GROMACS - Bug #2954

genion changes residue numbering

05/14/2019 03:48 PM - Eiso AB

Status:	New		
Priority:	Normal		
Assignee:			
Category:			
Target version:			
Affected version - extra info:		**Difficulty:**	uncategorized
Affected version:	2019-beta1		

Description

Using the following commands to add ions:

```
gmx grompp -f ions.mdp -c solv.gro -p topol.top -o ions.tpr
gmx genion -s ions.tpr -o solv_ions.gro -p topol.top -pname NA -nname CL -neutral
```

My system is the following with protein residue nrs 79-405 and lig 411

```
[ molecules ]
; Compound        #mols
Protein_chain_A     1
lig                 1
SOL             35973
NA                  4  ; after genion
```

In solv.gro the residue numbering is still 79-405 (protein),411 (lig)

But in solv_ions.gro the residue numbering changed to 79-405 (protein),406 (lig) so gmx genion doesn't conserve the residue numbering in the .gro file

probably happens when there are gaps in the sequence numbering.

-Eiso

History

#1 - 05/15/2019 03:50 PM - Paul Bauer

Is this behaviour different to previous versions of GROMACS? I would see this as expected behaviour, but maybe it should be mentioned in the tool help text.

#2 - 05/15/2019 04:38 PM - Eiso AB

Not sure if it is different for this version, but I dont' think it is (or should be) expected behaviour.
There are convenience reasons for having a certain user defined residue numbering (e.g. easy comparison with sequence aligned homologues, with multimers it's easiest when all monomers start at n*100 + i)

Since the residue nr is just a label IMHO it would be best if by default the user supplied residue numbering is conserved as much as possible. It's annoying if you need to renumber the pdb files again after the md run because you want to compare it to something with a different numbering - much easier if it just keeps it as it is.

Long time ago gmx required that the system was always renumbered from 1 to n 9 (or maybe 0 ..n)
At some point this was changed so now it keeps for the first molecule at least the numbering as it is supplied (79-405), but apparently for the second molecule it doesn't (411->406). It see no obvious reason why it would keep the numbering of one molecule and change it for another - consistent behaviour would be to keep al resnrs as they are.

That said, it's probably easily circumvented by modifying the residue nrs in the final topology file working with that.
(not actually tested).

#3 - 05/15/2019 05:06 PM - Paul Bauer

I see your point and will investigate this.
I quick read through genion did not show anything that would do this, the only place I know of where residues are explicitly renumbered is in genconf.

#4 - 05/15/2019 05:15 PM - Paul Bauer

can you provide your inputs and command line so I can check this in the debugger? :)

#5 - 05/16/2019 01:31 AM - Eiso AB

- File newbox.gro added

- File ions.mdp added

- File lig_GMX_1.itp added

- File lig_GMX_2.itp added

- File topol-orig.top added

- File source-this-rc added

I've attached files that should allow you to reproduce the problem.
Just put all in one dir and do:

```
source source-this-rc
```

upon further inspection, it's a bit more complicated than I imagined.

first I have to take this back

> That said, it's probably easily circumvented by modifying the residue
> nrs in the final topology file working with that.
> (not actually tested).

unlike what I expected the top/tpr files doesn't seem to explicitly contain residue nrs for all of the system,
so it doesn't have the information to reproduce the residue numbering in the .gro file that was used to run grompp and make the tpr file.

gmx mdrun will also have the same renumbering behaviour since it also only has a tpr to work from.

```
gmx dump -s ions.tpr |grep resid
         residue (208):
            residue[0]={name="VAL", nr=17, ic=' '}
            residue[1]={name="GLU", nr=18, ic=' '}
           [..snip..]
           residue[207]={name="LYS", nr=224, ic=' '}
         residue (1):
            residue[0]={name="LIG", nr=300, ic=' '}
         residue (1):
            residue[0]={name="SOL", nr=1, ic=' '}
```

Here the residue nr for the protein 18-224 come from the topol.top,
the LIG residue nr 300 comes from lig_GMX_2.itp (#included in topol.top)
and the SOL resnr=1 from the amber99sb-ildn.ff/spce.itp

Since the tpr doesn't contain info about the input .gro file, for the SOL residues it obviously has to make up residue numbers for the multiple SOL residues, and simplest thing is just renumber from the last avaiable residue.
Apparently mdrun/genion gets the residue number for the first molecule from the topology and numbers on from there.
Not sure what happens with topologies with multiple protein molecules with (non)overlapping residue ranges..

I guess in theory it could use the the residue nr from the itp file (nr=300) if it is higher than the last residue but I have to admit that is a bit ugly way to do it.

I still think that conserving input residuenrs would be desirable from a user perspective but I can see it's not an easy fix

I can see a few options to make it easier to get back coordinate files with a certain residue numbering:

1. use the resnr in the itp if higher than previous (ugly)
2. include explicit residue info into the tpr file
3. add an option to genion/mdrun etc (or trjconv!) to accept a *reference* coordinate file from which it can take correct residue numbering assuming identical atom ordering. That would make it easy to extract pdb file with some arbitrary desirable numbering scheme after an mdrun

The 2nd option (explict resnrs in the tpr(top?) would allow general conservation of resnrs by genion/mdrun and I guess would be the least surprising behaviour for users (at least this one;-)

The 3rd option, I just realized ;-) , is already possible at least for trjconv if you give '-s' an appropriately numbered pdb or gro file !!
turns out I was doing this without realizing it ;-) https://en.wikipedia.org/wiki/Rubber_duck_debugging

Hmm, I guess this makes implementation of the second option less likely ;-(

E

#6 - 05/22/2019 11:54 AM - Paul Bauer

So, I checked the code and the issue is in the generation of the global atom information datastructure during preprocessing (in mtop_util.cpp atomcat for those interested).
All of this code is marked for major rework anyway, so I'm not sure if it is worth the effort to change the behavior now.

#7 - 05/26/2019 12:08 AM - Eiso AB

waiting for the larger rework sounds fine to me.

Files

newbox.gro	145 KB	05/15/2019	Eiso AB
ions.mdp	896 Bytes	05/15/2019	Eiso AB
lig_GMX_1.itp	799 Bytes	05/15/2019	Eiso AB
lig_GMX_2.itp	12.5 KB	05/15/2019	Eiso AB
topol-orig.top	902 KB	05/15/2019	Eiso AB
source-this-rc	300 Bytes	05/15/2019	Eiso AB