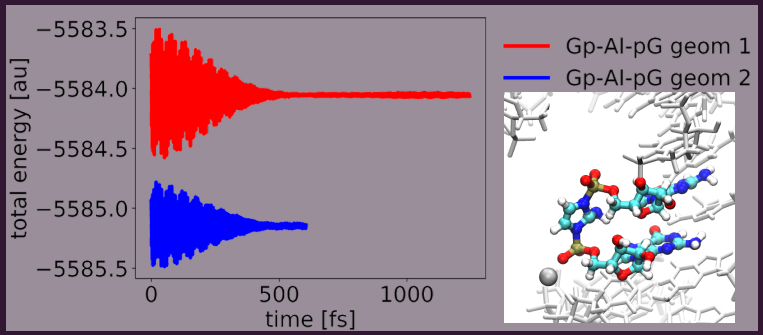
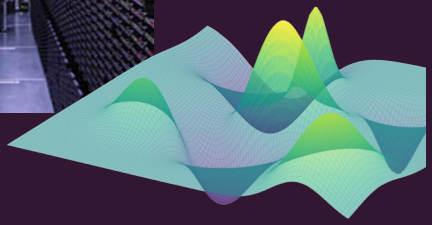
Welcome to MolSim 115!





```
> grep “energy” *.dat
> ssh wonderful@mox.hyak.edu
> python cool_script.py
> vmd
> amber
> date
```

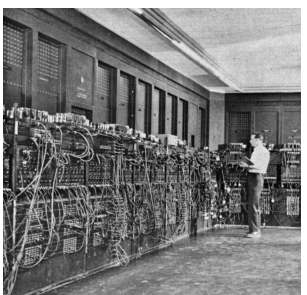
BASH
THE BOURNE-AGAIN SHELL

GROMACS
fast, flexible & free

Why do molecular simulations matter?

Understand and accelerate the design of materials!

We are now in 2020 – incredible availability of computational resources



1946 – ENIAC (Electronic Numerical Integrator and Computer), the first operational electronic digital computer developed for the U.S. Army by J. Presper Eckert and John Mauchly at in Philadelphia.

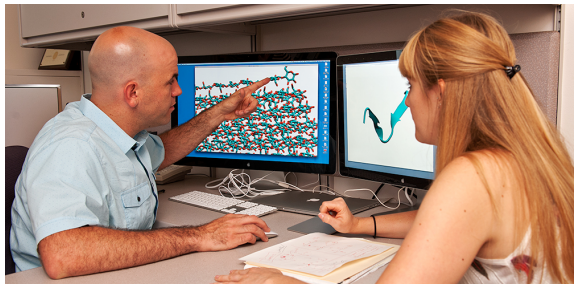
Clock time was **100kHz – 10^5Hz**

Macbook Pro 2019

Clock time **4.1GHz – $4.1 \cdot 10^9$ Hz**

What can you learn from computation?

- Information on stability of structures – energy, kinetics
- Role of the environment on chemical processes
- Optimal ligands for enzymes (drug design)
- Why some catalysts are better than others? Electronic
- Structure, geometry, binding affinity?
- Dynamics of molecules as a function of temperature, pressure
- Design of chemical reactions



Course Goal and Content

GOAL: Understand and apply models / theories for the computation and visualization of molecular properties

Course content

- Shell scripting (bash), Python, Github
- Electronic Structure/Quantum Calculations (HF, DFT)
- Force fields, classical MD
- Simulation Parameters (Thermostat/Barostats/Cutoffs)/Gromacs
- VMD / Visualization
- Trajectory Analysis (RDF/RMSD/Autocorrelation)
- Parallel Computing
- Master equations

Course Instructors

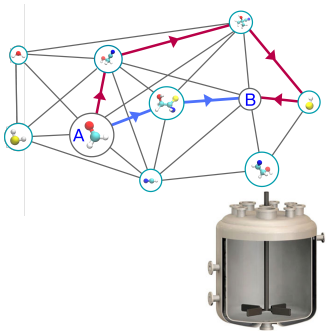
Sarah Alamdari

Office: BNS B17

Email: salamda1@uw.edu

Course Instructors

Professor Stéphanie Valleau



Office: Benson Hall 351

Email: valleau+cheme565@uw.edu

Course Webpage & Github

- Course webpage: https://canvas.uw.edu/courses/1378847

- Github: We will also be using Github https://github.com/UWPRG/MOLSIM-2020

  Course syllabus, documents, code, homework projects, grades, announcements will be posted on the webpage – please check it regularly!

Homework / Final Project / Grading

- One homework every week (9 total)
- Final project

Assignment	Contribution to final grade
Homeworks	60%
Final Project	40%

Questions?

Outline for today

Introduction to bash scripting

- What is bash?
- Dealing with files and directories
- Vi a visual file editor
- Special characters
- Bash scripts
- Practice writing a bash script

What is Bash?

Bash (*Bourne-again shell*) is a **Unix shell** and **command language** written by Brian Fox in 1989

- Bash is a program which allows you to **read files**, **start** other **programs**, do math and control devices. It provides the user with an interface to interact with other programs.
- Bash is a text language, there is no graphic interface

Let's start playing with Bash by using a **terminal** as a graphical interface!

Note: Other examples of shell programs are C shell (csh), Z shell (zsh)

Dealing with files and directories

General help / info commands `> bash --help` `> man command`

Commands to manage and work with files / directories

Command	Options	Purpose	Example
ls	-al -hl	list files	ls –al *
cd		change directory	cd ~
mkdir	-f	create directory	mkdir testing
vi		text editor	vi script.sh
chmod		change permissions	chmod +xg script.sh
cp		copy	cp script.sh new_script.sh
mv		move file or directory	mv script.sh testing/
rename		rename file	rename ‘s/.sh/1.sh/’ testing/*
rm	-rvf	remove	rm new_script.sh
pwd		lists path of current directory	pwd

https://www.computerhope.com/unix/rename.htm
https://ss64.com/bash/

Vi – visual editor

Other common editors: emacs, visual studio, atom, sublime

Some commands (for more see e.g. http://www.lagmonster.org/docs/vi2.html)

Command	Where	Purpose
`vi filename`	terminal	open or create a file names filename
`[ESC]`	file	returns editor to command mode
`i`	file in command mode	go into edit mode
`:x`	file in command mode	quit vi, writing out modified file to filename
`:wq`	file in command mode	quit vi, writing out modified file to filename
`:q`	file in command mode	quit or exit vi
`:q!`	file in command mode	quit vi and do not save changes
`:n`	file in command mode	move to line n (first line is n=0)
`:$`	file in command mode	move to last line
`$`	file in command mode	move to end of line
`0`	file in command mode	move to start of line
`dd`	file in command mode	delete current line
`Ndd`	file in command mode	delete N lines beginning with the current line

Vi – visual editor

Some commands (for more see e.g. http://www.lagmonster.org/docs/vi2.html)

Command	Where	Purpose
`yy`	file in command mode	copy the current line into the buffer
`Nyy`	file in command mode	copy the next N lines including the current line into the buffer
`p`	file in command mode	paste
`/`	file in command mode	search for string

Dealing with files & other bash commands

Command	Options	Purpose	Example
`echo`		prints the text your write after echo	`echo "Hello world! I love Chemistry!"`
`cat`		lists the content of a file to terminal	`cat script.sh`
`tac`		lists the content of a file to terminal in reverse order	`tac script.sh`
`diff`		display the differences between two files	`diff script.sh script2.sh`
`head`	`-n`	display top n lines of a file	`head –n 1 script.sh`
`tail`	`-n`	display last n lines of a file	`tail –n 10 script.sh`
`man`		help manual	`man cp`
`paste`		paste two files into one	`paste script.sh script2.sh > script3.sh`
`sleep`		delay for a specific time	`sleep 10 ; echo "Done sleeping"`

use `man` command to see all the options

Bash special characters

Character	Options	Purpose	Example
>	1> 2>	character is placed after a command to forward its output 1, 2 are for output and error	`cat script.sh > foutput.sh`
;		separates commands	`sleep 10 ; cat script.sh#`
#		comment	`echo "I love the tutorial" #but I'm hungry`
|		pipe – redirects output of first command to second	`cat *.sh | sort`
\		escape	`echo "I love the tutorial" \#but I'm hungry`
"		partial quoting – preserves from interpretation for most special characters	
'		full quoting – preserves all characters from interpretation	
./		Execute in same directory	`./script.sh`
sh		Execute a file which is in another directory	`sh ~/script.sh`

http://tldp.org/LDP/abs/html/special-chars.html

Bash special characters & extra commands

Character	Options	Purpose	Example
*		wild card	`echo *`
$		call a variable	`a=1; echo $a`
&		run in background	
~		home directory	`pwd; cd ~ ; pwd`
Ctl-C		terminate a foreground job	`./script5.sh ; Ctl-C`
clear		clear terminal	`clear`
wc		word count	`cat script.sh | wc -l`

http://tldp.org/LDP/abs/html/special-chars.html

Bash scripts

If you close your terminal you loose what you typed! 

One way to get it back is to look at the `.bash_history` file in your home directory 

Best practice is to write and save a bash script ! 

To generate a bash script from your terminal type

```
> vi my_bash_script.sh
```

This will open an empty file (call it what you prefer).
To allow the system to identify it is a bash script, type `#!/bin/bash` in the first line of the file

Let's practice Bash scripting

if statement (conditional)

```
if [[ condition ]]; then
        …
elif [[ condition ]]; then
        …
fi
```

for and **while** statement (loops)

```
for ((i=0; i<100; i++)); do
        …
done
while []; do … ; done
```

For more info see

http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html

Exercise

Generate 1000 files, move them all to a new directory and rename each file with a different name

Awk

Awk – basic function is to search files for lines that contain certain patterns, to print parts of files, etc.

https://www.gnu.org/software/gawk/manual/gawk.html

It is useful to **print lines and columns** of text:

```
> head file.txt
> awk '{ print $1 $9 }' file.txt
> awk '{ print $1 "\t" $9 }' file.txt
> awk '{ print $1 "\t" $9 "\n" }' file.txt
> awk 'NR==2' file.txt
> i=23; awk "NR==$i" file.txt
```

Sed

Sed – is used to perform text transformation of files provided as input.

https://www.gnu.org/software/sed/manual/sed.html

Useful to replace text or numbers in files:

```
> head file.txt
> sed 's/D/ZZZ/' file.txt > output.txt
> sed –i 's/D/ZZZ/' file.txt (Does not work on MacOSX terminal)
> sed –i '' 's/D/ZZZ/' file.txt (Works on MacOSX terminal)
> sed 's/1./122./g' file.txt
```

If you install brew on MacOS and run

```
brew install coreutils
```

You won't have probems with the difference between Mac and Linux

Grep

Grep – Given one or more patterns, grep searches input files for matches to the patterns.

https://www.gnu.org/software/grep/manual/grep.html

```
> grep “Hello” script.sh
> grep “1.” file.txt
> grep “Saddle” output.dat
> grep –C 1 “Hello” script.sh
> grep –C 2 “Hello” script.sh
> grep –C 1 “6.39045411” output.dat
```