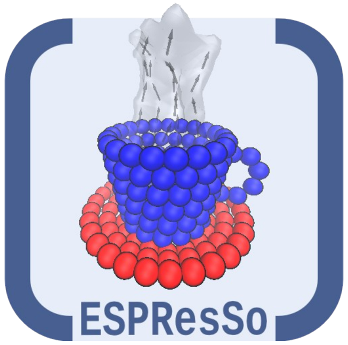
# Introduction to ESPResSo and Tcl



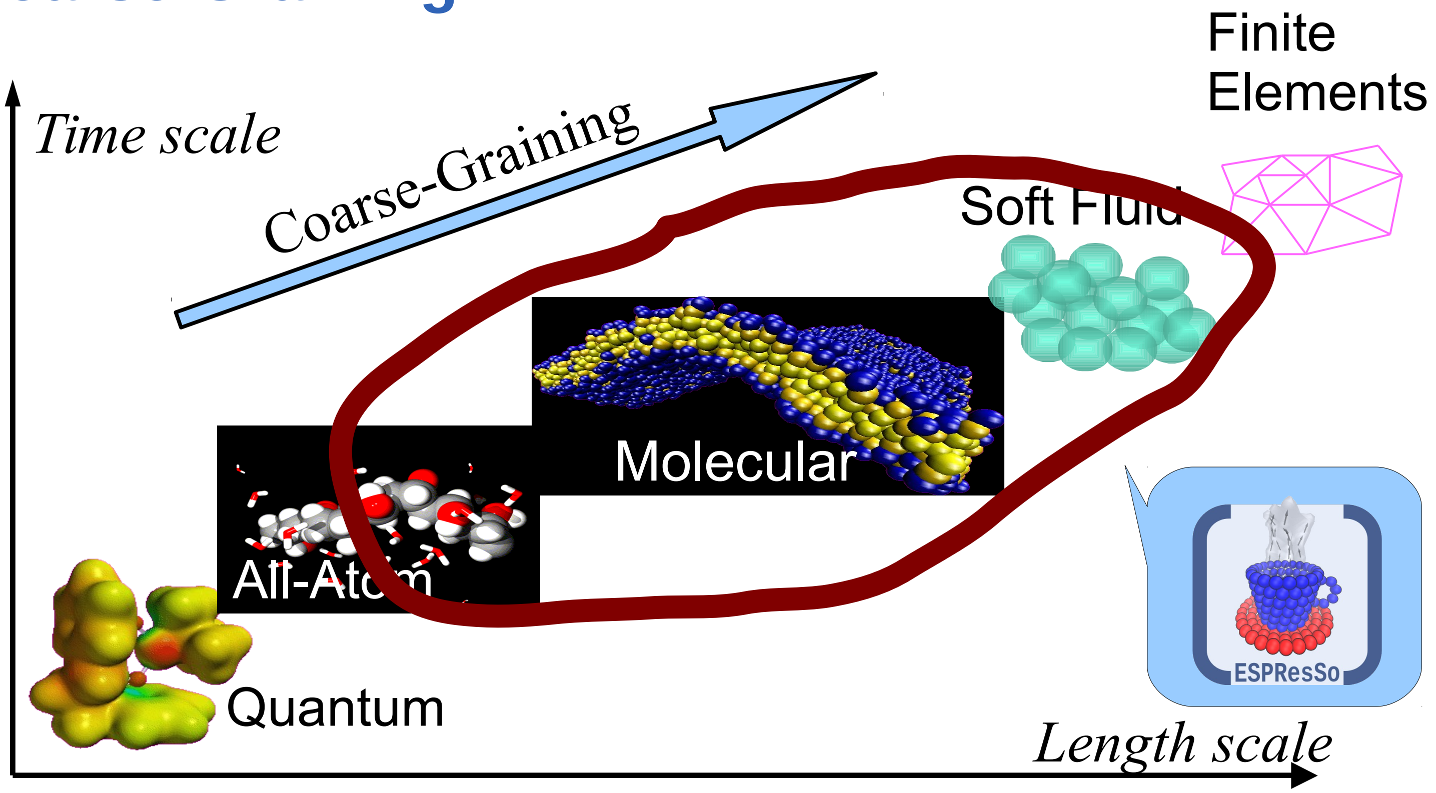
**Olaf Lenz**

Institut für Computerphysik, Universität Stuttgart
Stuttgart, Germany

University of Stuttgart
Germany

INSTITUTE FOR COMPUTATIONAL PHYSICS

## Coarse-Graining



- All-Atom Models
  - Model all atoms and their interactions with semi-quantitative parameters
  - Only small systems and short times can be simulated
- Coarse-Grained Models
  - Only model “important” degrees of freedom
  - Allows for much larger time and length scales

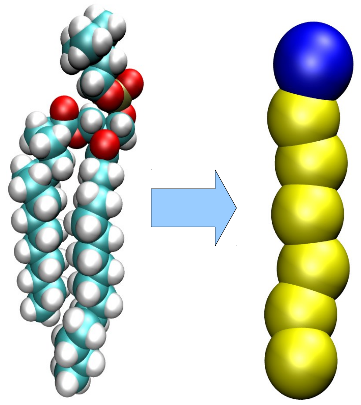
## Why yet another Simulation Package?

- Coarse-grained Bead-spring models:
  Combine several atoms into a single bead
- Often combined with other methods
  - Special interactions (DPD, Gay-Berne ellipsoids, ...)
  - Special integrators (MCPD, Hybrid MC/MD, ...)
  - Combined with lattice models (Lattice-Boltzmann, MEMD, ...)
  - Uncommon simulation protocols (Simulated annealing, Parallel tempering, …)
  - Special constraints (Walls, Pores, …)
- Standard MD simulation packages (GROMACS, NAMD, AMBER, …) are not flexible enough to deal with these models

⇨ **Package must be flexible!**

- In research, new methods are developed
- Building new methods into highly optimized code (GROMACS, NAMD, AMBER, …) is very hard

⇨ **Package must be extensible!**



## Philosophy

- ES is intended as a research tool and a production platform
- ES provides the methods for coarse-grained simulations
- However, an understanding of the methods is required to be able to use ESPResSo
- ES can not check whether what you do makes sense!

**Golden Rules**

1. ESPResSo can *not* be used as a black box
2. ESPResSo does *not* do the physics for you

## Methods

- **Integrators and ensembles**: Velocity-Verlet algorithm (NVE), Langevin thermostat (NVT), Barostat by Dünweg (NPT), Generalized Hybrid Monte-Carlo, Quarternion integrator for non-spherical particles or point-like dipoles, ...
- **Nonbonded interactions**: Lennard-Jones, Gay-Berne, Buckingham,, …
- **Bonded interactions**: harmonic, FENE, tabulated, bond-angle interaction, dihedral interaction, ...
- **Long-range interactions**: for electrostatics: $P^3M$, MMM1D, MMM2D, Ewald, ELC and MEMD; for point-like dipoles: dipolar $P^3M$, ScaFaCoS (FMM, ...), ...
- **Hydrodynamic interactions**: DPD, Lattice-Boltzmann fluid (on GPU) coupled to particle simulation
- **Constraints**: Particles can be fixed in any directions; walls, pores, spheres...
- **Analysis**: energy components, pressure tensor, forces, distribution functions, structure factors, polymer-specific analysis functions (radius of gyration, ...), output to VMD
- **...and it is continuously growing...**

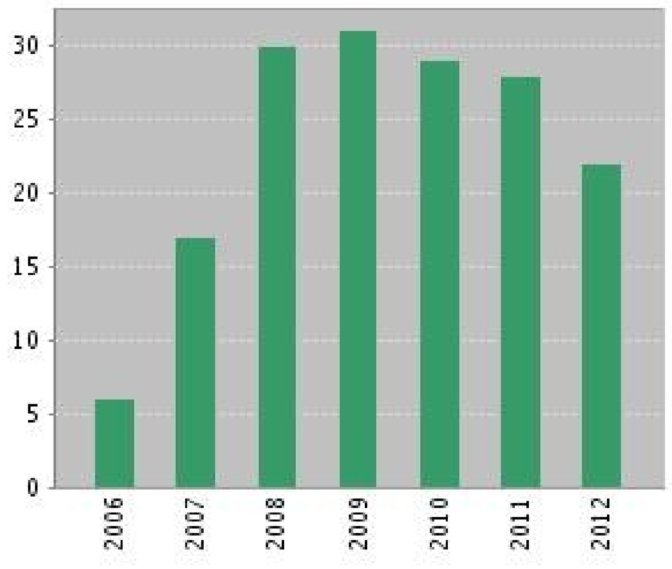
## Availability

- Free, open-source
- Source code hosted at GNU Savannah (not a GNU core project, though)
- GNU General Public License (GPLv3)
  - Code may be freely downloaded, modified and redistributed
  - Provided that the GPL is kept
- Portable: POSIX, Windows, Mac OS X
- Distribution packages exist for
  - Gentoo Linux (Christoph Junghans)
  - Fedora Linux (Thomas Spura; in progress)
  - … anybody interested in packaging for other distributions?

https://savannah.nongnu.org
/projects/espressomd/

## Distribution

- 163 Citations of the 2006 article (Web of Knowledge)
- Used by ~20 scientific working groups



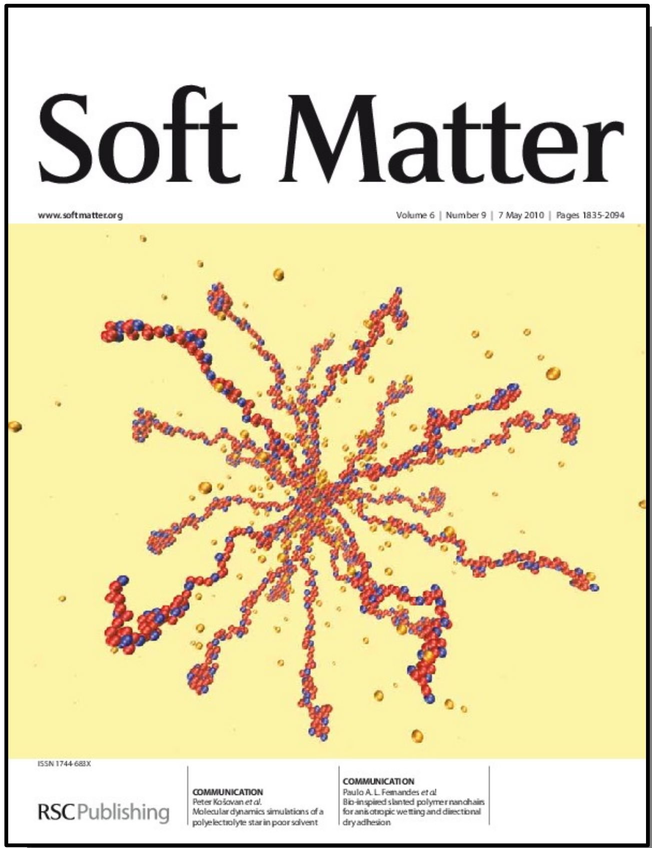
Citations per year

(from Web of Science, including self-citations)

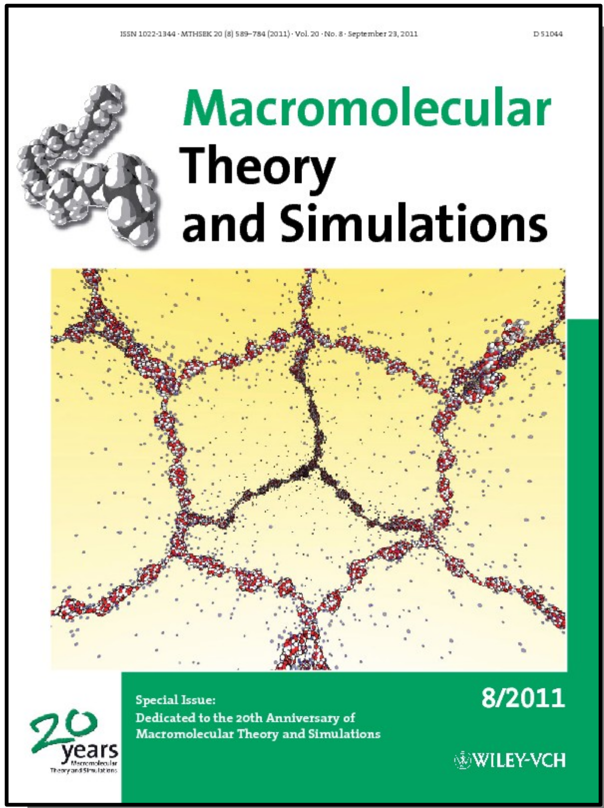
## Impact

IN THIS ISSUE NATURE INSIGHT: EPIGENETICS
24 May 2007 | www.nature.com/nature | £10
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
nature
FOLLOW THE CURVE
The forces that shape cell membranes
DOPAMINE JUBILEE
A molecule with a history
GLOBAL CLIMATE
The pull of the oceans
NEURODEGENERATIVE DISEASES
A clutch of new drug targets
NATUREJOBS
Spotlight on Seattle

2007


Soft Matter
www.softmatter.org
Volume 6 | Number 9 | 7 May 2010 | Pages 1835-2094
ISSN 1744-683X
RSCPublishing
COMMUNICATION
Peter Košovan et al.
Molecular dynamics simulations of a polyelectrolyte star in poor solvent
COMMUNICATION
Paulo A. L. Fernandes et al.
Bio-inspired slanted polymer nanohairs for anisotropic wetting and directional dry adhesion

2010

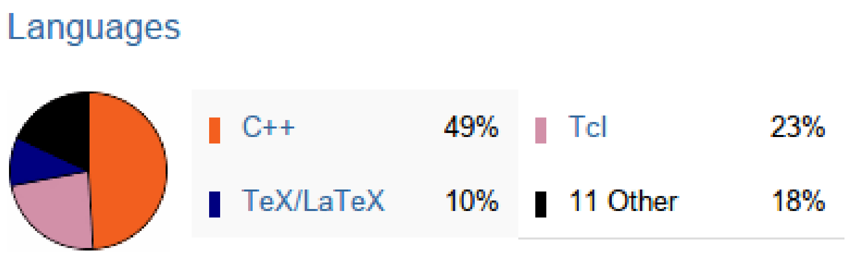

Macromolecular Theory and Simulations
Special Issue:
Dedicated to the 20th Anniversary of Macromolecular Theory and Simulations
8/2011
20 years
WILEY-VCH

2011

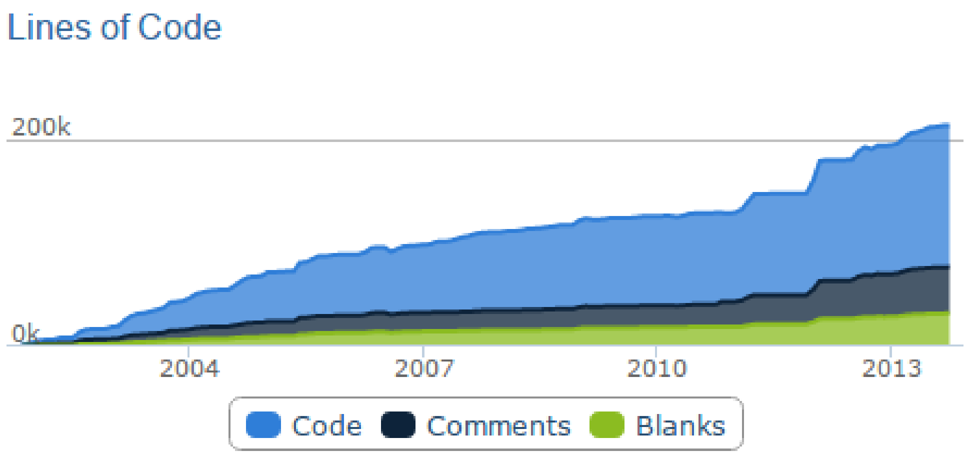
## Code

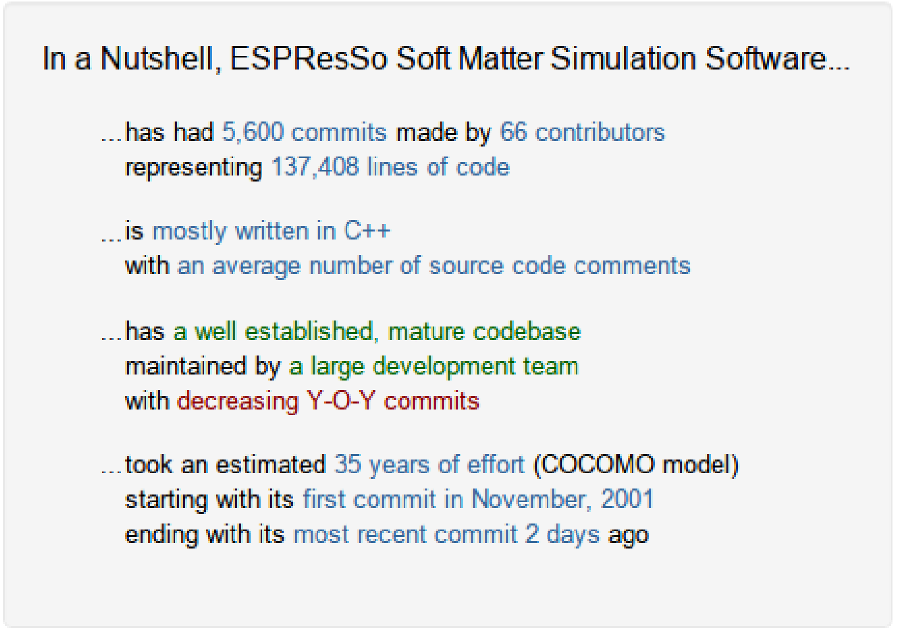
From Ohloh:
https://www.ohloh.net/p/ESPResSo_MD

- 5,600 commits from 66 contributors
- ~ 137,000 lines of code
- Estimate: ~34 person years, ~1.9M$ cost


Languages

| | | | |
|---|---|---|---|
| 🟧 C++ | 49% | ▮ Tcl | 23% |
| 🟦 TeX/LaTeX | 10% | ⬛ 11 Other | 18% |



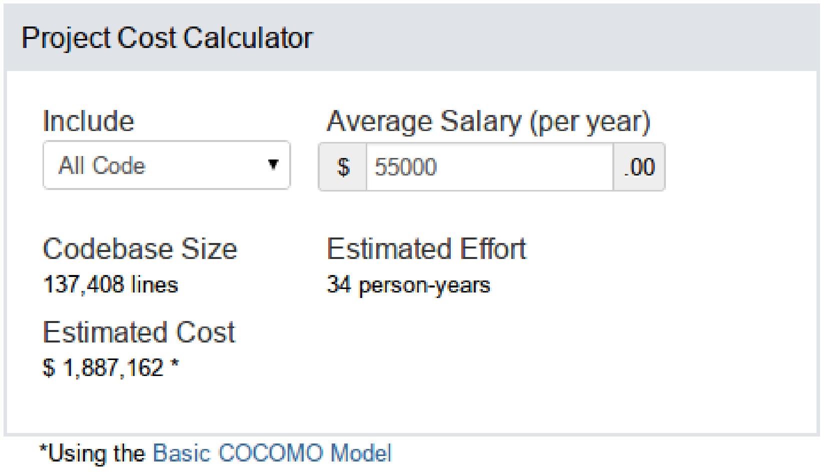

In a Nutshell, ESPResSo Soft Matter Simulation Software...

...has had 5,600 commits made by 66 contributors
representing 137,408 lines of code

...is mostly written in C++
with an average number of source code comments

...has a well established, mature codebase
maintained by a large development team
with decreasing Y-O-Y commits

...took an estimated 35 years of effort (COCOMO model)
starting with its first commit in November, 2001
ending with its most recent commit 2 days ago


Project Cost Calculator

Include: All Code
Average Salary (per year): $ 55000 .00

Codebase Size
137,408 lines

Estimated Effort
34 person-years

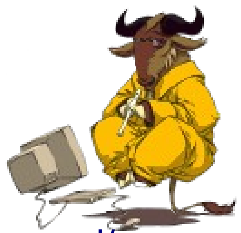
Estimated Cost
$ 1,887,162 *

*Using the Basic COCOMO Model

## Web Resources

Home page
http://espressomd.org

- Hosted at ICP
- Central resource for users
- Downloads
- Documentation → next slide
- Community and Support
  - (Link to) Bug tracker
  - (Link to) Mailing lists
  - (Link to) Wiki
- Developer's Zone (in the wiki)
  - Developer's docs
  - (Link to) Savannah project
  - (Link to) Source code repository
  - (Link to) Build server

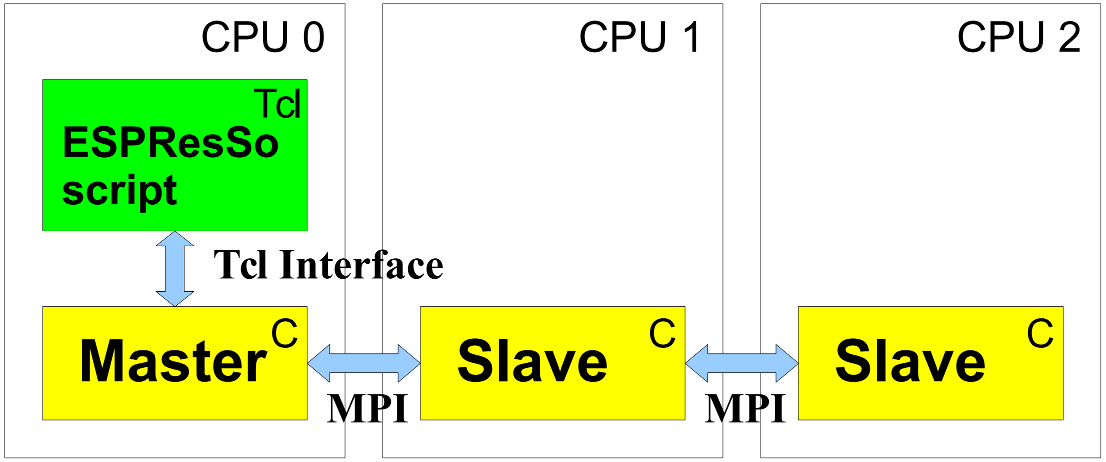


Savannah Project Page
https://savannah.nongnu.org/projects/espressomd/

- Hosted at GNU Savannah servers
- Download area
  - Release tarballs and NEWS
- Mailing list espresso-users@nongnu.org
  - Only mailings from members are accepted
- Bug tracker
  - Report bugs in releases!
- Mostly intended for Developers
  - Mailing list espresso-devel@nongnu.org
  - News
  - Source code repository
  - Task manager
  - Patch manager

## Architecture

- Simulation core
  - Written in C with some C++ enhancements
  - MPI parallelized
  - Optimized
- Control layer
  - Simulation core is controlled via the scripting language Tcl
  - High-level Tcl commands to control the simulation and analyze the system
  - A simulation is defined by an “ESPResSo script”
  - Tcl script is *not* executed in parallel!



```
.
.
setmd box_l 10.0 10.0 10.0
integrate 1000

# compute kinetic energy
set e_kin \
  [analyze energy kinetic]
.
.
```

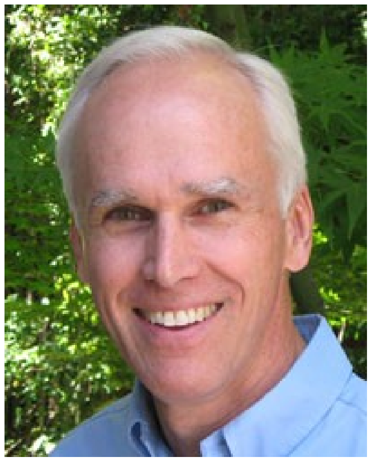
Example ESPResSo script

## Tcl

## History

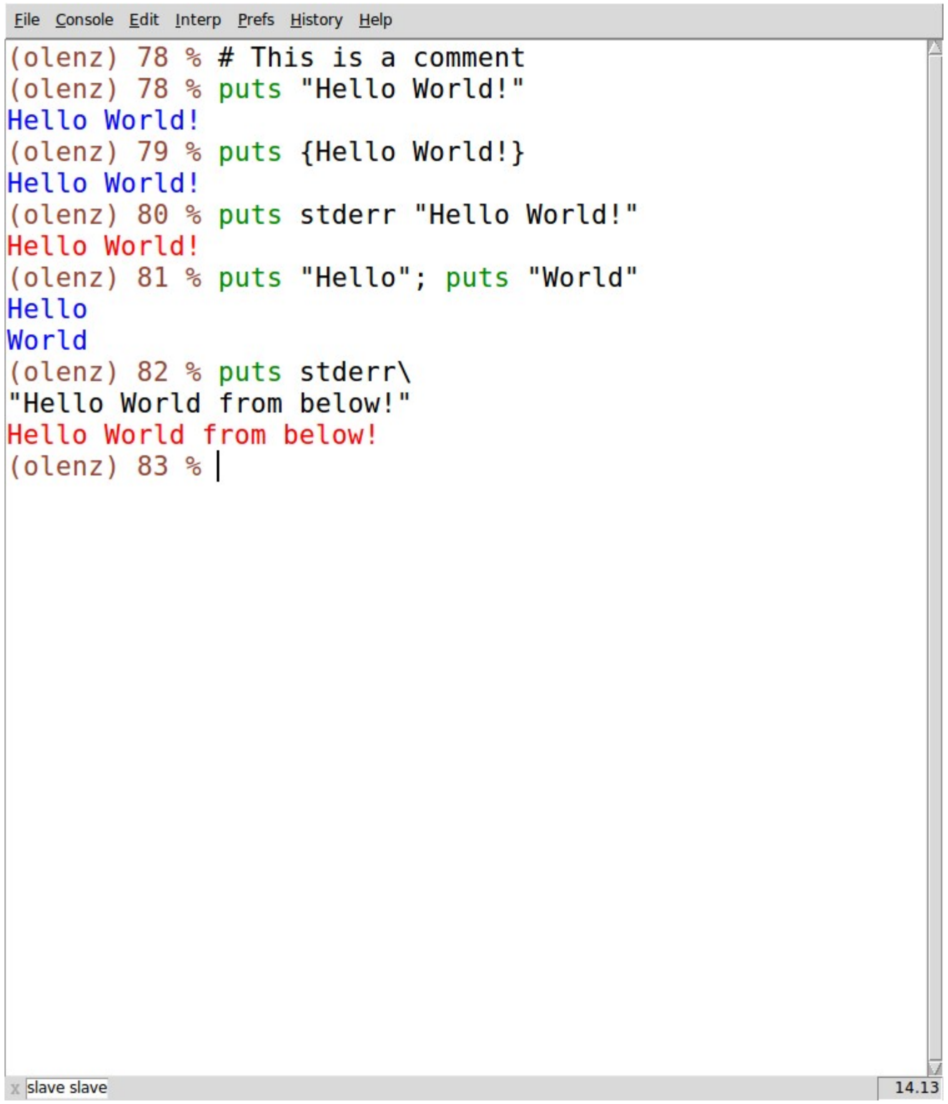
- “**T**ool **c**ommand **l**anguage”, pronounce “*tickle*”
- John Ousterhout, Berkley, 1988
- Originally invented for GUI programming (Tcl/Tk)
- Interpreted, procedural scripting language
- Motto: “Radically simple”
  - Simple syntax
    - All operations are commands
    - Including control structures (i.e. loops, conditionals)
  - No types: All data are strings
  - Dynamic: new procedures can be (re-)defined easily
- Simple C-API
- Free, open-source (BSD license)
- Current version 8.6.1 (20 Sep 2013)
- ...currently seem to regain some drive!
- Some programs use Tcl/Tk, e.g. VMD and NAMD
- … but most are slowly switching to Python...

http://www.tcl.tk


John Ousterhout

## Language Basics

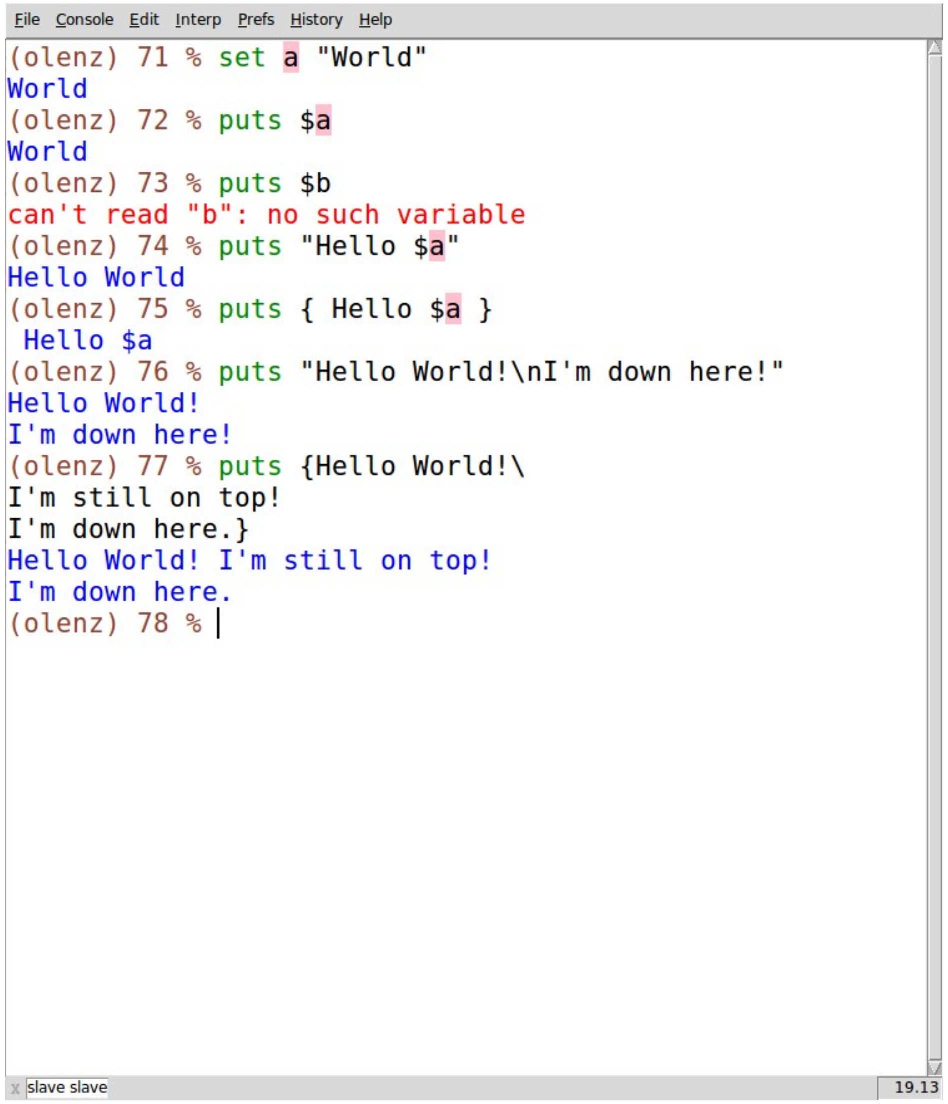
- Standard interpreter
  - tclsh, wish (with Tk)
  - Can be used interactively
- Improved console: tkcon
- Getting help on Tcl
  - Hompage http://www.tcl.tk
  - Unix manpages (e.g. `man n open`)
- Comment char: #
- Command to print to screen: puts
- General syntax
  - First word in a line is a *command*
  - Rest are *arguments*
  - Several commands in a line with ;
- "" or {} can be used to group arguments
- Continuation lines with “\” at the end of line



```
(olenz) 78 % # This is a comment
(olenz) 78 % puts "Hello World!"
Hello World!
(olenz) 79 % puts {Hello World!}
Hello World!
(olenz) 80 % puts stderr "Hello World!"
Hello World!
(olenz) 81 % puts "Hello"; puts "World"
Hello
World
(olenz) 82 % puts stderr\
"Hello World from below!"
Hello World from below!
(olenz) 83 %
```

## Variables, Substitution and Grouping

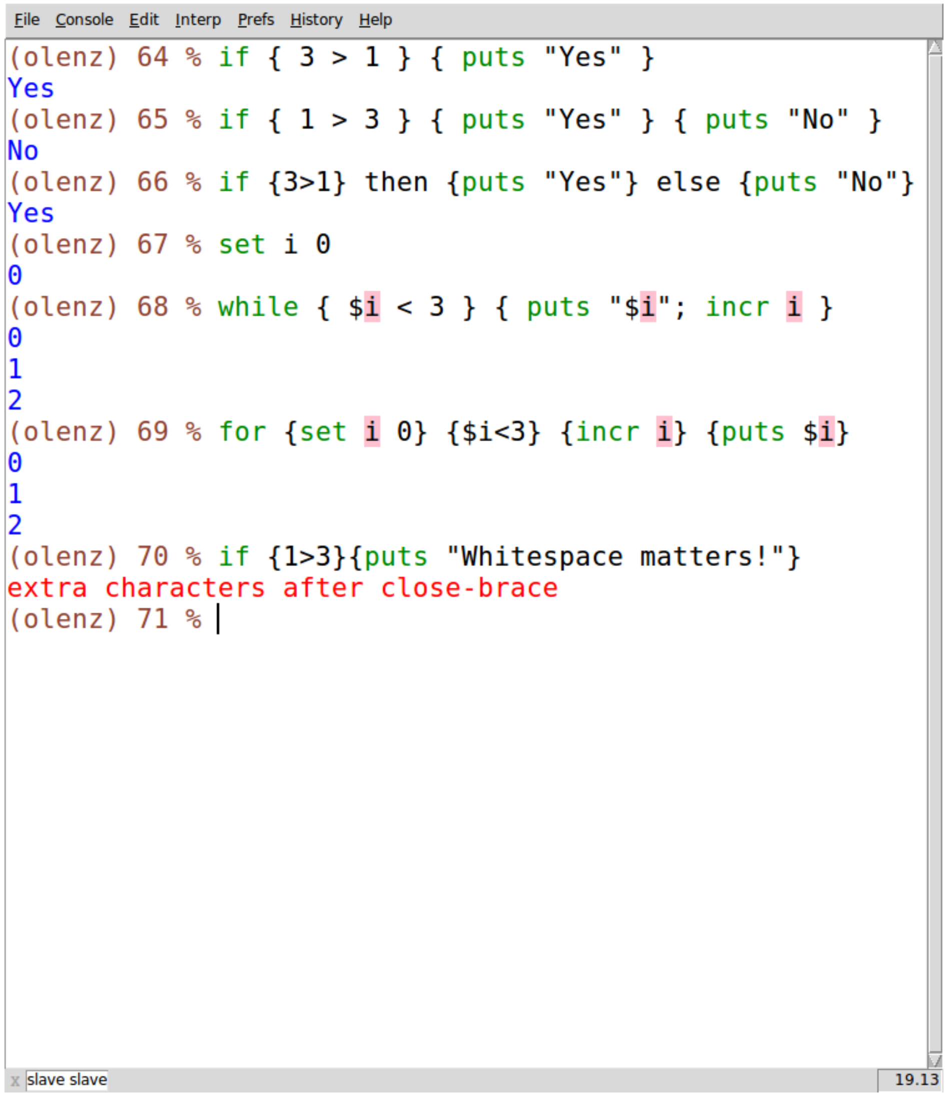
- Variables are set via
  `set` *`varName ?value?`*
  - What? No “a=13”?
  - Remember? Everything is a command!
- `$varName` is substituted for variable value
- Unknown variables are reported
- Argument grouping via `""`
  - Substitution works
  - Backslash-sequences work (\n, \t, …)
  - Use for strings
- Argument grouping via `{}`
  - No substitution
  - No backslash-sequences
  - Use for code blocks
  - Can stretch multiple lines!



```
(olenz) 71 % set a "World"
World
(olenz) 72 % puts $a
World
(olenz) 73 % puts $b
can't read "b": no such variable
(olenz) 74 % puts "Hello $a"
Hello World
(olenz) 75 % puts { Hello $a }
 Hello $a
(olenz) 76 % puts "Hello World!\nI'm down here!"
Hello World!
I'm down here!
(olenz) 77 % puts {Hello World!\
I'm still on top!
I'm down here.}
Hello World! I'm still on top!
I'm down here.
(olenz) 78 % |
```

## Control Structures

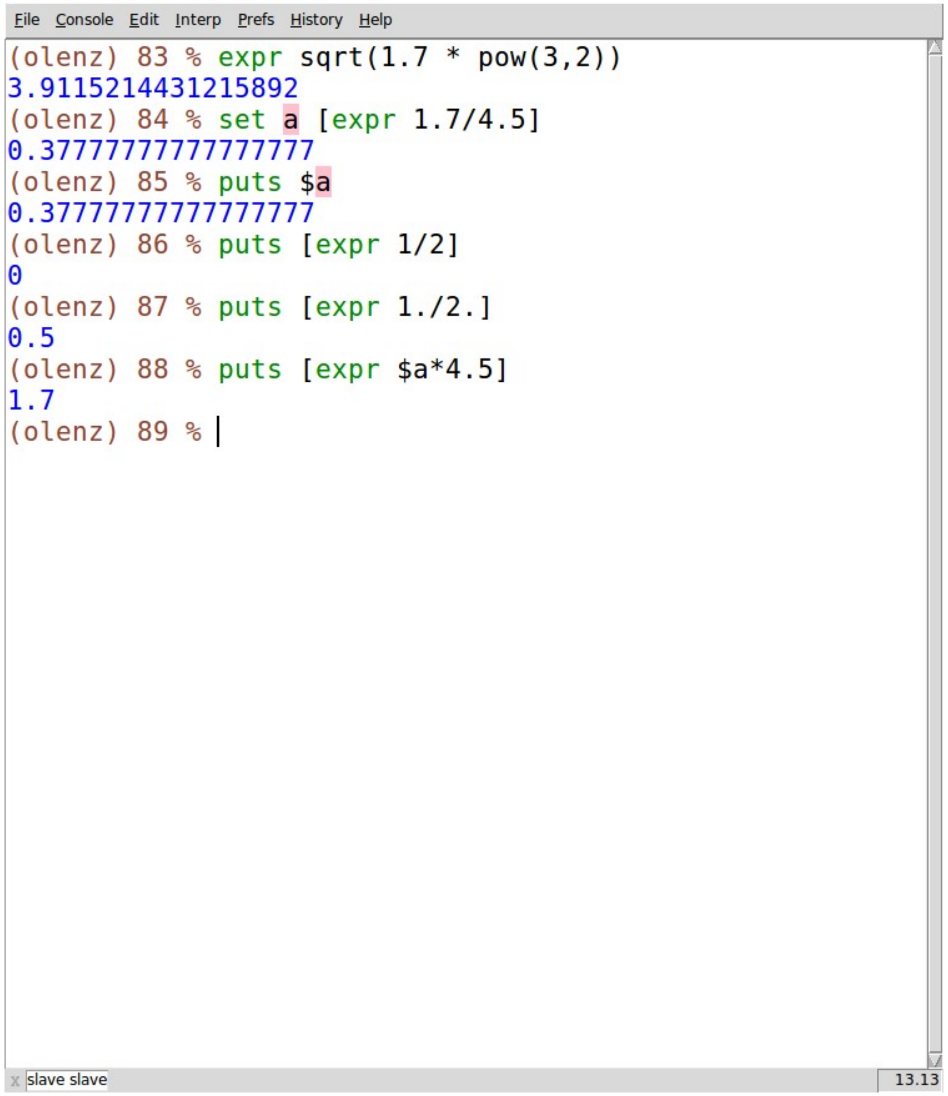
- Control structures are just commands!
- Conditional
  `if` *`expr1`* `?then?` *`body1`*
  `elseif` *`expr2`* `?then?` *`body2`*
  `elseif ...`
  `?else?` *`?bodyN?`*
  - Keywords `then` and `else` are optional
- Loops
  `while` *`test body`*
  `for` *`start test next body`*
  - `break` breaks a loop
- Mind the spaces between the arguments!



```
(olenz) 64 % if { 3 > 1 } { puts "Yes" }
Yes
(olenz) 65 % if { 1 > 3 } { puts "Yes" } { puts "No" }
No
(olenz) 66 % if {3>1} then {puts "Yes"} else {puts "No"}
Yes
(olenz) 67 % set i 0
0
(olenz) 68 % while { $i < 3 } { puts "$i"; incr i }
0
1
2
(olenz) 69 % for {set i 0} {$i<3} {incr i} {puts $i}
0
1
2
(olenz) 70 % if {1>3}{puts "Whitespace matters!"}
extra characters after close-brace
(olenz) 71 %
```

## Evaluating Expressions and Nested Commands

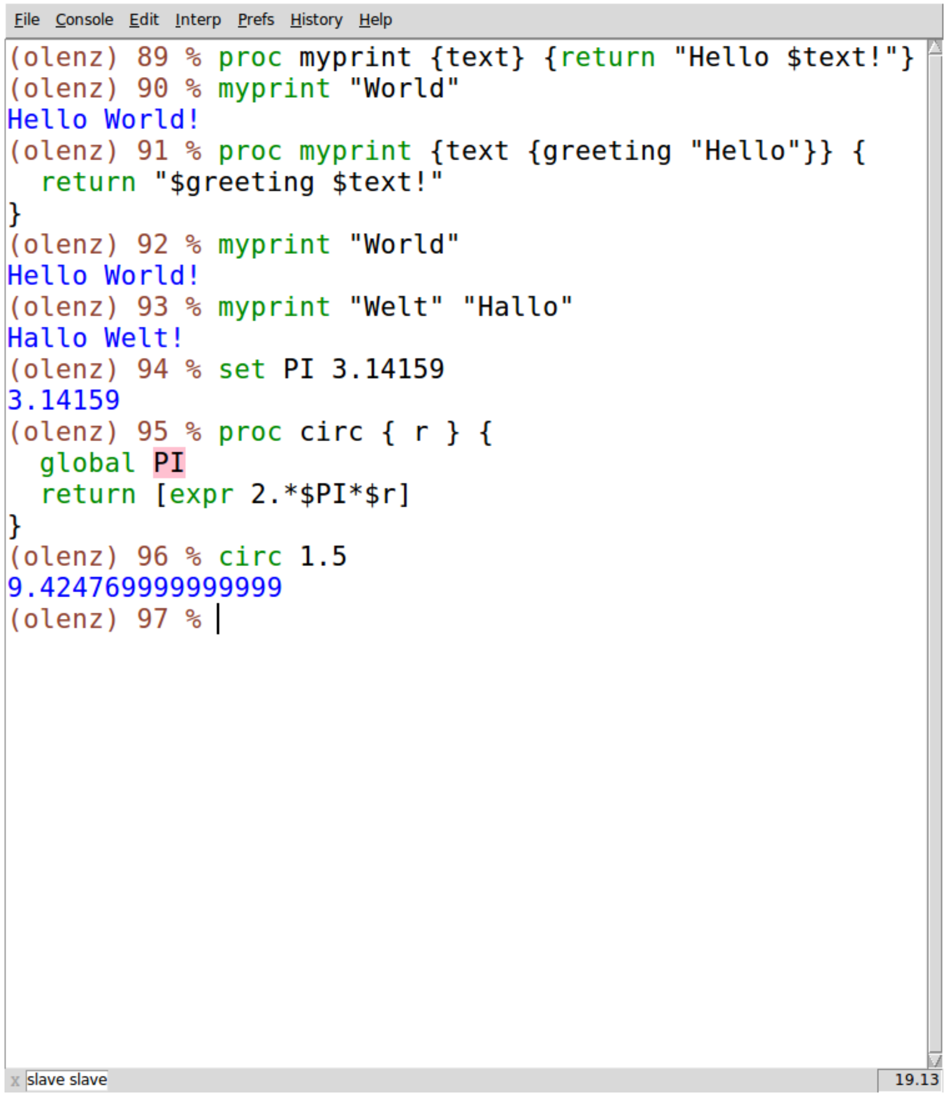
- Mathematical expressions can be computed using the command `expr` *?expression?*
  - *Expressions are mostly like C*
- Grouping via [ ]
  - Equivalent to shell backticks ` `
  - Executed as a nested command
  - Variable substitution works
  - Output is substituted
- "Everything is a string"
  - Numbers have to be transformed to and from a string
  - Slow numerics in Tcl!



```
(olenz) 83 % expr sqrt(1.7 * pow(3,2))
3.9115214431215892
(olenz) 84 % set a [expr 1.7/4.5]
0.37777777777777777
(olenz) 85 % puts $a
0.37777777777777777
(olenz) 86 % puts [expr 1/2]
0
(olenz) 87 % puts [expr 1./2.]
0.5
(olenz) 88 % puts [expr $a*4.5]
1.7
(olenz) 89 % |
```

## Adding New Commands

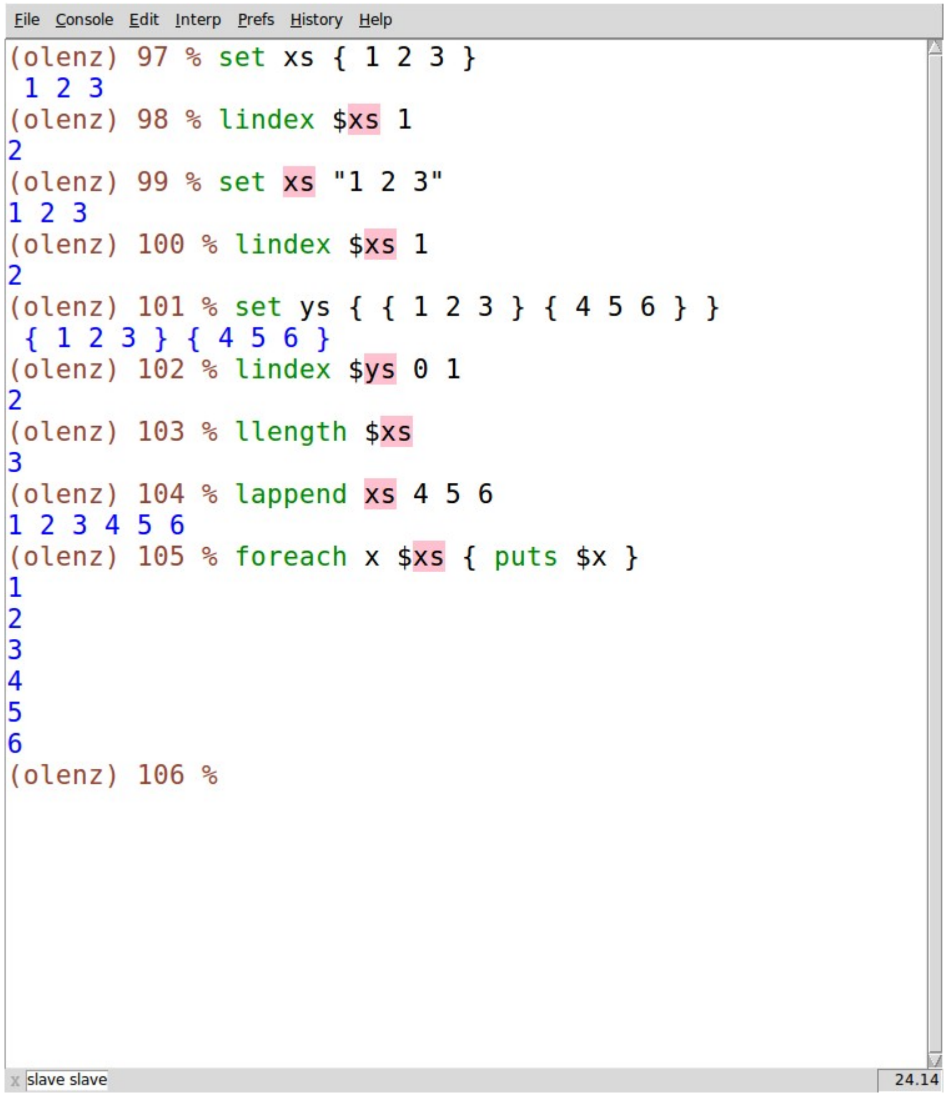
- Define a new command via `proc` *`name args body`*
  - Creates a new command with the given *name*
  - *`args`* defines the names of the arguments
  - In the body, local variables exist for each argument
  - The `return` command defines the return value of the command
  - It is possible to specify default arguments
- Variables are local if they are not declared `global`
- For further reference: `uplevel` and `upvar` to define control structures



```
(olenz) 89 % proc myprint {text} {return "Hello $text!"}
(olenz) 90 % myprint "World"
Hello World!
(olenz) 91 % proc myprint {text {greeting "Hello"}} {
  return "$greeting $text!"
}
(olenz) 92 % myprint "World"
Hello World!
(olenz) 93 % myprint "Welt" "Hallo"
Hallo Welt!
(olenz) 94 % set PI 3.14159
3.14159
(olenz) 95 % proc circ { r } {
  global PI
  return [expr 2.*$PI*$r]
}
(olenz) 96 % circ 1.5
9.424769999999999
(olenz) 97 %
```

## Lists

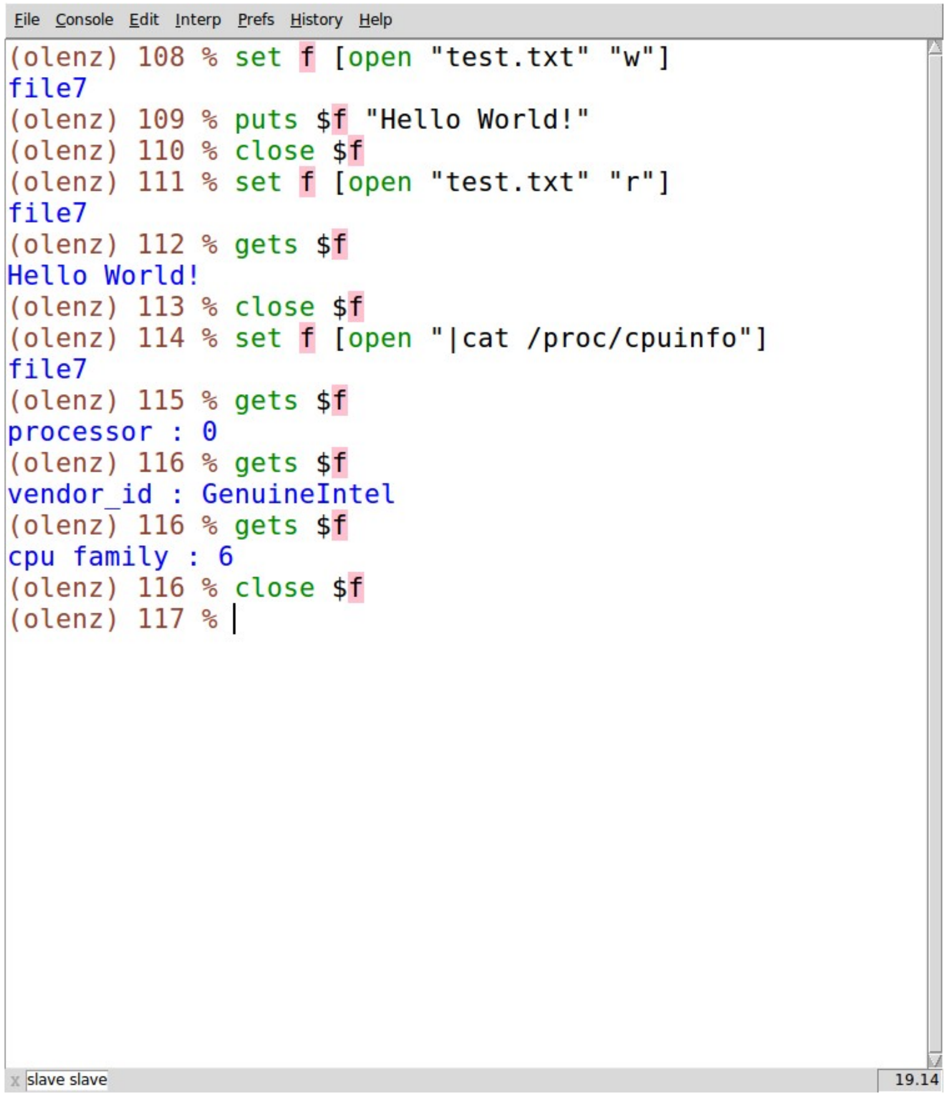
- `{ 1 2 3 }` is a list
- A list is a string!
- Nested lists
  `{ { 1 2 3 } { 4 5 6 } }`
- Commands
  - Access single elements with
    <u>`lindex`</u> *`list ?index...?`*
  - Get number of elements with
    <u>`llength`</u> *`list`*
  - Append elements to a list with
    <u>`lappend`</u> *`varname ?value...?`*
  - Loop over elements with
    <u>`foreach`</u> *`varname list body`*



```
(olenz) 97 % set xs { 1 2 3 }
 1 2 3
(olenz) 98 % lindex $xs 1
2
(olenz) 99 % set xs "1 2 3"
1 2 3
(olenz) 100 % lindex $xs 1
2
(olenz) 101 % set ys { { 1 2 3 } { 4 5 6 } }
 { 1 2 3 } { 4 5 6 }
(olenz) 102 % lindex $ys 0 1
2
(olenz) 103 % llength $xs
3
(olenz) 104 % lappend xs 4 5 6
1 2 3 4 5 6
(olenz) 105 % foreach x $xs { puts $x }
1
2
3
4
5
6
(olenz) 106 %
```

## File I/O

- Open a file with
  `open` *`filename access`*
  - Returns a *channelId*
  - *access* is a letter, e.g. “r” for reading, “w” for writing
  - If *filename* starts with “ | ”, open a pipe to a command
- Write to a channel with
  `puts` *`channelId`*
- Read from the channel with
  `gets` *`channelId`*
- Close file with
  `close` *`channelId`*



```
(olenz) 108 % set f [open "test.txt" "w"]
file7
(olenz) 109 % puts $f "Hello World!"
(olenz) 110 % close $f
(olenz) 111 % set f [open "test.txt" "r"]
file7
(olenz) 112 % gets $f
Hello World!
(olenz) 113 % close $f
(olenz) 114 % set f [open "|cat /proc/cpuinfo"]
file7
(olenz) 115 % gets $f
processor : 0
(olenz) 116 % gets $f
vendor_id : GenuineIntel
(olenz) 116 % gets $f
cpu family : 6
(olenz) 116 % close $f
(olenz) 117 % |
```

## … and there is more

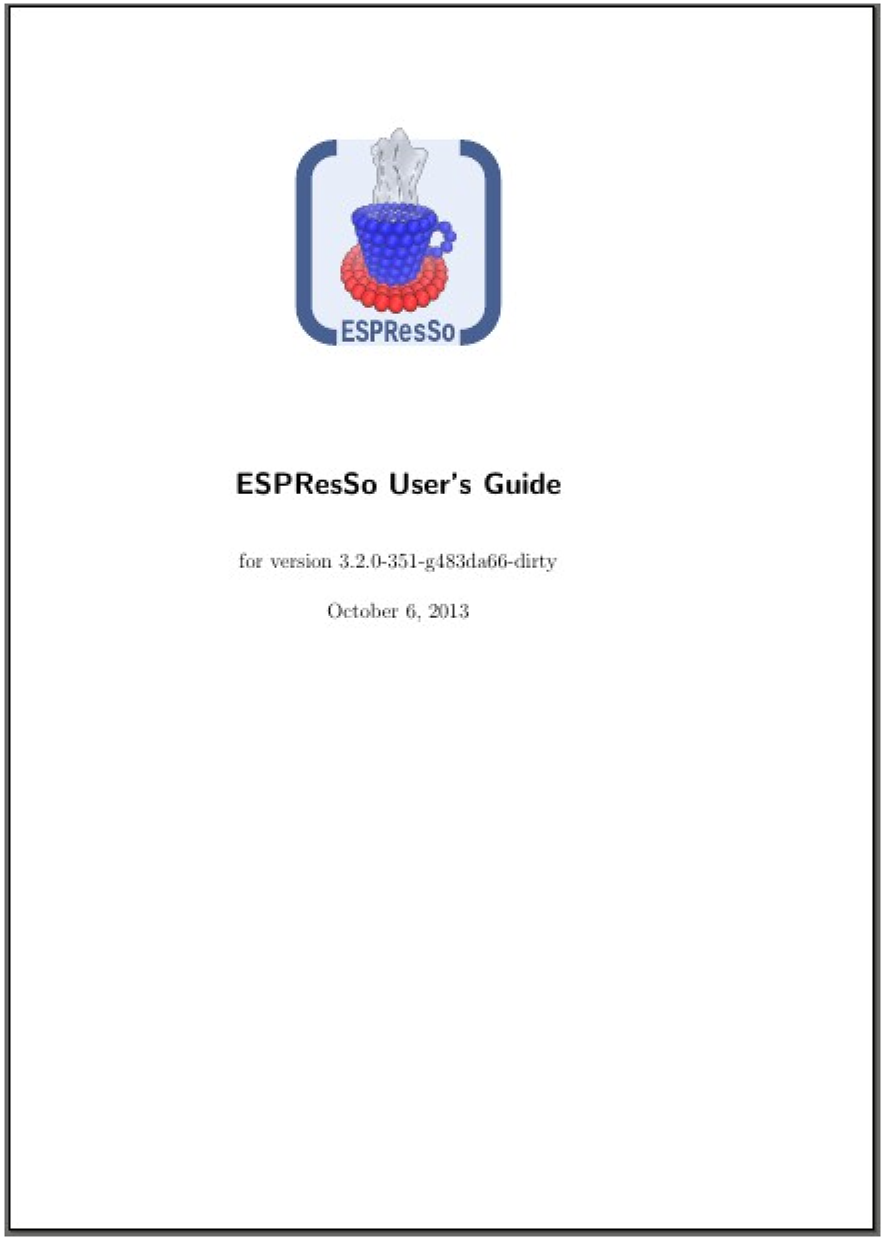
| | |
|---|---|
| Arrays (Hashmap) | `man n array` |
| Namespaces | `man n namespace` |
| String commands | `man n string` |
| Regular expressions | `man n regexp` |
| Packages | `man n package` |
| GUIs via Tk | |
| Etc. | |
| …but that would be too much for now... | |

## Hands-On: Tcl

## ESPResSo

## User Documentation

- User's Guide
  - PDF document
  - In release package (`doc/ug/ug.pdf`)
    - Off-line
    - Matches the release
  - On web site (from build server)
    - Up-to-date
    - Contains ToDo-Boxes
  - Outline
    - Introduction
    - First steps: Quick start
    - Rest: Reference manual
- FAQ (on home page)
  - Not very complete
  - Please contribute!
- Mailing list archive
- Bug tracker



ESPResSo

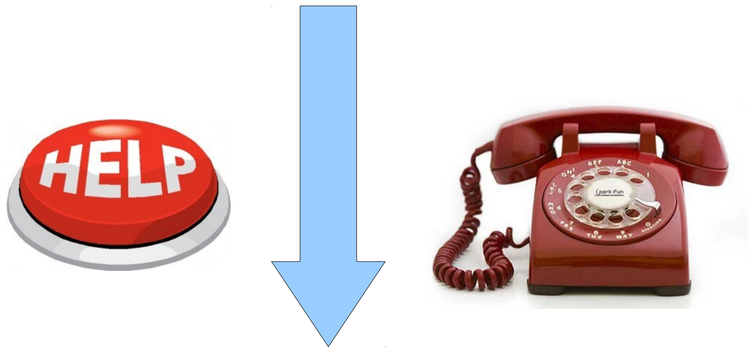
ESPResSo User's Guide

for version 3.2.0-351-g483da66-dirty

October 6, 2013

## Getting Help

- RTFM!
  - FAQ
  - User's Guide
  - Mailing list archives
- Use Mailing List
  - Include version, OS, features
  - Also send replies to the list
    - If you send huge files, better provide a link
    - In a long, detailed discussion you can just send a summary at the end
  - Please remember: the developers are not paid for replying!
  - Please do not write to developers personally
    - All mailings are archived so others can benefit
    - Mailing list reaches everybody



## Requirements

- C++-Compiler (GNU CC is best tested)
- Bourne shell, GNU make
- Tcl (Including headers / devel package!)
- Optional
  - FFTW
    - Including headers
    - Required for P3M
  - MPI
    - e.g. OpenMPI, MPICH
    - Including headers
    - Required for parallel execution
    - Useful to know how to use it
  - CUDA
    - For GPU code
    - … it is getting more all the time

## Compiling

- Necessary evil
  - Few binary packages exist
  - For optimal performance, recompilation is necessary
- Typically 3 steps
  - Configure code `configure`
    - Use `--help` to get options
    - Use CPPFLAGS and LDFLAGS when libraries are installed non-standard
    - Logfile config.log contains additional information
  - Compile code `make`
    - Use `-j` *np* to compile in parallel
  - Run testsuite `make check`
    - Use `processors="1 2"` to specify the numbers of tasks

- Installation is usually *not* required
- Separate *source* and *build* dir
  - The source dir is where the source code resides
  - The build dir is where all files created by the compilation are created
  - No file in the source dir is modified by compilation
  - Call configure from the build dir

```
cd $builddir;
$srcdir/configure
```

## Activating and Deactivating Features

- ESPResSo supports various different features
- Not all features are compiled in
- To check, call `code_info`
- Create file `myconfig.hpp` in build or source dir to change the default set of features
- Use minimal set of features for optimal performance
- The term “feature” is probably not well chosen
  - The code has a lot of features that do not have a compiler switch
  - Goal: remove all features

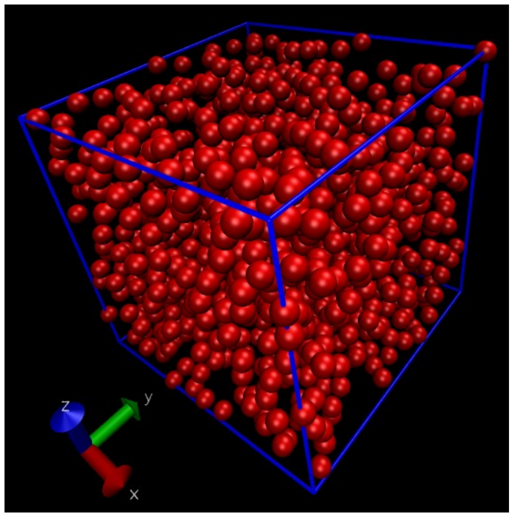
```
#define PARTIAL_PERIODIC
#define ELECTROSTATICS
#define DIPOLES
#define ROTATION
#define ROTATIONAL_INERTIA
#define MDLC
#define EXTERNAL_FORCES
#define CONSTRAINTS
#define MASS
#define EXCLUSIONS
#define COMFORCE
#define COMFIXED
#define MOLFORCES
#define MODES
#define BOND_VIRTUAL
```

Example myconfig.hpp

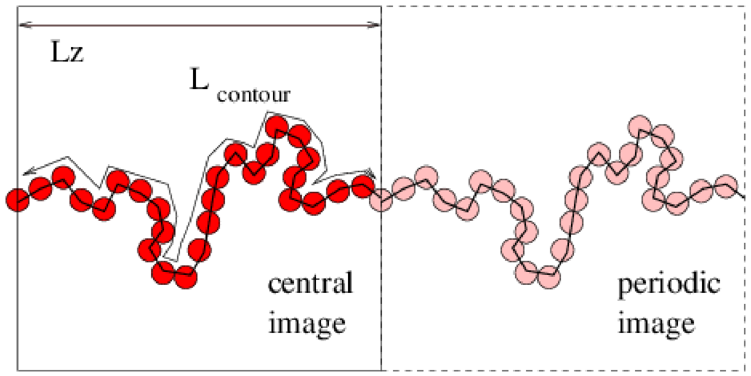
## Hands-On: Compiling ESPResSo

## Writing an ESPResSo Script

- Example files: `lj.tcl` and `stretched_polymer.tcl`
  - `lj.tcl`: Lennard-Jones fluid
  - `stretched_polymer.tcl`: Stretched polymer
- Outline
  - Set up the system
  - Set up the particles
  - Set up the interactions
  - Running the simulation
    - Warmup integration
    - Main integration
  - Analysis
- Sections correspond roughly to chapters in UG
- Detailed command syntax can be found in UG



Snapshot of the LJ system



Schema of the stretched polymer

## Setting Up the System: Global Variables

- Set global variables with `setmd` *`varname value`*
- e.g. Box size, Periodicity, Time step, Skin size, Cell size, …
- Many are set to sensible defaults
- Get global variable with `setmd` *`varname`*
- Many commands can be used without argument to get information

```
# define the system size
setmd box_l $box_size $box_size $box_size

# set up the integrator time step
setmd time_step 0.01

# the skin has no effect on the result, only on the speed
setmd skin 0.4

# uncomment the following to output the box size
#puts [setmd box_l]
```

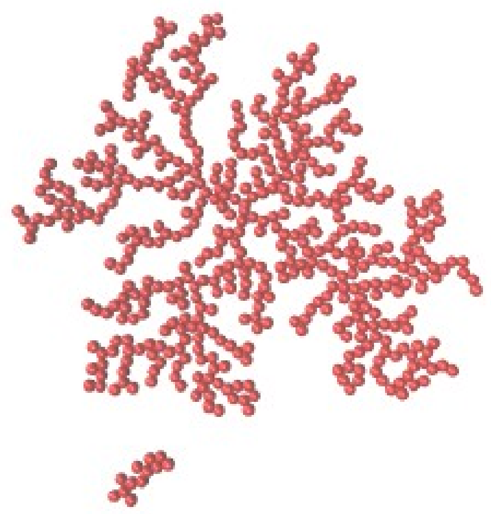
## Setting Up the System: Thermostat

- Command `thermostat`
- Misnomer: used to set ensemble (e.g. Barostat)
- Turn on Langevin thermostat
  `thermostat langevin` *`temperature gamma`*
- Turn off thermostat `thermostat off`
- Other “thermostats”
  - `npt_isotropic` (NPT)
  - Generalized Hybrid Monte-Carlo (GHMC; NPT and more)
  - Dissipative Partice Dynamics (DPD)

```
# set up the thermostat
set langevin_gamma 1.0
thermostat langevin $temperature $langevin_gamma
```

## Setting Up the System: More

- `nemd`: "Non-equilibrium MD": special method for creating a shear flow
- `cellsystem`: Changing the cell system
  - Turn on Domain decomposition (default)
  - Turn off Verlet lists
  - Turn off Cell lists (nsquare)
  - Use "layered" system (only for MMM2D)
- `adress`: Turn on ADResS (better use ESPResSo++)
- `cuda`: Set up CUDA device
- `on_collision`: Turn on collision detection
  - Generate a bond when two particles get close
- `reactions`: Turn on reactions
  - Change the type of a particle when it is close to a catalysator



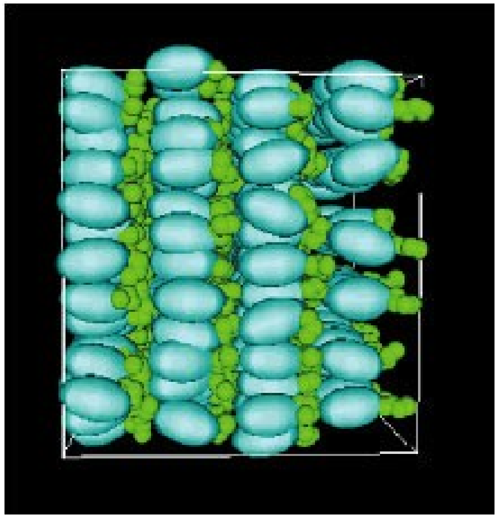
## Setting Up Particles

- Create a single particle: `part` *`pid arguments`*
  - *pid* specifies a numeric id
  - Holes in *pid* order cost memory
- Possible arguments: position (required in first call), velocity, charge, mass, type
- Create bonds to other particles (`bond`)
- Fix particle in one or more directions (`fix`)
- Apply external force to particle (`ext_force`)
- Set individual temperature (feature `LANGEVIN_PER_PARTICLE`)
- Delete a particle (`delete`)
- Get particle properties `part print` *`arguments`*

```
# generate $n_part particles at random positions
for {set i 0} { $i < $n_part } {incr i} {
    set x [expr $box_size*[t_random]]
    set y [expr $box_size*[t_random]]
    set z [expr $box_size*[t_random]]
    part $i pos $x $y $z type 0
}
```

## Oriented Particles, Particle groups, Constraints

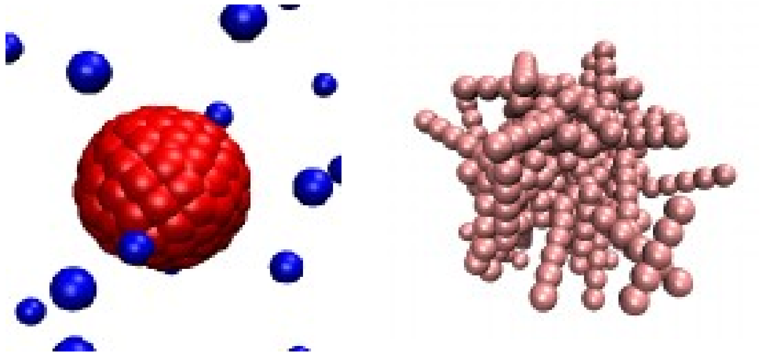
- Feature `ROTATION`
- Particles can be oriented
  - GB-ellispoids (coarse-grained liquid crystals)
  - Directional Lennard-Jones
  - Point-like dipoles
- Quarternion integrator
- Roughly 30% slower!



- Tcl commands to create many particles at once
  - Polymer `polymer`
  - Counterions `counterions`
  - Salt `salt`
  - Diamond polymer networks `diamond`
  - Icosaeder `icosaeder`
  - Copy existing particles `copy_particles`
- Extended objects (“Constraints”) `constraint`
  - Walls, Spheres, Cylinders, Pores, Rods, Rhomboid, Planes
  - External magnetic field

## Virtual Sites

- *Virtual sites* are particles that are not propagated by themselves
- The position depends on the position of other particles (reference particles)
- Forces acting on the virtual sites are transferred to the reference particles
- Create virtual sites with `part`
- Feature `VIRTUAL_SITES_COM:`
  Virtual site in the center-of-mass of other (non-virtual) particles
- Feature `VIRTUAL_SITES_RELATIVE:`
  Virtual sites in a position relative to a (non-virtual) reference particle
  - Allows to create rigid arrangements of particles
    (e.g. raspberry model, rods, …)
  - Requires `ROTATION`



## File I/O and Visualization

- No simple “save state” command! What should be saved? What comprises the state of the simulation?
  - Particle positions, Box size
  - Bonds? Particle Types? Interactions?
  - RNG state? Tcl variables?
  - “Position” in the Tcl code?
- No simple checkpointing!
- Blockfile format
  - Allows to write specified blocks of information:
    `blockfile` *chan* `write particles`
  - ESPResSo can read these blocks:
    `blockfile` *chan* `read auto`
  - ES defines different blocks (particles, bonds, interactions)
- Jump into the main loop needs to be done manually

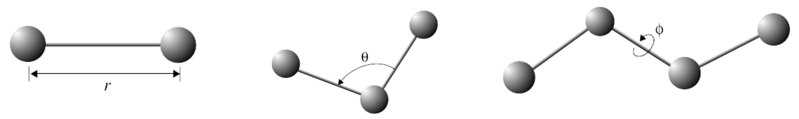
- Visualize best with VMD
  - Off-line
    - Recommended: Create VTF files
    - `writevsf` to output the structure into the VTF file
    - `writevcf` to output a configuration into the VTF file
    - Can also create PSF and PDB
  - On-line
    - VMD has a protocol for on-line visualization
    - In general, off-line is more useful

## Hands-On: First steps and Visualization

## Setting Up Interactions

- *Non-bonded* Short-range Interactions
  - Work between particle *types*
  - Lennard-Jones, Morse, Buckingham, Smooth-step (DPD), …
  - Tabulated
- Bonded interactions
  - Work between two (or more) specific particles (can be set in `part`)
  - Have a *bondid*
  - Bond-length 2-body interaction: Harmonic, FENE, …
  - Bond-angle 3-body interaction: Harmonic, Cosine, …
  - Dihedral 4-body interaction
  - 2-body interactions can be made rigid
    - Not well-tested

- Long-range Interactions
  - Electrostatics
  - Magnetostatics (point-like dipoles)
  - Hydrodynamic interactions
  - → next days
- No force fields built in!



```
set lj_epsilon 1.0
set lj_sigma 1.0
set lj_cutoff 2.5
inter 0 0 lennard-jones \
  $lj_epsilon $lj_sigma \
  $lj_cutoff
puts "Interactions:\n[inter]"
```

## Exclusions, Dynamic Bonding

- Often, neighboring particles in a chain should not interact via non-bonded interactions
- Variant 1:
  Bonded subtracted LJ potential
  `lj_subst`
- Variant 2:
  Exclusions (Feature `EXCLUSIONS`)
  - Explicitly exclude interactions between particles
    `part exclude`
  - Automatically exclude interactions of bonded particles
    `part auto_exclusions`

- `collision_on`:
  Create bonds between particles when they come close
- Useful e.g. for Diffusion-limited aggregation

## Running the simulation

- Main integrator: Velocity Verlet
- Do a number of integration steps: `integrate` *`steps`*
- Use `integrate 0` to update the forces or positions of virtual sites
- Warmup integration
  - Cap the maximal force: `inter` `ljforcecap` *`F_max`*
  - Prevents overlapping particles and very high forces
  - Do steps until the large forces disappear
- Main integration
  - Switching between Tcl and C has an overhead
  - Do as many steps in a single `integrate` command
- Advanced commands for integration
  - Parallel tempering
  - Metadynamics

```
for { set i 0 } { $i < 100 } { incr i } {
    integrate 1000
    .
    .
```

## Analysis in Tcl and in the core

- Analysis can be done in Tcl itself
  - Use `part print` to get particle positions, velocities, etc.
  - Allows for anything you can think of
  - … but maybe slow
    - slow numerics in Tcl
    - not parallel
- `analyze`: predefined observables
  - Implemented in C/C++, possibly in parallel
  - Initialized from Tcl
  - Many different observables
    - Energies, pressures, stress tensor…
    - Minimal distances, RDF, structure factor, …
    - Polymer observables: end-to-end distance, radius of gyration, …
- Commands to analyze several configurations
  - `analyze append`: Store configuration
  - `analyze configs`: Retreive stored configurations
  - Some analysis commands can handle stored configs (e.g. `analyze <rdf>`)
- "Analysis in the core"
  - Allows to turn on some measurements during run of integrate
  - Useful e.g. for MSD

## Hands-On: The rest