How to Start Using Princeton’s High Performance Computing Systems

Dawn Koffman
Office of Population Research
Princeton University
May 2019
Running Your Computation on Princeton’s High Performance Computing Systems

**significantly** more computing resources available compared to your laptop or desktop

but often, particularly at first, it is much easier to develop and debug locally, **and then**

- connect
- transfer files
- submit jobs

and make use of full computing power available on these remote systems
Workshop Outline

Part I
• Princeton’s High Performance Computing Systems

Part II - Linux
• Philosophy and User Interface
• Files and Directories
• Commands
Princeton’s High Performance Computing Systems

• Overview
• Obtaining Accounts
• Connecting
• Transferring Files
• Running R scripts and Stata .do Files
• Using a Scheduler to Submit Computing Jobs
Part I
Princeton’s High Performance Computing Systems

- remote computing systems managed by Princeton’s Research Computing group
  http://www.princeton.edu/researchcomputing/

- hardware location:
  - High Performance Computing Research Center (HPCRC)
  - 47,000-square-foot facility opened in 2011

- computing systems make up TIGRESS:
  Tera-scale Infrastructure for Groundbreaking Research in Engineering and Science

- in addition to remote computing systems, Research Computing also provides:
  software licenses and support (Stata, SAS, Matlab, ...)
  visualization lab
  office hours
How to Get Started

- request and obtain an account
- connect to the remote system
- transfer files (programming scripts, data, output)
- interact with remote system’s operating system (Linux)
- execute computational jobs, often using a resource manager/scheduler
two TIGRESS systems are available simply by registering on a webpage:

nobel
- load-balanced cluster of interactive computational Linux systems
- two Dell R610 servers named for Princeton Nobel Laureates, Compton and Davisson
- good entry point for researchers and students
- well-suited for:
  - access to commercially licensed software provided centrally
  - lower end computational tasks
  - teaching and coursework

adroit
- 8 node cluster, adroit-01 through adroit-08
- 160 processors available, twenty per node
- each node contains 64 GB memory
- intended for developing small production jobs
- all jobs other than those that last for just a few minutes must be run through a scheduler

To register for an account on these systems, go to:

https://researchcomputing.princeton.edu/access
Other TIGRESS Systems

**della** and **tiger**
- large general purpose clusters with hundreds of nodes
- della performs well for most parallel processing jobs and for users with large numbers of serial jobs
- tiger is used by the largest, most demanding parallel jobs

**perseus**
- particularly well suited to large, computationally intensive parallel jobs because of its relatively large number of cores/node that all include the latest vector processing units

Access to these large clusters (della, tiger and perseus) is granted on the basis of brief faculty-sponsored proposals
For Prospective Users - Proposals

Proposals for the large cluster systems should be submitted as PDF or MS Word documents not to exceed 3 pages. Proposals can be submitted through an online form via https://researchcomputing.princeton.edu/access#proposals and should include:

• Which system or systems you need to use
• A list of researchers who will need accounts
• The faculty member(s) who is sponsoring the project
• The scientific background for your project including scientific merit of the proposed work
• The programming approach for your project:
  • Programming language
  • Parallelization mechanism (MPI or OpenMP)
  • Required libraries
• The resource requirements for your project:
  • Number of concurrent cpus
  • Total cpu time
  • RAM per task
  • Total disk space
• A few references or citations
Connecting to Remote System

**SSH (secure shell)**
- provides access to remote systems over a network
- already installed on Mac OS X and Linux
- for Windows, can use ssh implementation at [http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html](http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html)
  - look for “Windows installer for everything except PuTTYtel”
  - download installer and run it

connecting to remote system from a Windows system
- start a new PuTTY terminal session by finding PuTTY in “Start Menu”
- create a new connection
  hostname: `adroit.princeton.edu` or `nobel.princeton.edu`
  username: Princeton netid
- if successful: will see terminal window with machine name, user name and prompt

connecting to remote system from a Mac
- open a terminal application window
- at the prompt (usually $ or %), enter
  `ssh nobel.princeton.edu` or
  `ssh adroit.princeton.edu`
- if successful, will see terminal window with machine name, user name and prompt
Transferring Files

between local system and remote system, need to transfer:
- code (R scripts, Stata .do files, etc)
- data
- output

between local **Windows** system and remote Linux system:
- FileZilla
  - graphical file transfer program
  - open-source: [https://filezilla-project.org/](https://filezilla-project.org/)
- PSCP and PSFTP
  - command line tools from PuTTY

between local **Mac** system and remote Linux system:
- rsync
  - command line tool
  - already installed
- scp
  - command line tool
  - already installed
- FileZilla
  - graphical file transfer program
  - open source: [https://filezilla-project.org/](https://filezilla-project.org/)
Transferring Files

between local **Windows** system and remote Linux system -
using command line tool PSFTP from PuTTY

Select PSFTP from Start Menu

```bash
psftp: no hostname specified; use "open host.name" to connect
psftp> open adroit.princeton.edu
login as: dkoffman
dkoffman@adroit.princeton.edu’s password: mypassword1234
Remote working directory is /n/homeserver/user/dkoffman
psftp> help
cd change your remote working directory
lcd change your local working directory
pwd print your remote working directory
lpwd print your local working directory
get download a file from the server to your local machine
put upload a file from your local machine to the server
exit finish your SFTP session
psftp> put hello.R
psftp> put datafile.csv
psftp> exit
```
### Transferring Files

between local **Mac** system and remote Linux system - using **rsync** command **from terminal window**

<table>
<thead>
<tr>
<th>SOURCE</th>
<th>DESTINATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ rsync ~/hello.R</td>
<td><a href="mailto:dkoffman@adroit.princeton.edu">dkoffman@adroit.princeton.edu</a>:~/hello.R</td>
</tr>
<tr>
<td><a href="mailto:dkoffman@adroit.princeton.edu">dkoffman@adroit.princeton.edu</a>’s password:</td>
<td></td>
</tr>
<tr>
<td>$</td>
<td></td>
</tr>
</tbody>
</table>

- **rsync**: fast and flexible; many options can be used to adjust its behavior, for example:
  - `-r` recurse through sub-directories
  - `-v` verbosely print messages
  - `-z` compress data before transfer and decompress after transfer

- for (many) more options, see manual page (**man rsync**)
- often put **rsync** command with appropriate options within a shell script on local machine
Transferring Files

between local **Mac** system and remote Linux system - using **scp** command **from terminal window**

<table>
<thead>
<tr>
<th>SOURCE</th>
<th>DESTINATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ scp <del>/hello.R <a href="mailto:dkoffman@adroit.princeton.edu">dkoffman@adroit.princeton.edu</a>:</del>/hello.R</td>
<td></td>
</tr>
<tr>
<td><a href="mailto:dkoffman@adroit.princeton.edu">dkoffman@adroit.princeton.edu</a>’s password:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
- differences between **rsync** and **scp**:

  - scp is more basic: regular, linear copy; fewer options for tweaking its behavior

  - rsync uses a delta transfer algorithm and some optimizations to make copying faster
    - for example, if destination file already exists, rsync will check file sizes and modification timestamps and skip further processing if both of those match
Running Stata or R Script Without a Scheduler

$ stata -b do hello.do

contents of results window sent to text file hello.log

$ Rscript -e 'a <- 5' -e 'a' > show_a.txt

$ Rscript hello.R > hello.txt

To run xstata on nobel, install:

https://sourceforge.net/projects/xming/ ... for Windows users

https://www.xquartz.org/ ... for Mac users

- Restart local machine
- Before connecting to nobel, run xming or xquartz
- Once on nobel, execute: xstata &
Submitting Computing Jobs to the Clusters

**SLURM (Simple Linux Utility for Resource Management)**

https://slurm.schedmd.com/sbatch.html

- cluster management and job scheduling system for large and small Linux clusters

- submitting a job is similar to taking a ticket at a deli counter
- once machine is ready to run a job, it comes out of the queue

- submitting a job requires using specific commands in a specific format that tell scheduler (1) what resources you are requesting:
  
  # CPU’s
  # of nodes
  GB memory
  how much time

  and (2) what commands you would like to have run

- commands will run when resources are available

- scheduler assigns hardware based on your requests
Submitting a Serial Job

- create a job script for SLURM, here named `serial_ex1.cmd`

```bash
$ cat serial_ex1.cmd
#!/usr/bin/env bash

#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH -t 10:00

Rscript -e 'rnorm(1e3)'
```

- submit the job to the batch system (queue)

```bash
$ sbatch serial_ex1.cmd
submitted batch job 220
$ ls *.out
slurm-220.out
$
Submitting a Serial Job

- create a job script for SLURM, here named serial_ex2.cmd

```bash
$ cat serial_ex2.cmd
#!/usr/bin/env bash

#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH -t 10:00
#SBATCH -o log.%j
#SBATCH –mail-type=begin
#SBATCH –mail-type=end

Rscript -e 'rnorm(1e3)'
```

- submit the job to the batch system (queue)

```bash
$ sbatch serial_ex2.cmd
submitted batch job 194717
$ ls log.*
log.194717
$
Email from SLURM

From: SLURM User [mailto:cses@princeton.edu]
Sent: Friday, May 01, 2015 2:45 PM
To: Dawn A. Koffman
Subject: SLURM Job_id=194717 Name=serial_ex2.cmd Began, Queued time 00:00:00

From: SLURM User [mailto:cses@princeton.edu]
Sent: Friday, May 01, 2015 2:46 PM
To: Dawn A. Koffman
Subject: SLURM Job_id=194717 Name=serial_ex2.cmd Ended, Run time 00:00:00

Job ID: 194717
Cluster: adroit
User/Group: dkoffman/pustaff
State: COMPLETED (exit code 0)
Cores: 1
CPU Utilized: 00:00:00
CPU Efficiency: 0.00% of 00:00:00 core-walltime
Memory Utilized: 1.52 MB
Memory Efficiency: 0.05% of 3.12 GB
Submitting a Serial Job

- create a job script for SLURM, here named serial_ex3.cmd

```
#!/usr/bin/env bash

#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH -t 10:00
#SBATCH -o log.%j
#SBATCH -m mail-type=begin
#SBATCH -m mail-type=end

Rscript hello.R
```

- submit the job to the batch system (queue)

```
$ sbatch serial_ex3.cmd
submitted batch job 194718
$ ls log.*
log.194718
$
SLURM Commands

sbatch - submit job

scancel  jobid  - cancel a running/submitted job

squeue  - display information about the jobs in the queue
  (jobid, name, user, time, nodes, nodelist (reason))

squeue –u userid  - display information about jobs in the queue for user = userid

srun  - submit a job for parallel execution

many other SLURM commands
see manual page (man sbatch) to see 1000’s of options
Case Study: Replicating Results


https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/HHAUDF

Sociological Methodology

Generalizing the Network Scale-Up Method:
A New Estimator for the Size of Hidden Populations

Dennis M. Feehan, Matthew J. Salganik

First Published September 20, 2016
Case Study: Issue #1

> From: Dawn Koffman
> Sent: Tuesday, March 08, 2016 10:59 AM
> To: Computational Science and Engineering Support
> Subject: RE: [CSES #11965] account on nobel

> ... I’m having trouble installing the R package "devtools“ ...

Should be ready for you to try again. openssl-devel has been installed.

How to get help:
- Send email to cses@princeton.edu
- Attend CSES office hours
Case Study: Issue #2

I'm now trying to run an R script on Nobel that produces approximately 200GB of data, and I'm getting a write error: Disk quota exceeded.

Would it be possible for my disk quota to be increased to that level?

We have no space on Nobel that size. Your home directory quota could be increased, but that's a question you must pose to the SOC (helpdesk@princeton.edu) as we do not have access to modify it. There is a fee involved I believe, so you may need to come up with other plans for data management.
> Is there another HPC linux machine that would have space that I
> (or a faculty member) might be able to use for this? Thanks again.

As of right now, there's 290GB free on /scratch/network on Adroit,
which requires only a registration to get access. For other clusters,
information on getting an account is available here:
https://askrc.princeton.edu/question/23/how-do-i-get-an-account-on-a-tigress-system/
Dear Dawn A. Koffman,

The Computational Science and Engineering Support group (CSES) has received your email or created a ticket for you regarding "installing RcppArmadillo on adroit" with the following content:

I have an account on adroilt and am trying to install R package RcppArmadillo there but am getting many compilation errors. Could someone please look into this and back to me?
Case Study: Issue #3

> I have an account on adroit and am trying to install R package RcppArmadillo there but am getting many compilation errors. Could someone please look into this and back to me?

Hi,

it looks like this package wants newer c++ compiler. Therefore run this first:

module load rh

and then proceed to install RcppArmadillo.
#!/usr/bin/env bash

#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH -o log.%j
#SBATCH --mail-type=begin
#SBATCH --mail-type=end
#SBATCH --mail-user=dkoffman@princeton.edu

cd /scratch/network/dkoffman/gnsum-paper-torelease/code
Rscript gnsum-draw-popns.R
Case Study: Issue #4

Emails from SLURM:

Subject: SLURM Job_id=328797 Name=serial_1.cmd Began, Queued time 00:00:00

Subject: SLURM Job_id=328797 Name=serial_1.cmd Ended, Run time 00:01:02

Job ID: 328797
Cluster: adroit
User/Group: dkoffman/pustaff
State: CANCELLED (exit code 0)
Cores: 1
CPU Utilized: 00:00:54
CPU Efficiency: 87.10% of 00:01:02 core-walltime Memory Utilized: 3.78 GB
Memory Efficiency: 120.85% of 3.12 GB
Case Study: Issue #4

> I'm trying to run an R script on Adroit, but the log file shows that my job exceeded the memory limit and was cancelled:
> slurmstepd: Job 328797 exceeded memory limit (3960080 > 3276800),
> being killed
> slurmstepd: Exceeded job memory limit
> slurmstepd: *** JOB 328797 ON adroit-07 CANCELLED AT 2016-04-12T11:02:16 ***
> slurmstepd: Exceeded step memory limit at some point.

Please see this for a solution

This error means that you will have to allocate more memory for your jobs - with slurm we allocate CPUs separately from memory. You didn't say which cluster so default will be either 3GB or 4GB per CPU core (you can see which by looking at /etc/slurm/slurm.conf - at the end are DefMem... settings). You can try increasing it by doing something like:

```
#SBATCH --mem-per-cpu=6000
```

It would be wise not to overreact by requesting too much memory - if you do that not only will you waste resources that others could've used but also you will take longer to run as the scheduler will have more trouble finding free nodes with that much memory. Ideally you should request just what you need or a bit more.

Case Study: Issue #4
Case Study: Issue #2 - Again

> Over the last several days, I've been trying to run an R script on adroit that aims to generate about 230GB of data. So far the script has been failing due to it exceeding the memory limit, so I am continuing to increase the memory limit using: SBATCH --mem-per-cpu directive.

> I am concerned though that even if the program does not exceed its memory limit, that it will fail to complete due to disk space constraints.
> Currently I am using disk space under scratch/network/dkoffman.

Looks like folks have cleaned up and there is now 450G available there.
Case Study: Issue #2 - Again

The /scratch/network filesystem is currently too full. Please remove/move files you have there to someplace else so that this shared resource can be used by all.

This filesystem is NEVER backed up so anything sitting there is at-risk.

What follows is the ordered size of what users currently have in /scratch/network in KBytes.

281031500 abc13
258879332 defgh
112202456 jessie
91466620 xyz123
72147092 smith
66059628 alison456
48538600 dkoffman
46238316 steve98989
41328828 joe45678
...
Case Study: Issue #5

Emails from SLURM:

Subject: SLURM Job_id=330033 Name=serial_1.cmd Began, Queued time 04:10:24

Subject: SLURM Job_id=330033 Name=serial_1.cmd Ended, Run time 20:05:02

Job ID: 330033
Cluster: adroit
User/Group: dkoffman/pustaff
State: TIMEOUT (exit code 1)
Cores: 1
CPU Utilized: 19:00:53
CPU Efficiency: 94.68% of 20:05:02 core-walltime
Memory Utilized: 49.18 GB
Memory Efficiency: 83.94% of 58.59 GB
Case Study: Final SLURM Script

#!/usr/bin/env bash

#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=60000
#SBATCH -t 30:00:00
#SBATCH -o log.%j
#SBATCH --mail-type=begin
#SBATCH --mail-type=end
#SBATCH --mail-user=dkoffman@princeton.edu

cd /scratch/network/dkoffman/gnsum-paper-torelease/code
Rscript gnsum-draw-popns.R
Case Study: Success!

Subject: SLURM Job_id=331265 Name=serial_1.cmd Began, Queued time 00:04:02

Job ID: 331265
Cluster: adroit
User/Group: dkoffman/pustaff
State: COMPLETED (exit code 0)
Cores: 1
CPU Utilized: 22:01:55
CPU Efficiency: 94.09% of 23:24:59 core-walltime
Memory Utilized: 47.53 GB
Memory Efficiency: 81.11% of 58.59 GB

Subject: SLURM Job_id=331265 Name=serial_1.cmd Ended, Run time 23:24:59
Finally!
I initially ran into issues with:
  - disk space
  - memory
  - time
and I also needed:
  - to have openssl-devel installed
  - a newer C++ compiler
BUT, I think this first R script has finally run as expected.
Part II

Linux User Interface and Philosophy

• Operating Systems
• Command-Line Interface
• Shell
• Linux Philosophy
• Command Execution Cycle
• Command History
User Interfaces

- **User** (human)
- **Operating System** (software)
- **Computer** (hardware)

**User Interfaces**

- **Command-line Interface**
- **Point-and-Click (GUI) Interface**

**Operating System**

- **Kernel** (manages computing resources: memory, hard-drive, time)
- **Programs** (text editors, compilers, commands for working with file system, many other utilities)

**Computer**

- CPU
- Memory
- Hard-drive
<table>
<thead>
<tr>
<th>command-line interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>- may have steeper learning curve, BUT provides constructs that can make many tasks</td>
</tr>
<tr>
<td>very easy</td>
</tr>
<tr>
<td>- scales up very well when have lots of: data, programs, tasks to accomplish</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>point-and-click interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>- may be more intuitive, BUT can also be much more human-manual-labor intensive</td>
</tr>
<tr>
<td>- often does not scale up well when have lots of: data, programs, tasks to accomplish</td>
</tr>
</tbody>
</table>
Shell

Command-line interface provided by Linux is called a shell:
- prompts user for commands
- interprets user commands
- passes them on to the rest of the operating system which is hidden from the user

How do you access a shell?
- if you have an account on a machine running Linux, just log in. A default shell will be running.
- if you are using a Mac, run the Terminal app. A default shell will be running.

-if Terminal app does not appear on the Shortcut Bar: Go -> Applications -> Utilities -> Terminal
- Even though there are differences between the various Linux operating systems, for the most part, we are going to ignore those differences, and just refer to Linux operating systems because the principles are largely the same.

- Different versions of Linux shells (more alike than different):
  - `sh` (original Unix shell, by Stephen Bourne)  `/bin/sh`
  - `ksh` (similar to sh, by David Korn)  `/bin/ksh`
  - `bash` (Bourne again shell, part of GNU project)  `/bin/bash`
  - `csh` (part of Berkely Unix, intended to be C-like, by Bill Joy)  `/bin/csh`
  - `tcsh` (based on and compatible with csh)  `/bin/tcsh`

```bash
echo $SHELL
```
Linux Philosophy

- provide **small** programs that do **one thing well**
  and
  provide mechanisms for **joining programs together**

- “silence is golden”
  when a program has nothing to say, it shouldn’t say anything

- users are very intelligent and **do what they intend to do**

  will not find **OK** or **Cancel** in Linux environment!
Examples of Tasks for Command-Line Interface

data management:
   - two types of administrative data – millions of observations of each type
   - need to standardize addresses for merging (or fuzzy matching)

file management
   - check number of lines in large file downloaded from the web

file management:
   - split huge files into subsets that are small enough to be read into memory

basic web scraping
   - list of UN countries and dates they became members
# Command Execution Cycle and Command Format

1. Shell prompts user
2. User inputs or recalls command ending with <CR>
3. Shell executes command

```bash
$ date
$ who
$ who  -q
$ cal 2017
$ cal 8 2017
$ pwd
$ ls
$ mkdir linux
$ cd linux
$ pwd
$ ls
```

### Command Format

- **Command**
  - first word on line
  - name of program

- **Options**
  - usually begin with `-`
  - modify command behavior

- **Arguments**
  - “object” to be “acted on” by command
  - often directory name, file name, or character string

use `man command` for options & arguments of each command

use `PS1="$ "` to change prompt string
Using Command History

commands are saved and are available to recall

to re-execute a previously entered command:

step 1. press ↑ to scroll through previously entered commands
step 2. press <CR> to execute a recalled command

OR

to re-execute a previously entered command:

$ history
$ !<command number>
Files

• Displaying File Contents
• File Management Commands
• File Access and Permission
• Redirecting Standard Output to a File
• File Name Generation Characters
Files

file names:
- should not contain spaces or slashes
- should not start with + or –
- best to avoid special characters other than _ and .
- files with names that start with . will not appear in output of ls command

created by:
- copying an existing file
- using output redirection
- executing some Linux program or other application
- using a text editor
- downloading from the internet

$ pwd
/u/dkoffman/linux

$ wget https://opr.princeton.edu/workshops/Downloads/2015May_LinuxTourKoffman.gz
# NOT standard on OS X

**** OR ****

$ curl https://opr.princeton.edu/workshops/Downloads/2015May_LinuxTourKoffman.gz -o wdata.gz
# available on OS X

$ gunzip wdata.gz

$ ls
Displaying File Contents

$ wc wdata

$ cat wdata

$ head wdata

$ head -1 wdata

$ tail wdata

$ tail -2 wdata

$ more wdata
File Commands

$ cp wdata wdata.old

$ mv wdata.old wdata.save

$ cp wdata wdata_orig

$ cp wdata wdata_fromweb

$ rm wdata_orig wdata_fromweb

$ diff wdata wdata.save
File Access and Permission

\$ ls -l

-rw------ 1 dkoffman rpcuser 8586 Apr 1 14:46 wdata
-rw------- 1 dkoffman rpcuser 8586 Apr 1 14:27 wdata.save

User classes

- owner (user) = u
- group = g
- other = o
- all = a

File permissions

- read (cat, tail, cp, ...) = r
- write (vi, emacs) = w
- execute (run program) = x

\$ chmod g+w wdata
\$ ls -l
\$ chmod go-r wdata.save
\$ ls -l
Redirecting Standard Output

most commands display output on terminal screen

```
$ date
```

call your command output can be redirected to a file

```
$date > date.save
$ cat date.save
```

*** note: output redirection using > overwrites file contents if file already exists

```
$date > date.save
$ cat date.save
```

use `>>` to append output to any existing file contents (rather than overwrite file)

```
$date >> date.save
$ cat date.save
```
File Name Generation Characters

shell can automatically put file names on a command line if user uses file name generation characters

? any single character
$ cat s?

* any number of any characters (including 0)
$ ls b*
$ ls *.R
$ wc -l *.do
$ ls *.dta
$ ls check_*.*

[...] any one of a group of characters
$ rm s[4-7]
Directories

- Directory Tree
- Pathnames: Absolute and Relative
- Copying, Moving and Removing Files & Directories
pwd  shows you where you are (present working directory)

cd  makes your “home” (login) directory your current directory

Directory Tree

/  
  /bin /etc /usr /u /tmp /dev  
    /who /date /passwd /bin /lbin /dkoffman /awest  
      /diff /curl /emacs /linux  
         /date.save /wdata /wdata.save
Changing Directory

absolute pathnames

$ pwd
$ cd /etc
$ cat passwd
$ cd /bin
$ ls e*
$ ls f*
$ cd /usr/bin
$ ls e*
$ ls f*
$ cd /u/dkoffman
$ cd /u/dkoffman/linux

relative pathnames

$ pwd
$ cd ../../../etc
$ cat passwd
$ cd ../bin
$ ls e*
$ ls f*
$ cd ../usr/bin
$ ls e*
$ ls f*
$ cd
$ cd linux

.. refers to the parent directory
## Accessing Files

### absolute pathnames

$ pwd

$ cat /etc/passwd

$ ls /bin/e*

$ ls /bin/f*

$ ls /usr/bin/e*

$ ls f*

$ pwd

### relative pathnames

$ pwd

$ cat ../../../etc/passwd

$ ls ../../../bin/e*

$ ls ../../../bin/f*

$ ls ../../../usr/bin/e*

$ ls ../../../usr/bin/f*

$ pwd

.. refers to the parent directory
$ cp date.save  date.save2  
$ mkdir savedir  
$ cp *.save* savedir  
    list of files  
$ cd savedir  
$ ls  
$ cp date.save2 date.save3  
$ cp date.save3 ..  
$ ls ..  
$ cp date.save2 date.save4  
$ cd ..  
$ cp savedir/date.save4 .  

. refers to the current directory
Moving Files

$ cp date.save date4move
$ mv date4move date.4move
$ ls
$ mkdir movedir
$ mv date.4move movedir
$ ls
$ ls movedir
$ mv date.save[23] movedir

list of files
$ ls
$ cd movedir
$ ls
$ mv ../date.save .
$ ls
$ cd ..
Removing Files and Directories

```bash
$ cd
$ cd linux
$ rm date.save4 wdata.save
$ rmdir movedir
  rmdir: failed to remove 'movedir':
  Directory not empty
$ ls movedir
$ rm movedir/*  # BE CAREFUL!
$ rmdir movedir
$ rm savedir/date*
$ ls savedir
$ ls
```
Commands

• Review of Commands
• More Commands
• Sequential Execution
• Command Grouping
• Pipelines
• Foreground/Background Command Execution
# Review of Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>gunzip</td>
</tr>
<tr>
<td>who</td>
<td>cat</td>
</tr>
<tr>
<td>cal</td>
<td>head</td>
</tr>
<tr>
<td>pwd</td>
<td>tail</td>
</tr>
<tr>
<td>ls</td>
<td>more</td>
</tr>
<tr>
<td>mkdir</td>
<td>cp</td>
</tr>
<tr>
<td>cd</td>
<td>mv</td>
</tr>
<tr>
<td>history</td>
<td>rm</td>
</tr>
<tr>
<td>curl</td>
<td>diff</td>
</tr>
<tr>
<td>wget</td>
<td>chmod</td>
</tr>
<tr>
<td></td>
<td>rmdir</td>
</tr>
</tbody>
</table>
More Commands

$ tail -40 wdata

$ sort wdata

$ tail -40 wdata

$ sort wdata > wdata.sort

$ more wdata.sort

$ sort -r wdata > wdata.revsort

$ more wdata.revsort

$ wc wdata

$ wc -l wdata

$ wc -wc wdata
$ head wdata

$ cut -d"","" -f1 wdata

$ head wdata

$ cut -d"","" -f1 wdata > wdata.countries

$ cut -c1,3-4 wdata

$ cut -d"","" -f5 wdata > wdata.le

$ paste wdata.le wdata.countries

$ sort wdata.le > wdata.le.sort

$ uniq wdata.le.sort

$ uniq -c wdata.le.sort
More Commands

`$ grep ",Oceania,\" wdata`

`$ grep ",Central America,\" wdata > wdata.centralamerica`

`$ grep pop2012 wdata`

`$ grep pop2012 wdata > wdata.hd`

`$ grep -v pop2012 wdata > wdata.clean`

`$ head wdata.clean`

`$ wc -l wdata.clean`

`$ grep -n ",Oceania,\" wdata.clean`

`$ grep -n -i ",oceania,\" wdata.clean`
Regular Expressions

describe a sequence of characters (pattern) to be matched

basics

. (dot) matches any single character: 1.6

[ ] (brackets) match any one of the enclosed characters: [aeiou]
can use – (dash) to indicate at range of characters: [A-Za-z] [24-6]

[^ ] match any character except the enclosed characters:[^Zz]

* (asterisk) matches zero or more of the preceding character: b* vs bb*

^ (caret) pattern must occur at the beginning of a line (anchor): ^ABC

$ (dollar sign) pattern must occur at the end of a line (anchor): ABC$ vs ^ABC$

\ (backslash) turns off (escapes) the special meaning of the next character: \.*

enclose regular expressions in single quotes to stop shell from expanding special characters
Using Regular Expressions

$ grep stan wdata.clean

$ grep '^B' wdata.clean

$ grep '^....,' wdata.clean

$ grep '/ ' wdata.clean

$ grep -i ira[qn] wdata.clean

$ grep '^.*,.*West' wdata.clean

$ grep '4.,[A-Z]' wdata.clean

$ grep '[56]..,[A-Z]' wdata.clean

$ grep '[67]..,[A-Z]..*Americas' wdata.clean
More Commands

$ split -l20 wdata.clean

$ ls

$ wc -l xa?

$ tail xah

$ cat xa? > wdata.clean.copy

$ wc -l wdata.clean.copy

$ tr "abcdefghijklmnopqrstuvwxyz" "ABCDEFGHIJKLMNOPQRSTUVWXYZ" < wdata

$ tr [:lower:] [:upper:] < wdata.clean > wdata.clean.uc

$ tr -d `:`` < wdata.clean

$ tr -s `` < wdata.clean
Sequential Execution

cmd1 arg1 arg2 ...; cmd2 arg1 arg2 ...; cmd3 arg1 arg2 ...

- series of commands on a single line separated by semicolons

- commands are executed left-to-right, one at a time

$ sort wdata.clean > wdata.clean.s; echo SORT DONE
Command Grouping

(cmd1 arg1 agg2 ...; cmd2 arg1 arg2 ...; cmd3 arg1 arg2 ...)

- allows several commands to be treated as one with respect to standard output

$ date > log

$ who am i >> log

$(
  > date
  > who am i
  > ) > log

$(date; who am i) > log
Pipeline

cmd1 arg1 ... | cmd2 arg1 ... | cmd3 arg1 ...

- series of commands separated by |

- output of one command used as input for next command

- commands run in parallel when possible!

- avoids use of temporary files ... faster!

$ who | sort

$ who > tempfile
$ sort < tempfile
$ rm tempfile
Pipeline Examples

$ who | wc -l

$ ls -l | grep "^d"

$ grep Africa wdata.clean | sort

$ sort wdata.le | uniq | wc -l

$ grep Americas wdata.clean | cut -d"","" -f5 | sort

$ grep Americas wdata.clean | cut -d"","" -f5 | sort | uniq

$ grep Americas wdata.clean | cut -d"","" -f5 | sort | uniq | wc -l

$ sort wdata.clean | tr [:lower:] [:upper:] | cut -d"","" -f1

$ sort wdata.clean | cut -d"","" -f1,5

$ sort wdata.clean | cut -d"","" -f1,5 | tr -d ".:.' | split -120 - wdata_le_part_
Writing to a File And to Standard Output

tee command
- reads from standard input
- writes to a file and standard output
- very useful for saving intermediate “results” in a pipeline
- use –a option to append to a file rather than overwrite

$ sort wdata.le | uniq | tee wdata.le.uniq | wc -l

$ cat wdata.le.uniq

$ sort wdata.le | uniq | tee wdata.le.uniq | wc -l > le.uniq.count

$ cat le.uniq.count

$ sort wdata.clean | cut -d"," -f1,5 | tee c.le | split -l20 - wdata_le_part_

$ cat c.le
Foreground and Background Command Processing

Foreground command processing

- one command line must complete execution before next command line begins execution
- “normal” way commands are processed

Background command processing

- next command line begins execution before background command completes
- any standard output is usually redirected to a file
- <BRK> and <DEL> are ignored
- identification number is displayed after background command is entered ... process id
- can stop a command running in the background using the kill command and the process id

$ command arg1 arg2 > outfile &
  10411
  execute command in the background
$ kill 10411
$
Background Command Processing

- normally, a hang-up signal (logging off) is not ignored by a command executing in the background, and will cause it to terminate

- `nohup` prefix allows a command to continue running even if a hang-up signal is received

$ nohup cmd arg1 arg2 ... &

- to check to see if a background command is still running and to obtain its process id, use `ps` command
The End!