



UW Tower Data Center
(Tours available when COVID isn't around!)

Research Computing Club Presents

Hyak Training Session

February 3, 2021 (3:30pm PST)

Jesse Prelesnik (Training Coordinator)

Outline

Where I assume you're at right now:

"I finished the steps to getting access* to Hyak, but don't really know what to do now. How do I get up and running with the computing I want to do?"

I. Hyak overview

- A. Hyak architecture
- B. Logging on to Hyak

II. Navigating Hyak

- A. Basic commands
- B. Important locations
- C. Transferring files

III. Slurm

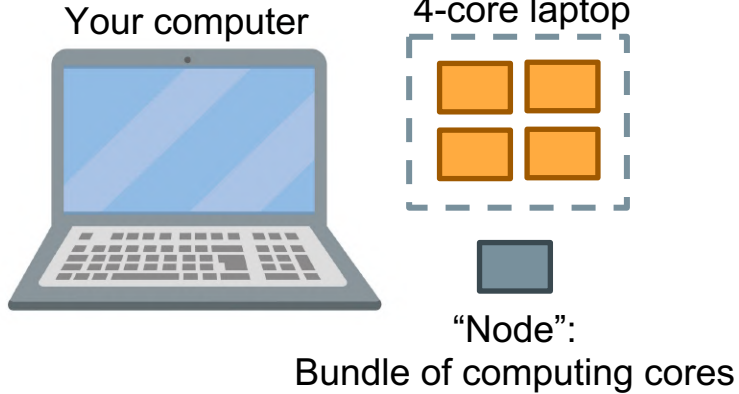
- A. Running a job
- B. Commands
- C. Modules
- D. Interactivity

IV. Other resources

- A. Topics outside of this tutorial
- B. Places to get help

* <https://depts.washington.edu/uwrcc/getting-started-2/getting-started/>

Your PC vs. Hyak



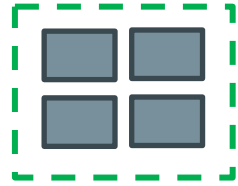
Mox nodes:

- 28 cores
- 128 GB RAM
- 92 regular nodes - stf partition

} *parallel*
computing

Hyak is a “condominium” supercomputing cluster:

Bundle of nodes



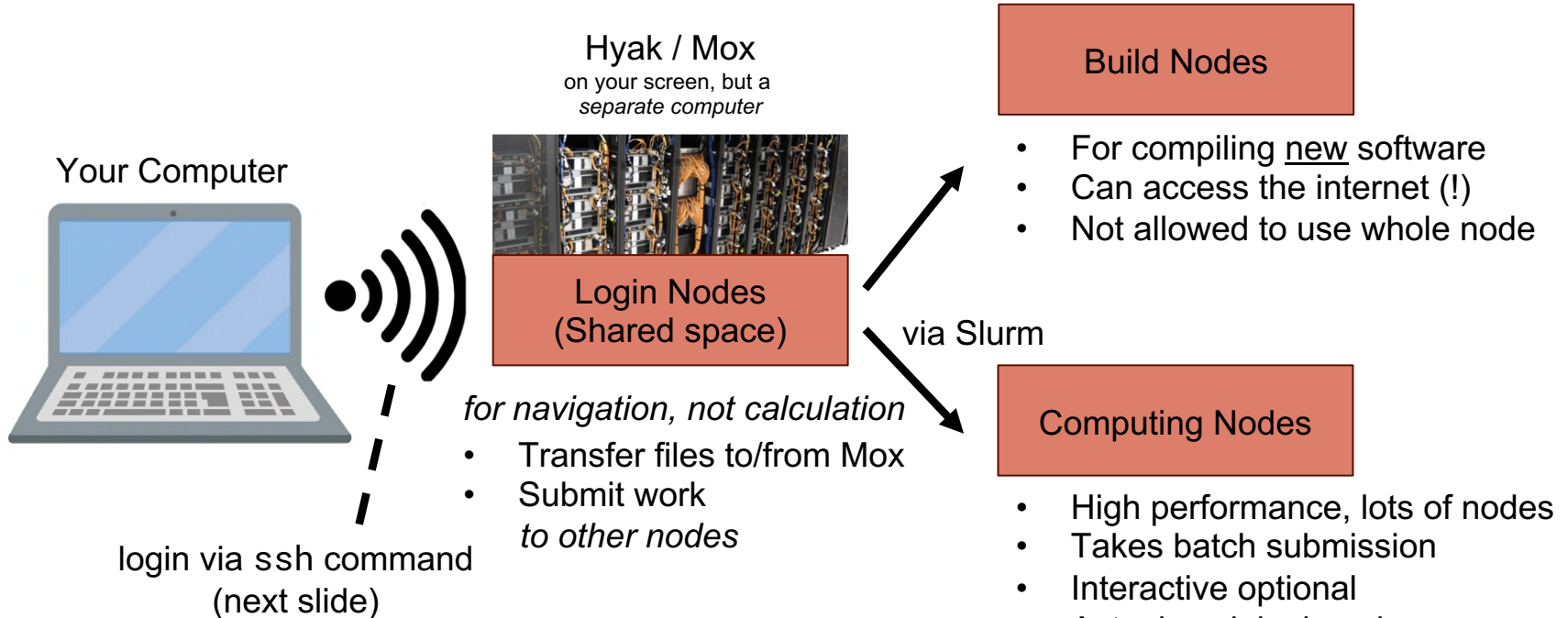
- Research groups own nodes in [partitions](#).
 - There’s one partition for all tuition-paying students:
 - Student Technology Fee – “stf”
- In addition to our partitions, groups have access to the `build` and `ckpt` partitions
- ~10,000 cores in total
 - A typical laptop might have 4-8 cores (CPUs)

One cluster for right now: “mox” ([current](#)), soon “klone” ([soft-launches this quarter](#))

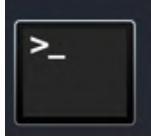
An old cluster, “ikt”, was just retired and is no longer usable

Accessing Nodes - Hyak “Architecture”

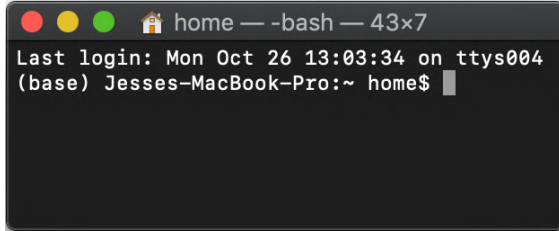
- All nodes share the same filesystem (except /tmp)
- Different “kinds” of nodes have different purposes
 - Don’t “clog up” organizational nodes



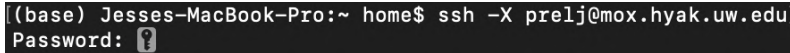
Logging on to Hyak



Terminal – “command line”
Easily accessible on Mac/Linux



“Secure Shell” ssh command
Enables connection to remote machine

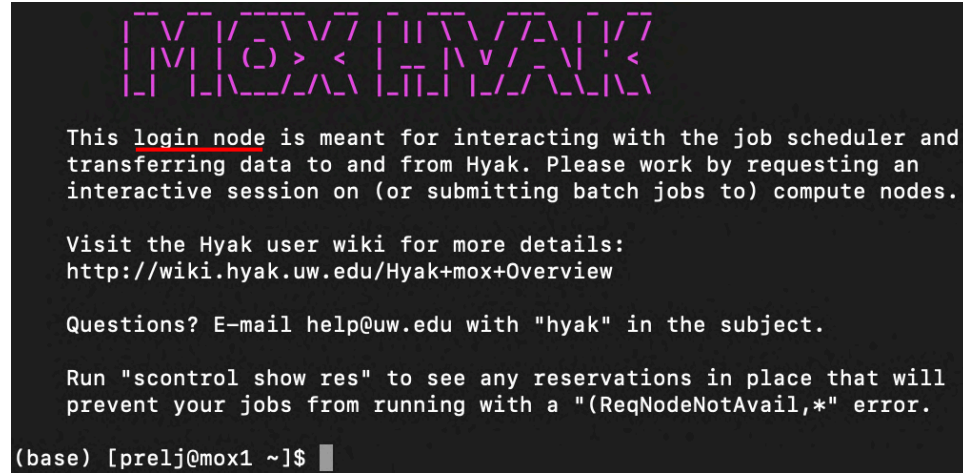


```
ssh <uwnetid>@mox.hyak.uw.edu
```



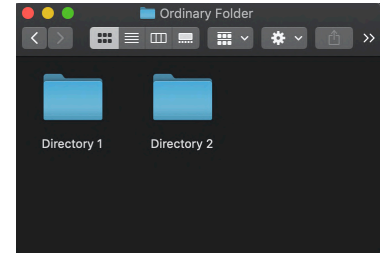
Two-factor authentication required
(Through UW-IT)

You're in!



- Notice you are defaulted to the *login node*
- Jobs are not run here

“Graphical user interface” (GUI)

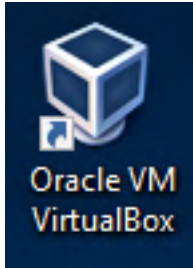


No GUI

Navigate via command line

What if I'm on Windows?

You'll need a terminal somehow, and Windows does not provide a good one by default



My preferred route: “Virtual Machine” (VM)
Sets up a mini environment that emulates a different operating system
Lets your Windows machine have some non-Windows capabilities (like terminal)

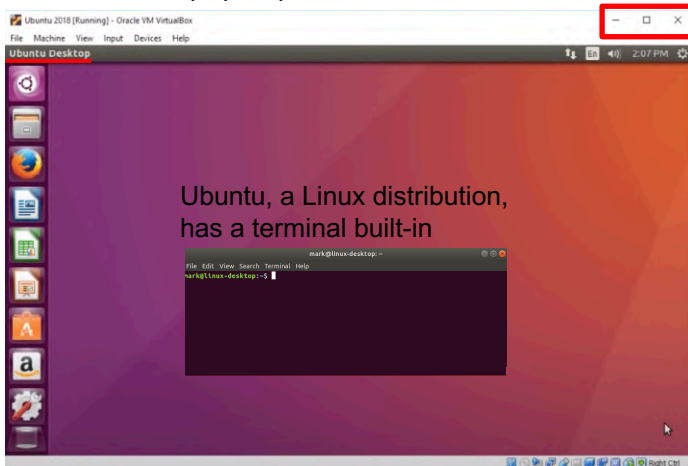
“Oracle” is one example of a VM

Other options for Windows:

- PuTTY (Terminal emulator)
- CygWin (Runtime environment)
- GitBash
- Windows Subsystem for Linux
- WinSCP (just for transferring files)
- cmdr
- xshell
- ... and probably more!

How you get a terminal does not matter,
they are all effectively the same

New window pops up within Windows to interface with Ubuntu



More help:

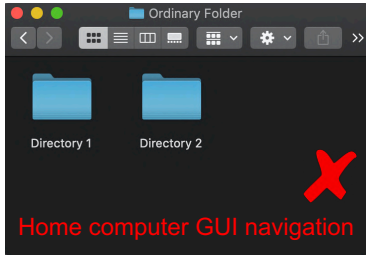
<http://wiki.cac.washington.edu/display/hyakusers/Logging+In>



I'm in

... Now what?

Exploring Hyak



Essential First Actions:

- `mkdir`
 - “Make directory”
 - Create a place to put your files
- `vim`, `emacs`, `nano` (etc.)
 - Command line text editors
 - These can be a pain, but are important!

Each research group has a folder in `gscratch`

Hyak file systems through **command line**



Where in the computer am I?
`pwd` - “Print working (current) directory”

```
(base) [prelj@mox1 ~]$ pwd  
/usr/lusers/prelj
```

My home directory (see next slide)

What folders and files are here?
`ls` - “List” files in current directory (folder)

```
(base) [prelj@mox1 ~]$ ls  
d g_perf intel opt
```

Let’s go to where everybody stores files
`cd` - “Change directory”

```
(base) [prelj@mox1 ~]$ cd /gscratch  
(base) [prelj@mox1 gscratch]$
```


Important locations on Hyak (get there with cd)

- ★ `/gscratch/stf`
 - Main work location for stf users
 - **Any files untouched for >30 days will be auto-deleted!!**
- ★ `/usr/lusers/<username>`
 - Home directory “~”, you are put here by default
 - **Only 10 GB of storage per user**
- `/tmp`
 - Node local storage (separate from shared filesystem)
- `/sw`
 - All software installs
- `/sw/contrib`
 - User installed software
- `/sw/modules-1.775/modulefiles/contrib`
 - User added modulefiles



Where pre-installed programs (and their supporting files) are housed

You will seldom need to go here, as programs can be accessed more easily (in a way we will soon see)

Basic shell (bash) commands: Try them out!

File system manipulation:

- `ls`
 - “List” files in current directory (folder)
- `cd`
 - “Change directory”
- `pwd`
 - “Print working (current) directory”
- `mkdir`
 - “Make directory”
- `mv`
 - Move (rename) file or directory
- `cp`
 - “Copy” files and/or directories (-r)
- `rm`
 - “Remove” files and/or directories (-r)

File editing and compression:

- `nano`
 - Edit files
 - Other editors: vim, emacs, etc.
- `tar`
 - Compress for a “tape archive”
- `zip` (and `unzip`)
 - Compress via zip algorithm
 - Windows friendly

Many, many more

- `man`
- `find`
- `top`
- `kill`
- `chmod`
- `curl`
- `grep`
- `sed`
- ...

Transferring files

scp – Secure Copy

- Send file To Mox:

```
scp <path/to/file> <username>@mox.hyak.uw.edu:<path/to/dest>
```

On your computer

Your login from ssh

Place on Mox you want
to put the file

- Get file From Mox:

```
scp <username>@mox.hyak.uw.edu:<path/to/file> <path/to/dest>
```

Flag `-r` needed to copy whole folder at once

An alternative copier for synchronizing directories is `scopy`

Lolo:

- Magnetic tape archive (lolo archive)
- For long term storage - **only store compressed large files!**
- STF location: `/archive/hyak/stf`
- Transfer files the same as between local and Hyak

Running Your First Job

Real computing jobs are passed to Slurm

- Ensures fair share between users
- Run interactive or batch jobs
- Allows for running on the ckpt queue
 - (more on this later)

Some information is needed during job setup:

Partition: whose CPUs/GPUs are we using? (we'll use stf)

Allocation: which bank account pays for these? (also stf)

Locations: where to write/read files to/from

Resources: how many CPUs/GPUs are needed, and for how long?

Modules: which programs should be loaded to use?



<https://slurm.schedmd.com/>

“Job Scheduler”

Manages multi-node jobs and
communication protocols

} Packaged into a
“Slurm script”

Anatomy of a slurm script (written in “bash”)

In a text file called “submit.slurm”

```
#!/bin/bash
## The first line has to say this, as a bash script

## Job Name
#SBATCH --job-name=test_python

## Partition and Allocation
#SBATCH -p stf
#SBATCH -A stf

## Resources
#SBATCH --nodes=1
#SBATCH --time=0:01:00
#SBATCH --ntasks=1
#SBATCH --mem=100G

## Specify the working directory for this job
#SBATCH --chdir=.

## Import any modules here
module load contrib/anaconda/anaconda4.4.0

## Scripts to be executed here
python test_run.py

## Clean up
exit 0
```

Any line that begins with `##` is a comment and is not read
Each line that begins with `#SBATCH` specifies info for slurm
The `#SBATCH` options can be provided in any order

- Name is optional
- Use stf nodes from the stf budget
- Request (1) node
- Time – Hours : Minutes : Seconds
- A single node has 28 cores and 128 GB memory
 - This dummy job is not parallel, so only (1) core
- Directory “.” is *current location* to read/write files (the same place submit.slurm is saved)
- Load in a pre-installed python package
- The real run command

To execute this job,
`sbatch submit.slurm`

Then We Wait

- `sbatch <script>`
 - Submits a script for non-interactive use
 - Used this to submit a job on previous slide

Want to check the status?

- `squeue`
 - Flag `-u <uwnetid>` for only your jobs
 - Specify `-p` or `-A` for whole queue

What it looks like to check queue:

```
(base) [prelj@mox1 move]$ squeue -p stf
      JOBID PARTITION   NAME     USER ST       TIME  NODES NODELIST(REASON)
      417044      stf test_pyt  prelj CG        0:02      1 n2282
      416180      stf          PD        0:00      1 (Dependency)
      416186      stf          PD        0:00      1 (Dependency)
      413137      stf          R 1-04:27:53  10 n[2143,2151,2166,2168,2171-2172,
      415317      stf          R   15:51:35    6 n[2140,2165,2275,2277,2292,2298]
      414068      stf          R   22:51:51    6 n[2266,2269-2270,2285,2290,2308]
      413181      stf          R 1-04:04:16    6 n[2139,2144,2146,2279,2284,2312]
```

Something wrong? / Taking too long?

- `scancel <jobid>`
 - Cancels an unfinished job
 - Can only cancel your own

Always check your output!

JOBID is used to refer to specific jobs
NODELIST has ID #s for individual nodes being used
ST column is status

R – Running (currently active)

PD – Pending (in queue)

CG – Completing (in process of termination)

Slurm: Commands

- `sbatch <script>`
 - Submits a script for non-interactive use
- `squeue`
 - Get status of jobs in batch queue
- `scancel <jobid>`
 - Cancels an unfinished job

- `srun`
 - Submits job for interactive use or initiate job steps inside batch script
 - (More on interactive jobs in a minute)
- `sinfo`
 - Get state of partitions and nodes (is the system operational)
- `sacct`
 - Gets accounting information about active and completed jobs

**I use these
most often**

Slurm docs and `man <slurm-command>` are very useful!
(“man” for manual)

Running Jobs Interactively (command line, not script)

```
srun -p stf -A stf --ntasks=8 --mem=10G --time=0:10:00 --pty bash -l
```

- Partition
- Account
- Number of processes (*)
- Amount of RAM
- Time
- Command

- Much of the same specifications as submit.slurm
- `--pty bash -l` signifies *interactive*
- Now you have command line control *while* accessing a compute node

Before running interactive job command

```
[(base) [prelj@mox1 jesse]$
```

After running interactive job command

```
[(base) [prelj@n2232 jesse]$
```

- Interactive nodes are computing nodes. You **can** submit jobs to other nodes from interactive nodes.

Be aware of difference between:

- Number of tasks (processes, MPI)
 - `--ntasks (-n)`
 - `--ntasks-per-node`
- Number of cpus per task (threads, OpenMP)
 - `--cpus-per-task (-c)`

<https://slurm.schedmd.com/srun.html>

man srun

Loading software: the modules system

- `module avail`
 - Show all available modules
- `module load <module>`
- `module unload <module>`
 - (Un)load a given module
 - Provide full module name
- `module list`
 - List loaded modules
- `module purge`
 - Unload all modules
- `module help`
 - Print help with commands

Running `module avail`

```
(base) [prelj@mox1 jesse]$ module avail
----- /sw/modules-1.775/modulefiles -----
anaconda2_4.3.1      contrib/multiz/20090121
anaconda2_5.3       contrib/multiz/20190527
anaconda3_4.3.1     contrib/mummer/3.23
anaconda3_5.3       contrib/mummer/4.0.0b2
ansys_18            contrib/muscle/3.8.31
cmake/3.11.2        contrib/mysql/8.0.11
cmake_3.8           contrib/NAMD2019/namd2
contrib/3ddna/170123 contrib/nauty26r11/nauty26r11
contrib/9.0         contrib/ncbi-blast/2.7.1
contrib/abaqus/2020  contrib/ncbi-vdb/2.9.0
contrib/abyss/2.0.3  contrib/nektar
contrib/adaptor-removal/2.1.7 contrib/newton-x
contrib/admixtools/1.0.1 contrib/newtonX
contrib/admixture/1.3.0 contrib/ngs/2.9.0
contrib/anaconda-2019.03 contrib/ngsRelate/1.0
contrib/anaconda/2-5.0.1 contrib/ngsRelate/2.0
contrib/anaconda/3-4.4.0 contrib/nvidia_toolkit/9.0
contrib/anaconda/3-5.3.0 contrib/nvidia_toolkit/10.0
contrib/anaconda/anaconda4.4.0 contrib/NWChem/NWChem
```

... and more

> 400 modules available on Hyak right now!

Useful programs you'd use on your own computer, i.e.

- Python/Anaconda, Gromacs, NAMD, R, Mathematica, Matlab, Gaussian, compilers and more

Parallel computing tools

- CUDA, IMPI, etc.

Advanced user's note:

The modules system works by keeping track of and modifying environment variables (e.g. `PATH`, `LD_LIBRARY_PATH`, `CPATH`, etc.)

<https://modules.readthedocs.io/en/latest/>

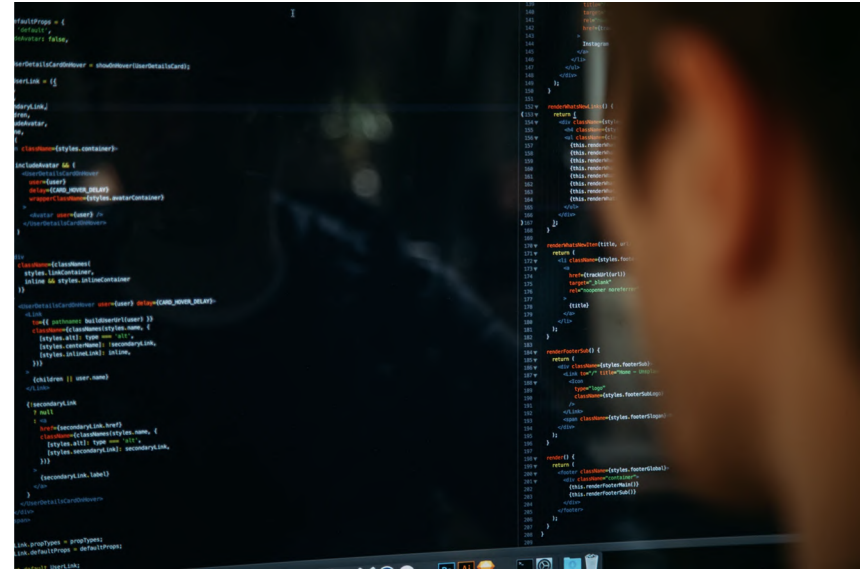
Checkpoint queue – special case

The checkpoint queue allows any user to run on other groups' unused nodes!

- Partition: ckpt
- Account: <group>-ckpt (stf-ckpt)
- Jobs can be interrupted at any time
- Jobs *will* be interrupted after 4 contiguous hours
- Jobs will be resubmitted after interruption (if under total requested time limit)
- **Your code must be checkpointed to take advantage of this**
 - Save a binary file containing state of some objects, restart from logs, etc.
 - Your script should also account for any checkpointing that is done

Topics not covered

- How to install new software
 - Installation and build systems
 - Writing your own modulefiles
 - What software would make good examples?
- How to parallelize your own code
 - Different paradigms for parallelism
 - Data locality and contiguity
 - GPU parallelization
 - Parallel patterns
 - Some programs have parallelization options built-in
- Anything with the cloud (This is the Hyak training session!)
- General architecture of HPC systems
 - Interconnectivity and node locality
 - Physical infrastructure



```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Where to get help

- Documentation (man or webpages)
- Hyak wiki:
<https://wiki.cac.washington.edu/display/hyakusers/WIKI+for+Hyak+users>
- Slack channel: <https://uw-rcc.slack.com/>
- Website: <https://depts.washington.edu/uwrcc/>
- Emails: hpsc@uw.edu or uwrcc@uw.edu
- Office hours: Fridays from 1-3 pm

Happy computing!

