Introduction to Programming
The tcsh Shell

D. Thiebaut
CSC212 — Fall 2014
OS = manager of everything!

- Disk
- Display
- Printer
- USB Device
- Clock
- USB Device
- USB Device
- RAM
- Processor
- Window Manager
- Operating System
- Users
Display

Window Manager

Operating System

Shell

\{ \text{bash, sh, csh, tcsh, etc...} \}

shell = interpreters
tcsh, the tee-cee-shell

http://en.wikipedia.org/wiki/Tcsh
History

- The built-in `history` command displays the previously-entered commands
- Use of ↑ / ↓ at the command line to allow the user to select a command from the history to edit/execute
- Invoking previous commands using `command history`
  - `!!` executes the previous command
  - `!n` executes the nth command that was previously executed
  - `!-n` executes the command that was executed n commands ago
  - `!string` executes the most recently executed command that starts with string
  - `!?string` executes the most recently executed command that contains string
- Using history in new commands
  - `!*` - refers to all of the arguments from the previous command
  - `!$` - refers to the last argument from the previous command
  - `!^` - refers to the first argument from the previous command
  - `!n` - refers to the nth argument from the previous command
  - `!m-n` - refers to the mth through nth arguments from the previous command
  - `!n-$` - refers to the nth through the last argument from the previous command
Variables

```
beowulf2:$ handout> set x=100
beowulf2:$ handout> echo $x
100
beowulf2:$ handout> set name=Sophia
beowulf2:$ handout> echo $name
Sophia
beowulf2:$ handout> set fullName="Smith College"
beowulf2:$ handout> echo $fullName
Smith College
beowulf2:$ handout> set files=`ls`
beowulf2:$ handout> echo $files
file1.txt file2.txt file3.txt fulghum generateDNAFiles.py generateDNA.py
last.txt README temp/ ulysses.txt
```
Loops

```
beowulf2:~$/handout> foreach x ( one two three )
foreach? echo $x
foreach? end
one
two
three
```

More Loops

```bash
beowulf2:~/handout/temp> foreach n ( `seq 0 9` )
foreach? echo "DNA_$n.txt"
foreach? end
DNA_0.txt
DNA_1.txt
DNA_2.txt
DNA_3.txt
DNA_4.txt
DNA_5.txt
DNA_6.txt
DNA_7.txt
DNA_8.txt
DNA_9.txt
beowulf2:~/handout/temp> 
```
Some Linux Commands
Linux Command: `grep`

```
beowulf2:~/temp> getcopy ulysses.txt
beowulf2:~/temp> grep magenta ulysses.txt
of an elder in Zion and a smokingcap with magenta tassels. Horned
beowulf2:~/temp>
beowulf2:~/temp> grep -i magenta ulysses.txt
of an elder in Zion and a smokingcap with magenta tassels. Horned
beowulf2:~/temp> wc ulysses.txt
  32663  264965 1520798 ulysses.txt
beowulf2:~/temp> wc -l ulysses.txt
32663 ulysses.txt
beowulf2:~/temp> 
```
Linux Commands: **tail & head**

```bash
beowulf2:~/temp> head -7 ulysses.txt

ULYSSES

by James Joyce
```

```bash
beowulf2:~/temp> tail -7 ulysses.txt

heart was going like mad and yes I said yes I will Yes.

Trieste–Zurich–Paris 1914–1921
```

```bash
beowulf2:~/temp>  
```
Redirection & Pipes
(No) Redirection

stdin ➔ Program ➔ stdout

`java HelloWorld`
Hello World!

`ls -1`
- file1.txt
- file2.txt
- file3.txt
Redirection

stdin → Program → output file

```
java HelloWorld > outputFile.dat

ls -1 > listFiles.txt
cat listFiles.txt
file1.txt
file2.txt
file3.txt
```
Repeated Redirection

stdin ➔ Program ➔ output file

```
java HelloWorld > outputFile.dat
java OtherProg >! outputFile.dat

head -10 ulysses.txt > sample.txt  # (create)
echo “…” ➔ sample.txt  # (append)
tail -10 ulysses.txt ➔ sample.txt  # (append)
```
Redirection

input file → Program → stdout

cat > data.in

1
2
3
^D

java MyProg < data.in
Redirection

input file → Program → output file

$ wc < ulysses.txt > ulysses.info$

$ cat ulysses.info$

32663  264965  1520798
Pipes

grep orange ulysses.txt | wc -l

grep orange ulysses.txt | tail -3
Problem:

• Get several DNA files from http://cs.smith.edu/~212a/DNAFiles/, named DNA_0000.txt to DNA_0099.txt

• Find how many files have the special string “CGTGACTCAA” in them, aka TPA-response element.

• Identify the files.
Summary

• Using programming languages, one writes programs where the library functions allow one to manipulate data.

• Using shell commands, one can write programs, called scripts, where the functions available are operating-system commands, allowing one to manipulate operating system objects, such as files, Web data, devices, network information, or user information.
#!/bin/tcsh
# findDNA.sh
# D. Thiebaut
# Demo script for tcsh lecture, CSC212
#

# URL where DNA files are residing
set url=http://cs.smith.edu/~212a/DNAFiles

# pattern to search in DNA files
set pattern=CGTGACTCAA

# download all files DNA_000x where x is 0 to 9
foreach n (`seq 0 9`)
    wget -q $url/DNA_000$n.txt
end

# download all files DNA_00x where x is 10 to 99
foreach n (`seq 10 99`)
    wget -q $url/DNA_00$n.txt
end

# count number of files downloaded
set num=`ls DNA* | wc -l`
echo "Downloaded $num DNA files"

# identify files containing pattern and save to file
grep -l $pattern DNA* >! $pattern.txt

# remove all DNA files
unalias rm
rm DNA_*