



Introduction to HPC on Blanca



Research Computing
UNIVERSITY OF COLORADO BOULDER

10/14/20

PAPP: Intro to Blanca

1

Be Boulder.

Introduction to HPC on Blanca

- Andrew Monaghan
- *Email:* Andrew.Monaghan@Colorado.edu
- *RC Homepage:* <https://www.colorado.edu/rc>
- Slides available for download at:
https://github.com/ResearchComputing/BLANCA_PAPP_FALL_2020



Outline for this presentation

- Overview of RC, Blanca
- Logging in
- Basic Linux commands
- File editing
- Linux filesystem
- Environment variables
- Software modules on Blanca
- Bash scripts and Job Submission



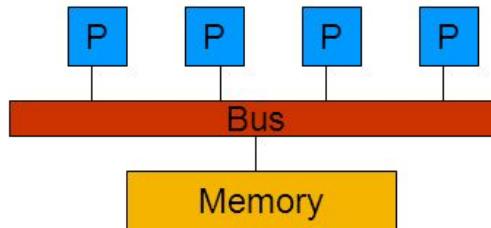
What is Research Computing?

- Provide services for researchers that include:
 - Large scale computing
 - Data storage
 - High speed data transfer
 - Data management support
 - Consulting
 - Training
- We are likely best known for:
 - Summit Supercomputer (~12,000 cores)
 - **Blanca "condo" cluster** (~4,000 cores)
 - PetaLibrary storage

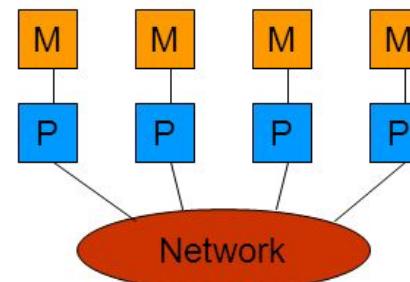
What Would I Use Blanca For?

Solving large problems that require more:

- Memory than you have on your personal computer
- Cores/nodes/power than you have on your personal computer
- Blanca is set up for both shared memory (single node) and distributed memory (multi-node) parallelization.
 - Can also use Summit for "big" distributed memory parallelization



- Shared memory



- Distributed memory

Source: https://images.slideplayer.com/25/7599921/slides/slide_4.jpg

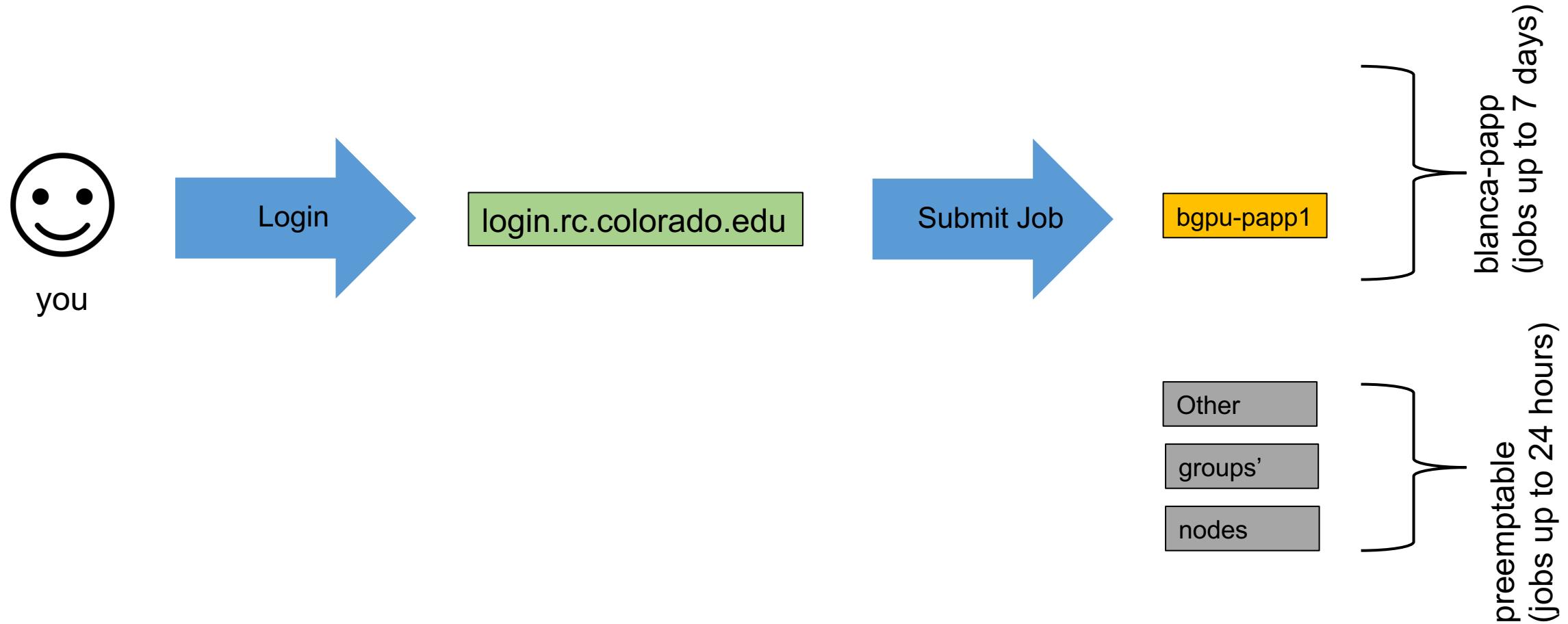
Blanca

- A “condo” cluster whereby individual research groups own nodes
- List of nodes and groups can be found [here](#)
- Users have dedicated access to their group’s nodes (e.g., blanca-papp)
 - Jobs up to 7 days long.
 - Can also run ‘preemptable’ jobs on other groups nodes (jobs up to 24 hours long)
- More documentation on Blanca:
<https://curc.readthedocs.io/en/latest/access/blanca.html>

Blanca PAPP node

- bgpu-papp1, (1 node)
 - 32 core Intel Skylake w/ 5.8 GB RAM/core (192 GB RAM total)
 - 1 Tesla V100 GPU w/ 16 GB RAM
- This means that you can use the node for both CPU and GPU applications.

Blanca Workflow



Logging In

- ssh [<identikey>@login.rc.colorado.edu](ssh <identikey>@login.rc.colorado.edu)
- Enter your identikey_password
- Authenticate by accepting the Duo push to your smartphone
 - Can also authenticate by text message, phone call, or token
- More info here:
<https://curc.readthedocs.io/en/latest/access/logging-in.html>

Linux

- Part of the Unix-like family of operating systems.
- Started in early ‘90s
- Several distributions are available – from enterprise-grade, like RedHat Linux (RHEL), to more consumer-focused, like Ubuntu.
 - Blanca nodes presently run RHEL7
- Runs on everything from embedded systems to supercomputers.
- Linux is simple, flexible, fast, many potent tools



Anatomy of a Linux command

- command [flags] [flag arguments] [target(s)]
 - `ls -l myworkdir/`
- Case is important!
- Help on commands is available through the "man" command (short for manual). E.g.,
 - `man ls`



File and directory related commands

- **pwd** – prints full path to current directory
- **cd** – changes directory; can use full or relative path as target
- **mkdir** – creates a subdirectory in the current directory
- **rm** – removes a file (**rm -r** removes a directory and all of its contents)
- **cp** – copies a file
- **mv** – moves (or renames) a file or directory
- **ls** – lists the contents of a directory (**ls -l** gives detailed listing)
- **chmod/chown** – change permissions or ownership
- **df** – displays filesystems and their sizes
- **du** – shows disk usage (**du -sh** more useful)

Process and Program related commands

- **ps** – lists processes
- **top** – shows processes currently using the CPUs
- **kill** – sends a signal to a process (use “kill -9” for best results)
- **time** – shows how much time a process has used
- **free** – memory usage



File-viewing commands

- **more** – displays a file one screen at a time
- **cat** – prints entire file to the screen
- **head** – prints the first few lines of a file
- **tail** – prints the last few lines of a file
- **diff** – shows differences between two files
- **grep** – prints lines containing a string or other regular expression (ps -ef | grep XX)
- **sort** – sorts lines in a file
- **find** – searches for files that meet specified criteria
- **wc** – count words, lines, or characters in a file

File editing with nano

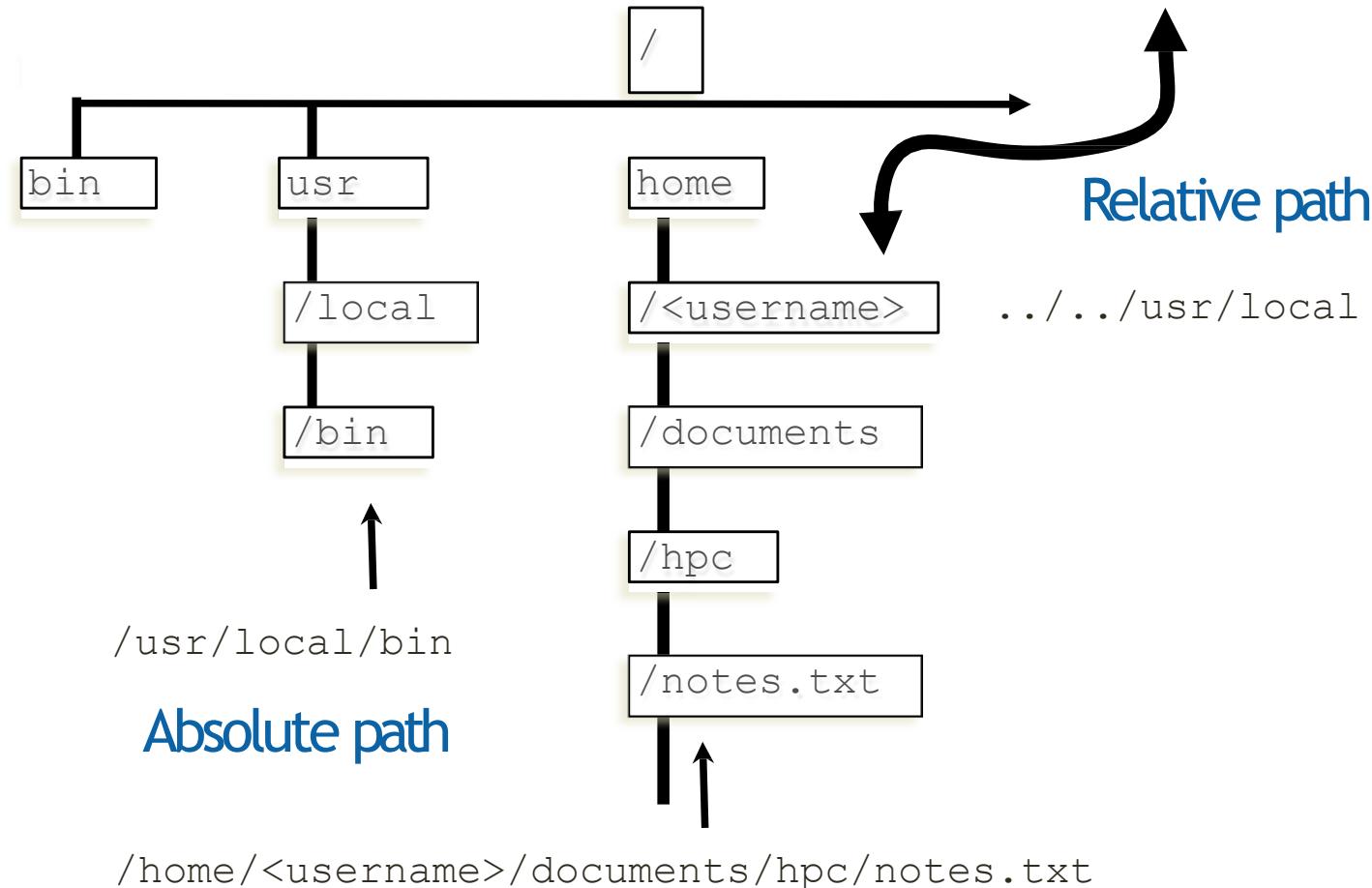
- To edit a file:
 - `nano myfile.txt`
- From within Nano:
 - `Ctrl+o` save (need to confirm filename)
 - `Ctrl+x` exit
 - `Ctrl+k` cut
 - `Ctrl+u` paste
- Other popular Linux editors: vi, emacs

The Linux Filesystem

- System of arranging files on disk
- Consists of directories (folders) that can contain files or other directories
- Levels in full paths separated by *forward* slashes, e.g.
- /home/user/scripts/analyze_data.sh
- Case-sensitive; spaces in names discouraged
- Some shorthand:
 - . (the current directory)
 - .. (the directory one level above)
 - ~ (home directory)

Filesystem

Multiple Users



Your personal directories on Blanca

- `/home/<username>`
 - Very small: 2GB.
 - Backed up daily.
 - Good for ‘can’t lose’ files
- `/projects/<username>`
 - 250 GB
 - Backed up regularly
 - Good for storing scripts, self-installed software, some data
- `/rc_scratch/<username>`
 - Group-shared 100 TB partition.
 - Good for jobs with lots of I/O
 - Not backed up
 - Temporary: data deleted 90 days from creation.

Environment variables

- Environment variables store important information needed by Linux users, programs, etc.
- Type ‘`env`’ to see your currently set environment variables
- Useful Environment variables:
 - `PATH`: directories to search for commands
 - `HOME`: home directory
 - `PWD`: current working directory
 - `USER`: username
 - `LD_LIBRARY_PATH`: directories to search for shared objects (dynamically-loaded libs)

Shell Wildcards and Special Characters

- * - matches zero or more characters
- ? - matches a single character
- # - comment; rest of the line is ignored
- \ - escape; don't interpret the next character



Software

- Common software is available to everyone on the systems
- Research Computing uses modules to manage software
 - You load modules to prepare your environment for using software
 - Modules set any environment variables, paths, etc.
 - Set environment so application can find appropriate libraries, etc.
- You can also install your own software
 - It is best if you are responsible for support
 - We are happy to assist



Using Modules

- Must be on a **compute** node to browse the modules (e.g, bnode0501)
- To set up your environment to use a software package, type
module load <package>/<version>
- Some modules might require a specific hierarchy to load
 - For some modules, you may need to specify a specific version
 - For example, **module load R/3.3.0**
 - For other modules, you may be able to be more generic
 - For example, **module load matlab**
- Some modules may require you to first load other modules that they depend on
- To find dependencies for a module, type **module spider <package>**
- To find out what software is available, you can type **module avail**

Next Topics: Bash Scripting and Job submission

- Bash scripts (needed to submit jobs)
- Examples of submitting jobs to the supercomputer
 - Batch jobs
 - Interactive jobs



Bash

- A shell is the environment in which commands are interpreted in Linux
- GNU/Linux provides various shells; **bash is the most popular**
 - sh Bourne-again Shell (Bash)
 - csh C-Shell
 - tcsh Tc-Shell
 - ksh Korn-shell
- Shell scripts are files containing collections of commands for Linux systems that can be executed as programs...

Bash Script

- To create a bash shell script file, the first line must be:

```
#!/bin/bash
```

- Program loader recognizes the `#!` as an interpreter directive. This is followed by `/bin/bash` which tells the OS which shell should be used.
- Example:

```
#!/bin/bash

cd /home/$USER

hostname
echo "Hello!" > file.out
```



Why I am learning this?

Bash shell scripts are used to create job scripts that -- when submitted to the Slurm job manager -- request resources and execute commands and software on RMACC Summit.

Let's get started!...



Jobs and Job Scripting

- **Job** – a system allotment of resources that run a particular application.
- **Slurm** -- resource manager
- Because Summit and Blanca are shared resources among a variety of groups on campus, users must run their applications through jobs.
 - Ensures everyone can utilize the system
 - No one person taking up too much of the system!
- Blanca is a bit different – Dr. Carlson’s group owns a portion of the system that is solely shared between group members.
 - Substantially shorter wait times!

Job Types

- **Batch Jobs**
 - Non-interactive allotment of resources that run applications in the background
 - Think “baking a batch of cookies”
 - Useful for:
 - Applications that take a substantial amount of time
 - Non-interactive Applications
- **Interactive Jobs**
 - Interactive allotments or resources where user run their applications manually
 - Useful for:
 - Debugging Applications
 - Running GUI Applications

Submitting Batch Jobs

- `sbatch` command: submit a batch job to Slurm
- The `sbatch` command usually takes in 1 parameter: A job script.
 - Job scripts provide all information on what is needed for the job.
 - Parameters can be overwritten or added externally by specifying the parameter as a flag.
- Example:

```
sbatch test.sh
```

or

```
sbatch test.sh --time=02:00:00
```



Anatomy of a job script

Job parameters

```
#!/bin/bash  
  
#SBATCH --ntasks=1          # Number of requested tasks  
#SBATCH --time=0:01:00       # Max wall time  
#SBATCH --partition=blanca-papp # Specify Papp nodes  
#SBATCH --qos=blanca-papp    # Specify you are in papp group  
#SBATCH --account=blanca-papp # Specify you are in papp account  
#SBATCH --output=test_%j.out  # Rename standard output file
```

Your job commands

```
# Now run job (commands below here)  
  
# Purge and load needed modules  
module purge  
module load R  
  
# Run commands  
Rscript myscript.R
```



Job Parameters

- Allocation: `--account=<account-name>`
- Partition: `--partition=<partition-name>`
- Number of nodes: `--nodes=<nodes>`
- Number of Tasks: `--ntasks=<number-of-tasks>`
- Quality of service: `--qos=<qos>`
- Reservation: `--reservation=<name>`
- Wall time: `--time=<wall-time>`
- Job Name: `--job-name=<jobname>`
- Output File: `--output=<outputname>`

*More on slurm commands:
<https://slurm.schedmd.com/quickstart.html>*

FYI: You do NOT actually type <> above – this designates something specific you as a user must enter about your job

PAPP QoS, Account, and Partition

- When using the PAPP Blanca nodes, job scripts should specify the following information:

```
--account=blanca-papp  
--partition=blanca-papp  
--qos=blanca-papp
```

- Users of Blanca also have access to a low-priority preemptible QoS that will attempt to run your job on any available node on Blanca
 - Preemptable jobs will be booted off the node and restarted at a later time if a high priority user runs a job on the node.

```
--qos=preemptable
```

Submitting Your First Job!

Submit a slurm job with the following instructions:

1. The job should run the Linux “hostname” command
2. The job script will be named: `submit_hostname.sh`
3. The job will run on 1 core of 1 node
4. We will request a 1 minute wall time
5. Run from the blanca-papp QOS
6. Run on the blanca-papp partition and account
7. Use the ‘papp’ reservation
 - *This is only for this workshop*



submit_hostname.sh

```
#!/bin/bash

#SBATCH --nodes=1          # Number of requested nodes
#SBATCH --ntasks=1         # Number of requested cores
#SBATCH --time=00:01:00     # Max wall time
#SBATCH --qos=blanca-papp  # Specify QOS;
#SBATCH --partition=blanca-papp # Specify nodes
#SBATCH --account=blanca-papp # Specify account
#SBATCH --output=hostname_%j.out # Rename standard output file
#SBATCH --job-name=hostname # Job submission name

# purge all existing modules
module purge

hostname
echo ""
nvidia-smi
```

Running the job script

Submit the job:

```
$ sbatch --reservation=papp submit_hostname.sh  
(note “–reservation=papp” for this class only)
```

Look at the job output:

```
$ cat hostname_<job-id>.out
```

Check the status of the job:

```
$ squeue / $ squeue -u <user> /  
$ squeue -q <qos>  
...or  
$ sacct / $ sacct --format=<options>
```

(*note that *<job-id>* is your job number)

```
$ scontrol show job <job number>
```

More on slurm commands: <https://slurm.schedmd.com/quickstart.html>



Interactive jobs

- Sometimes we want to work with a program in real time
 - Great for testing, debugging
- For example, let's run an interactive job and use "python"...



Running an interactive job

- To work with R interactively, we request time from Summit
- When the resources become available the job starts

- Commands to run:

```
sinteractive --time=00:10:00
```

- Once we receive a prompt, then:

```
Source /curc/sw/anaconda/default  
python
```

- Once we finish we must exit! (job will time out eventually)



Topics we didn't cover today

- Job arrays (when you need to run lots of similar tasks)
- Running preemptable jobs
- CURC JupyterHub (for running Jupyter Notebooks on Blanca)
- Use of conda on CURC
- ..and lots of other cool stuff
- See: <https://curc.readthedocs.io>

Thanks!

- Please fill out the survey: <http://tinyurl.com/curc-survey18>
- Contact: rc-help@Colorado.edu, Andrew.Monaghan@Colorado.edu
- Course materials for today:
 - https://github.com/ResearchComputing/BLANCA_PAPP_FALL_2020
- Blanca (and other) documentation: <https://curc.readthedocs.io/en/latest/access/blanca.html>
- Slurm Commands: <https://slurm.schedmd.com/quickstart.html>
- More detailed tutorials from our “HPC Fundamentals” course:
 - https://github.com/ResearchComputing/Fundamentals_HPC_Fall_2019

Supplemental Slides



Modes (aka permissions)

- Three classes of users:
 - User (u) aka “owner”
 - Group (g)
 - Other (o)
- Three types of permissions
 - Read (r)
 - Write (w)
 - Execute (x)

.. own grp oth
-|---|---|---

drwxr-xr--

Modes (continued)

- chmod changes modes:
- To add write and execute permission for your group:
`chmod g+wx filename`
- To remove execute permission for others:
`chmod o-x filename`
- To set only read and execute for your group and others:
`chmod go=rx filename`