More Advanced Shell

🐢

More Advanced Shell 🐢

🐢 grep, sed & regular expressions

🐢 awk

🐢 pipes and bash scripting

🐢 moving and synchronizing files between local and remote hosts (scp, sftp, rsync)

grep:
global regular expression print

🐢 grep searches input files for a search string and prints the lines that match it

grep reads through your file from the beginning,
copies a line into a buffer,
compares it against the search string,
and if the comparison passes,
prints the line to the screen

regex:
regular expressions

🐢 A compact way of describing complex patterns in text

🐢 Using regular expressions is the most efficient way to manipulate files in a text editor (*e.g.* BBedit or Text Wrangler) or in the shell

Examples:

display lines matching a pattern
search and replace (recursively)
file renaming
count matches in a file

regex:
special characters

[...]	a set of possible character matches
\	gives special meaning to the character following it
^	matches the beginning of a line or string
$	matches the end of a line or string
.	matches any character except a newline
|	separates alternate possibilities
?	matches the preceding pattern element zero or one times
*	matches the preceding pattern element zero or more times
+	matches the preceding pattern element one or more times
(...)	groups a series of pattern elements to a single element

regex:
special characters

\r	carriage return
\n	line feed character
\r\n	line separator in Windows
\t	tab

escaping special characters: \ before special character

regex:

an example

check out: www.rubular.com

```
>gi|310757913|gb|CP002287.1| Achromobacter xylosoxidans A8
>gi|283951607|gb|CP001859.1| Acidaminococcus fermentans DSM 20731
>gi|350278064|gb|CP003058.1| Acidaminococcus intestini RyC-MR95
>gi|332693428|gb|CP002535.1| Acidianus hospitalis W1
>gi|302328214|gb|CP001742.1| Acidilobus saccharovorans 345-15
```

```
regex: gi\|.*\.1\|
```

find gi| and include any character until you find .1|
replace with nothing

```
>Achromobacter xylosoxidans A8
>Acidaminococcus fermentans DSM 20731
>Acidaminococcus intestini RyC-MR95
>Acidianus hospitalis W1
>Acidilobus saccharovorans 345-15
```

sed = stream editor 🐢

🐢 sed works with streams of characters line by line

🐢 can get through huge files because it doesn’t hold them in memory

sed = stream editor

- sed can be used at the command-line:

```
$sed 's/old/new/'
```
(replace first occurrence of 'old' with 'new'- at each line)

```
$sed 's/old/new/g'
```
(replace all occurrences of 'old' with 'new' -'g' is for global)

- sed can also be used within a shell script (later!)
- output is written to stdout by default

  can instead redirect this to a new file

  or edit the existing file in place using '-i' flag

awk

- 🐢 awk: *Aho, *Weinberger, *Kernighan
- 🐢 compared to sed, awk is oriented toward delimited fields on a per-line basis

  Think of a table: lines are rows, each line has columns (space or tab as separators)

- 🐢 both awk and sed are tools that transform text, but awk is a programming language by itself with arrays, loops, etc.

awk: examples

```
xylosoxidans fermentans intestini hospitalis saccharovorans

$awk ‘{print $0}’
```

print whole line

```
xylosoxidans fermentans intestini hospitalis saccharovorans

$awk '{print $3}'
```

print 3rd column

```
intestini
```

Pipes and bash scripting

🐢

Redirecting

> (redirect operator) creates a file
>> (append operator) appends to the file
>& redirects stderr and stdout to a file

great for creating log files

```
/apps/allpaths-lg-old/bin/RunAllPathsLG PRE=/pool/genomics
DATA_SUBDIR=M_zebra REFERENCE_NAME=fish RUN=run1 THREADS=24
OVERWRITE=True > fish_5_19.log
```

Using pipes

🐢 A pipe is a form of redirection that sends the output of one program to another program for further processing

🐢 Redirection is the transferring of standard output to some other destination, such as another program, a file or a printer, instead of stdout

🐢 Pipes create temporary direct connections between two or more simple programs

```
command_1 | command_2 [| command_3 . . . ]
```

Using pipes

simple example: pipe the standard output of one program to the standard input of another program

```
ls -l | grep ‘rwx’
```

you will see all of the files in the working directory whose permissions (or name) contain the letters rwx in order

🐢 ‘ls’ lists files to its standard output

🐢 ‘grep’ takes its input and sends any lines that match a particular pattern to its standard output

🐢 the pipe operator (|), which tells the shell to connect the standard output of ls to the standard input of grep

bash scripting

Best practices:

🌻 be modular
🌻 isolate things logically using variables
🌻 make readable
🌻 good choice of names
🌻 use layout like spaces, indentation and width

bash scripting

- `$chmod +x` to make script executable
    - how to run a script: `$./script`

      `$bash <script>` also works

bash scripting

- #!/bin/bash

#!   informs the operating system that it is providing the name and location of the shell (or other scripting language)

/bin/bash   the shell is bash, which is located in the /bin directory

try:

```
$which bash
```

bash scripting

```
#!/bin/bash
lastz target query1 –options > output1.sam
lastz target query2 –options > output2.sam
lastz target query3 –options > output3.sam
lastz target query4 –options > output4.sam
lastz target query5 –options > output5.sam
lastz target query6 –options > output6.sam
```

bash scripting

1 Every new line is a new command unless you use \
2 Comment lines start with a #
3 Commands are surrounded by ()

bash scripting

```
/lustre/medusa/proj/SI/allpathslg-48268_mod_PBlood/bin/RunAllPathsLG \
PRE=/lustre/medusa/proj/SI/aupadhy_assembly/fish/PRE \
REFERENCE_NAME=REFERENCE \
DATA_SUBDIR=attempt_1 \
RUN=run_1 \
THREADS=32 \
OVERWRITE=True
```

bash scripting

```
#!/bin/csh
#
#$ -cwd -j y
#$ -N trinity-hm -o trinity-hm.log
#$ -q himem.q
#$ -l hm,mem_reserved=126G,h_data=126G,h_vmem=130G
#
setenv PATH ${PATH}:/pool/genomics/lattice/samtools-0.1.19/
setenv PATH ${PATH}:/pool/genomics/lattice/bowtie-0.12.9/
setenv PATH ${PATH}:/pool/genomics/lattice/RSEM/

# Place your trinity command line here.
# Note - Use absolute (full) paths for input files.
/pool/genomics/lattice/trinityrnaseq_r20131110/Trinity.pl --seqType fq --JM
450G --single /pool/genomics/savagea/PGF_all_reads.fastq --CPU 40
```

```
#!/bin/sh

#
# Downloads sequence for H. sapiens (human) from NCBI.  This script was
# used to build the Bowtie index for H. sapiens.
#
# From README_CURRENT_BUILD:
#  Organism: Homo sapiens (human)
#  NCBI Build Number: 36
#  Version: 3
#  Release date: 24 March 2008
#

GENOMES_MIRROR=ftp://ftp.ncbi.nih.gov/genomes
FILE_PATH=${GENOMES_MIRROR}/H_sapiens/ARCHIVE/BUILD.36.3/Assembled_chromosomes

BOWTIE_BUILD_EXE=./bowtie-build
if [ ! -x "$BOWTIE_BUILD_EXE" ] ; then
      if ! which bowtie-build ; then
            echo "Could not find bowtie-build in current directory or in PATH"
            exit 1
      else
            BOWTIE_BUILD_EXE=`which bowtie-build`
      fi
fi

for c in 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 X Y ; do
      if [ ! -f hs_ref_chr$c.fa ] ; then
            if ! which wget > /dev/null ; then
                  echo wget not found, looking for curl...
                  if ! which curl > /dev/null ; then
                        echo curl not found either, aborting...
                  else
                        # Use curl
                        curl ${FILE_PATH}/hs_ref_chr$c.fa.gz
                  fi
            else
                  # Use wget
                  wget ${FILE_PATH}/hs_ref_chr$c.fa.gz
            fi
            gunzip hs_ref_chr$c.fa.gz
      fi

      if [ ! -f hs_ref_chr$c.fa ] ; then
```

Loops and Arrays

🐢 A few kinds of loops:

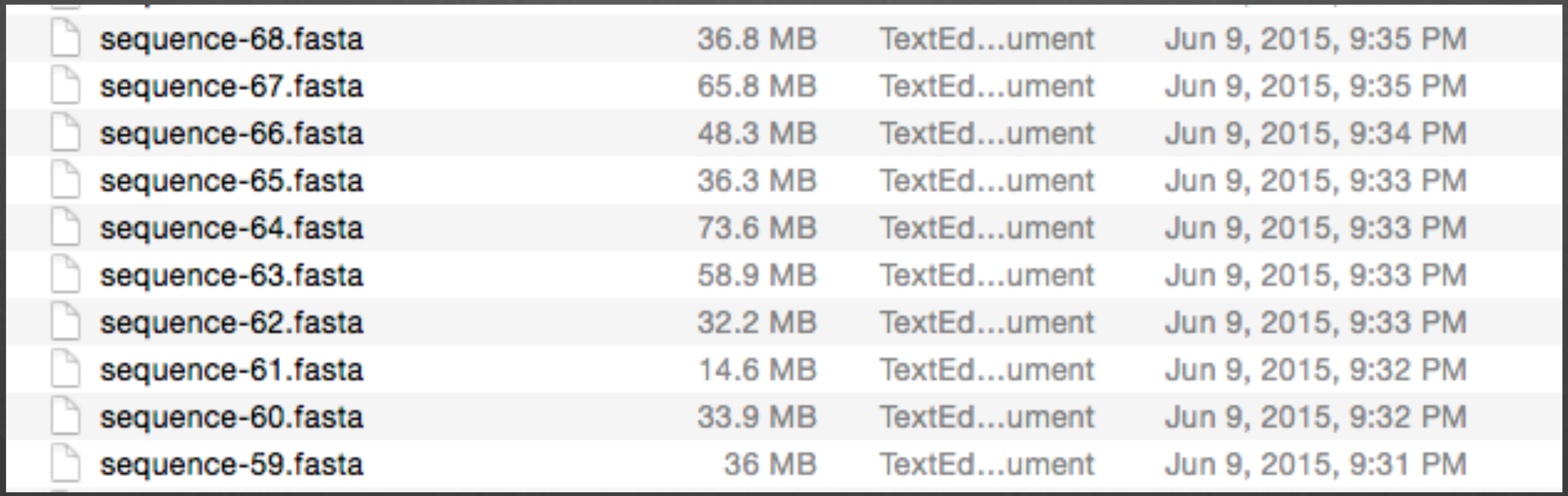
[while]	executes while a condition is true
[for]	operates on lists of items. It repeats a set of commands for every item in a list.
[until]	executes until a condition is true

🐢 An array is a variable containing multiple values. Array index starts with zero.

Loops and Arrays

🐢 A for loop example:

input: a list of fasta files with weird names
what I want: to rename the files with taxon info from the contigs inside the file.



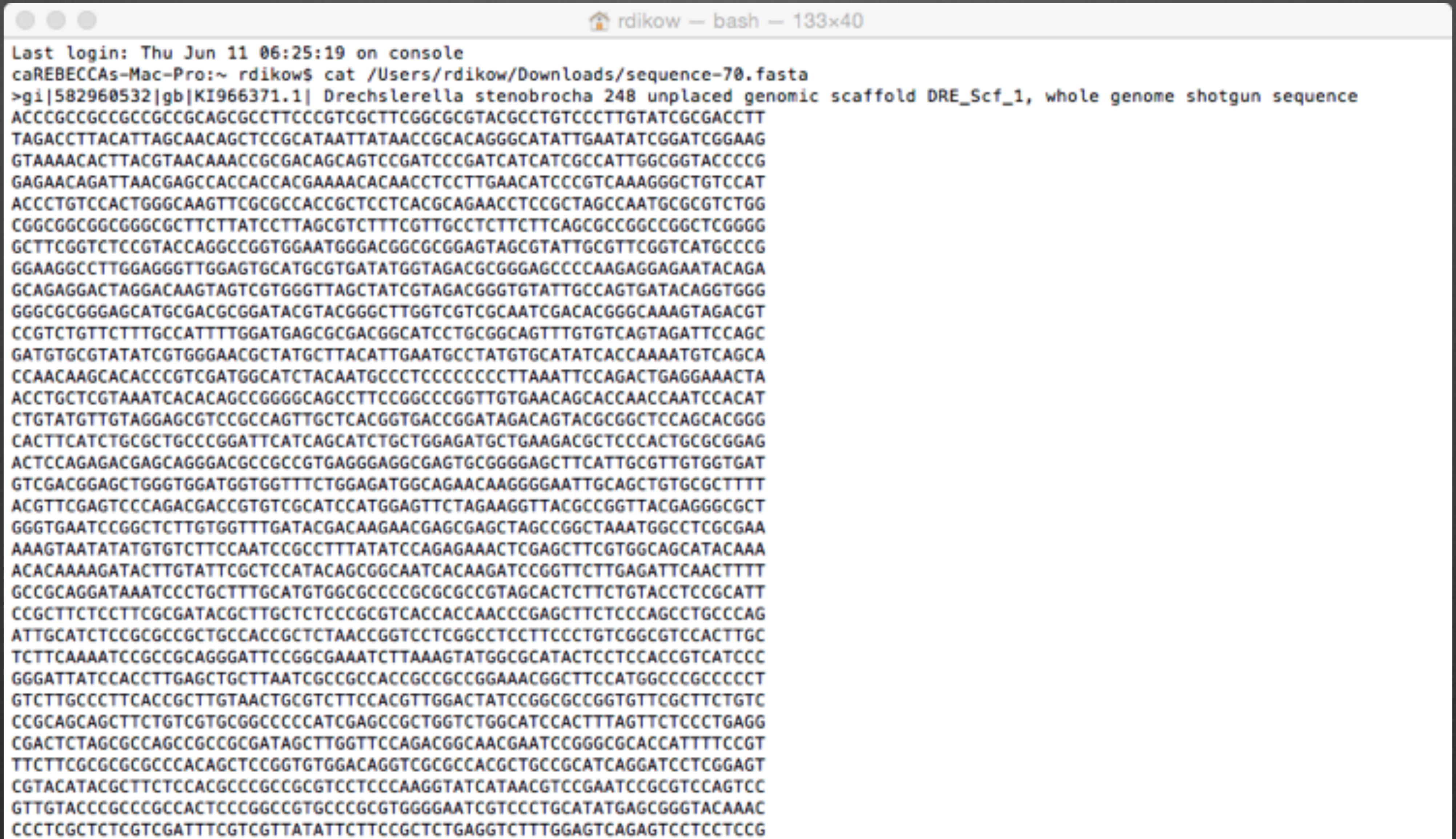
```
sequence-68.fasta    36.8 MB   TextEd...ument   Jun 9, 2015, 9:35 PM
sequence-67.fasta    65.8 MB   TextEd...ument   Jun 9, 2015, 9:35 PM
sequence-66.fasta    48.3 MB   TextEd...ument   Jun 9, 2015, 9:34 PM
sequence-65.fasta    36.3 MB   TextEd...ument   Jun 9, 2015, 9:33 PM
sequence-64.fasta    73.6 MB   TextEd...ument   Jun 9, 2015, 9:33 PM
sequence-63.fasta    58.9 MB   TextEd...ument   Jun 9, 2015, 9:33 PM
sequence-62.fasta    32.2 MB   TextEd...ument   Jun 9, 2015, 9:33 PM
sequence-61.fasta    14.6 MB   TextEd...ument   Jun 9, 2015, 9:32 PM
sequence-60.fasta    33.9 MB   TextEd...ument   Jun 9, 2015, 9:32 PM
sequence-59.fasta      36 MB   TextEd...ument   Jun 9, 2015, 9:31 PM
```

Loops and Arrays

A for loop example:

input: a list of fasta files with weird names

what I want: to rename the files with taxon info from the contigs inside the file.



rdikow — bash — 133×40

```
Last login: Thu Jun 11 06:25:19 on console
caREBECCAs-Mac-Pro:~ rdikow$ cat /Users/rdikow/Downloads/sequence-70.fasta
>gi|582960532|gb|KI966371.1| Drechslerella stenobrocha 248 unplaced genomic scaffold DRE_Scf_1, whole genome shotgun sequence
ACCCGCCGCCGCCGCCGCAGCGCCTTCCCGTCGCTTCGGCGCGTACGCCTGTCCCTTGTATCGCGACCTT
TAGACCTTACATTAGCAACAGCTCCGCATAATTATAACCGCACAGGGCATATTGAATATCGGATCGGAAG
GTAAAACACTTACGTAACAAACCGCGACAGCAGTCCGATCCCGATCATCATCGCCATTGGCGGTACCCCG
GAGAACAGATTAACGAGCCACCACCACGAAAACACAACCTCCTTGAACATCCCGTCAAAGGGCTGTCCAT
ACCCTGTCCACTGGGCAAGTTCGCGCCACCGCTCCTCACGCAGAACCTCCGCTAGCCAATGCGCGTCTGG
CGGCGGCGGCGGGCGCTTCTTATCCTTAGCGTCTTTCGTTGCCTCTTCTTCAGCGCCGGCCGGCTCGGGG
GCTTCGGTCTCCGTACCAGGCCGGTGGAATGGGACGGCGCGGAGTAGCGTATTGCGTTCGGTCATGCCCG
GGAAGGCCTTGGAGGGTTGGAGTGCATGCGTGATATGGTAGACGCGGGAGCCCCAAGAGGAGAATACAGA
GCAGAGGACTAGGACAAGTAGTCGTGGGTTAGCTATCGTAGACGGGTGTATTGCCAGTGATACAGGTGGG
GGGCGCGGGAGCATGCGACGCGGATACGTACGGGCTTGGTCGTCGCAATCGACACGGGCAAAGTAGACGT
CCGTCTGTTCTTTGCCATTTTGGATGAGCGCGACGGCATCCTGCGGCAGTTTGTGTCAGTAGATTCCAGC
GATGTGCGTATATCGTGGGAACGCTATGCTTACATTGAATGCCTATGTGCATATCACCAAAATGTCAGCA
CCAACAAGCACACCCGTCGATGGCATCTACAATGCCCTCCCCCCCCTTAAATTCCAGACTGAGGAAACTA
ACCTGCTCGTAAATCACACAGCCGGGGCAGCCTTCCGGCCCGGTTGTGAACAGCACCAACCAATCCACAT
CTGTATGTTGTAGGAGCGTCCGCCAGTTGCTCACGGTGACCGGATAGACAGTACGCGGCTCCAGCACGGG
CACTTCATCTGCGCTGCCCGGATTCATCAGCATCTGCTGGAGATGCTGAAGACGCTCCCACTGCGCGGAG
ACTCCAGAGACGAGCAGGGACGCCGCCGTGAGGGAGGCGAGTGCGGGGAGCTTCATTGCGTTGTGGTGAT
GTCGACGGAGCTGGGTGGATGGTGGTTTCTGGAGATGGCAGAACAAGGGGAATTGCAGCTGTGCGCTTTT
ACGTTCGAGTCCCAGACGACCGTGTCGCATCCATGGAGTTCTAGAAGGTTACGCCGGTTACGAGGGCGCT
GGGTGAATCCGGCTCTTGTGGTTTGATACGACAAGAACGAGCGAGCTAGCCGGCTAAATGGCCTCGCGAA
AAAGTAATATATGTGTCTTCCAATCCGCCTTTATATCCAGAGAAACTCGAGCTTCGTGGCAGCATACAAA
ACACAAAAGATACTTGTATTCGCTCCATACAGCGGCAATCACAAGATCCGGTTCTTGAGATTCAACTTTT
GCCGCAGGATAAATCCCTGCTTTGCATGTGGCGCCCCGCGCGCCGTAGCACTCTTCTGTACCTCCGCATT
CCGCTTCTCCTTCGCGATACGCTTGCTCTCCCGCGTCACCACCAACCCGAGCTTCTCCCAGCCTGCCCAG
ATTGCATCTCCGCGCCGCTGCCACCGCTCTAACCGGTCCTCGGCCTCCTTCCCTGTCGGCGTCCACTTGC
TCTTCAAAATCCGCCGCAGGGATTCCGGCGAAATCTTAAAGTATGGCGCATACTCCTCCACCGTCATCCC
GGGATTATCCACCTTGAGCTGCTTAATCGCCGCCACCGCCGCCGGAAACGGCTTCCATGGCCCGCCCCCT
GTCTTGCCCTTCACCGCTTGTAACTGCGTCTTCCACGTTGGACTATCCGGCGCCGGTGTTCGCTTCTGTC
CCGCAGCAGCTTCTGTCGTGCGGCCCCCATCGAGCCGCTGGTCTGGCATCCACTTTAGTTCTCCCTGAGG
CGACTCTAGCGCCAGCCGCCGCGATAGCTTGGTTCCAGACGGCAACGAATCCGGGCGCACCATTTTCCGT
TTCTTCGCGCGCGCCCACAGCTCCGGTGTGGACAGGTCGCGCCACGCTGCCGCATCAGGATCCTCGGAGT
CGTACATACGCTTCTCCACGCCCGCCGCGTCCTCCCAAGGTATCATAACGTCCGAATCCGCGTCCAGTCC
GTTGTACCCGCCCGCCACTCCCGGCCGTGCCCGCGTGGGGAATCGTCCCTGCATATGAGCGGGTACAAAC
CCCTCGCTCTCGTCGATTTCGTCGTTATATTCTTCCGCTCTGAGGTCTTTGGAGTCAGAGTCCTCCTCCG
```

Loops and Arrays

🐢 A for loop example:

input: a list of fasta files with weird names

what I want: to rename the files with taxon info from the contigs inside the file

```
for f in *.fasta; do mv "$f" "`head -1 $f|cut -d">" -f2`.fasta"; done
```

Moving and synchronizing files between local and remote hosts

(scp, sftp, rsync)

🐢

scp: secure copy

- a means of securely transferring computer files between a local host and a remote host or between two remote hosts
- based on SSH
- unsecure method rcp (remote copy) is not allowed

scp: secure copy

example - local to remote:

```
scp mygenome.fasta me@hydra-3.si.edu:/pool/genomics/me
```

-moves your file called “mygenome.fasta” to your pool/genomics directory

-will require authentication to hydra

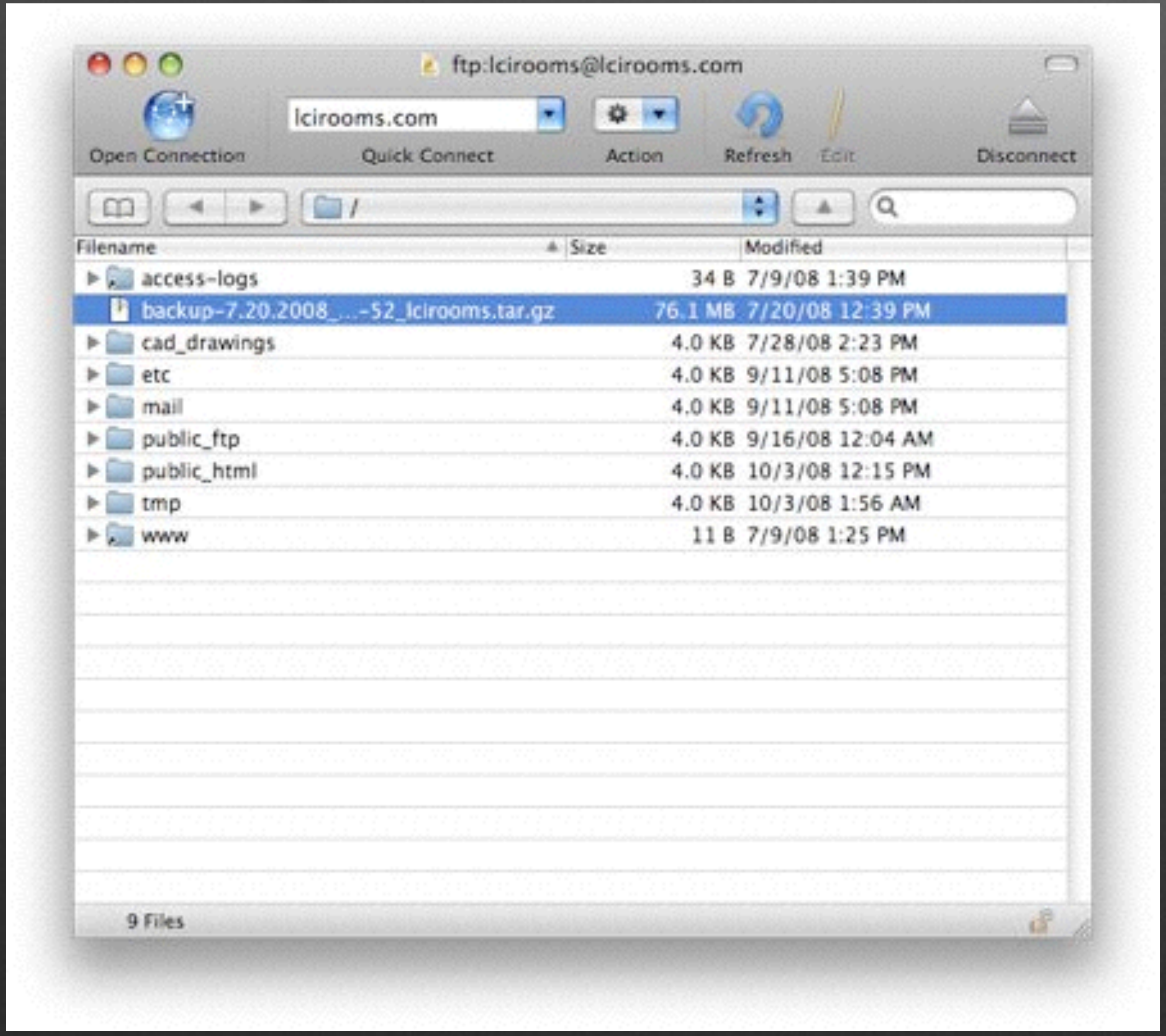
sftp: SSH File Transfer Protocol 🐢

- 🐢 Compared to scp, which only allows file transfers, the sftp allows for a range of operations on remote files, including resuming interrupted transfers, directory listings, and remote file removal
- 🐢 This is the secure version of ftp. Plain ftp is not allowed

scp: secure copy

CyberDuck

FileZilla



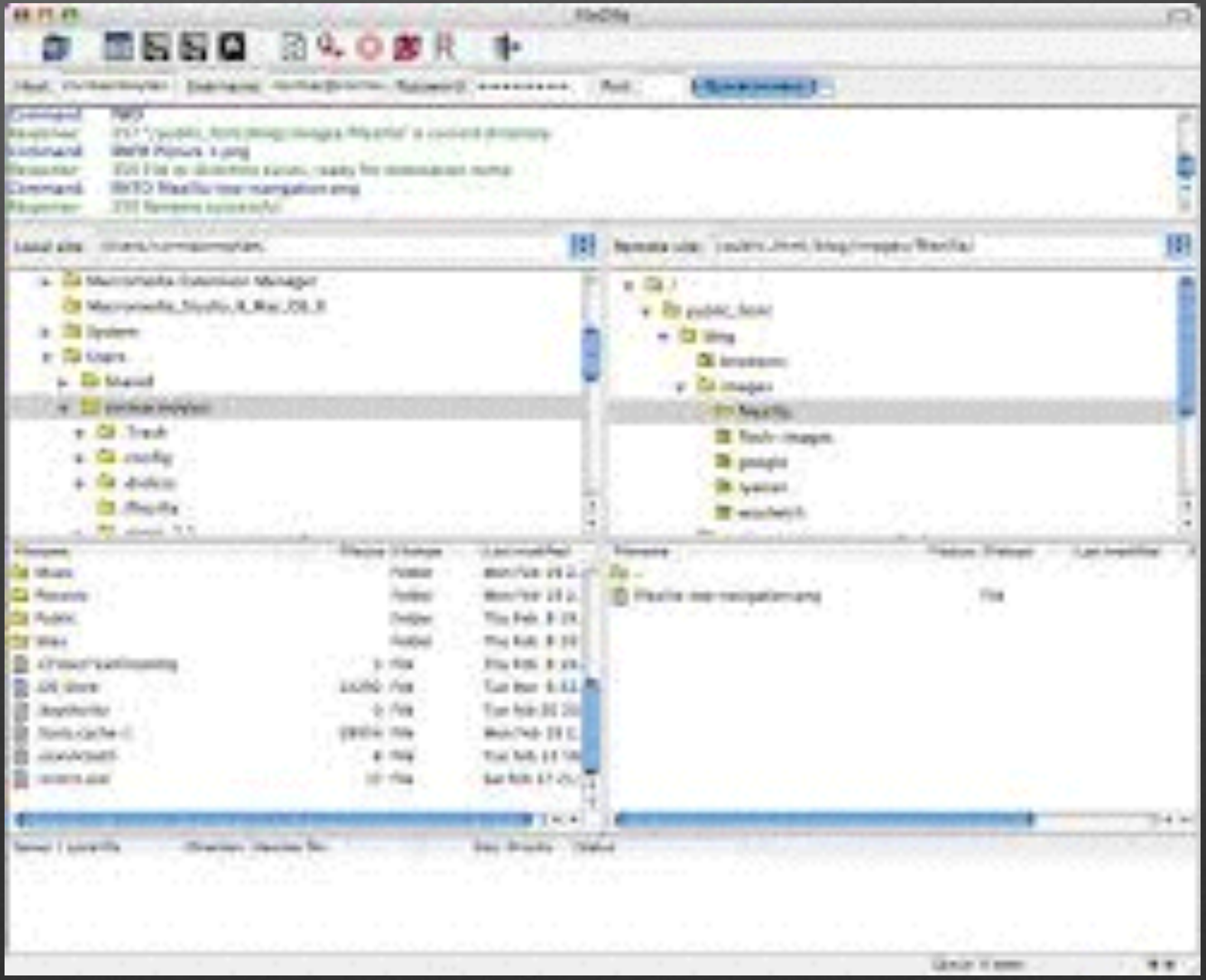
ftp:lcirooms@lcirooms.com

lcirooms.com

/

Filename	Size	Modified
access-logs	34 B	7/9/08 1:39 PM
backup-7.20.2008_...-52_lcirooms.tar.gz	76.1 MB	7/20/08 12:39 PM
cad_drawings	4.0 KB	7/28/08 2:23 PM
etc	4.0 KB	9/11/08 5:08 PM
mail	4.0 KB	9/11/08 5:08 PM
public_ftp	4.0 KB	9/16/08 12:04 AM
public_html	4.0 KB	10/3/08 12:15 PM
tmp	4.0 KB	10/3/08 1:56 AM
www	11 B	7/9/08 1:25 PM

9 Files



rsync

- 🐢 utility to keep copies of a file on two computer systems the same
- 🐢 functions as both a file synchronization and file transfer program
- 🐢 Zlib may be used for additional compression, and SSH can be used for data security
    - 🌻 Compression reduces network traffic
- 🐢 rsync determines which files differ between the sending and receiving systems by checking the modification time and size of each file

rsync

example:

```
rsync mygenome.fasta me@hydra-3.si.edu:/pool/genomics/me/mygenome.fasta
```

🐢 uses SSH to connect as user to remote-host (Hydra)

🐢 once connected, it will invoke Hydra's rsync and the two programs will determine what parts of the file need to be transferred over the connection

🐢 can do the same for whole directories

other useful commands

`tar`	archive a file system:
	`[-c]` create new archive
	`[-z]` uncompress
	`[-x]` extract to disk
	`[-v]` verbose
	`[-f]` file
`gzip`	compress and replace with a .gz file
`gunzip`	uncompress a .gz file