small) test case
Creating a distribution

- Autoconf comes with built-in capabilities to create source ‘tarballs’ (like ours) that include your modifications:

  $> \text{make dist}$

  \textit{Just creates gromacs-<RELEASE>.tar.gz}

  $> \text{make distcheck}$

  1. Create gromacs-<RELEASE>.tar.gz
  2. Check that all files are there
  3. Try to build it in a temporary directory
  4. Make sure that it cleans up fine
Creating a patch

- Once you have created a new feature or fixed a bug, how do you contribute it back?
- In the top (gmx) directory, issue:
  ```
cvs diff > fix.patch ( better: cvs diff -U 3 )
  ```
- Try to clean the resulting fix.patch so it only contains the important changes!
- To apply it to another source tree:
  ```
patch < fix.patch
  ```