Dr. Sebastian Jentschke
Cluster of Excellence „Languages of Emotion“

Freie Universität Berlin

Shell Programming

MSc course
„Social Cognitive and Affective Neuroscience“

Overview

- what is a shell
- why shell progamming?
- basics of command syntax
- basic concepts
- control structures
- dealing with files
  touch – exist – cp / mv
- extracting and changing information from files or variables
  grep – head – tail – sed p
  sed s – awk – cut
- input and outoput

What is a shell?

- interface to an operating system, usually based on a command line (as opposed to a graphical user interface)
- primarily to invoke or „launch“ another program
- certain operations can be performed much faster using a command line
  ↕
  graphical user interfaces usually have higher usability and simplicity
- the standard shell under Linux and on Macs is usually „bash“

Why shell programming?

- extracting information from text files (e.g. log files)
- preparing your stimuli (e.g. normalizing volume, change attributes of a picture)
- conversion and analysis of neurophysiological data
  - FSL, FreeSurfer, AFNI, MRIConvert
  - changing your directory structure – copying first level analysis files from SPM
  - EEP

Basics of command syntax

- most UNIX commands require two parameters, e.g.,
  `cp INPUT/SOURCE OUTPUT/DESTINATION`
- often they output information to the screen called „stdout“, e.g., `less INPUT/FILENAME`

  thus, you usually can concatenate commands using pipes:
  `less FILENAME | grep NEO`
  ([1] outputs the content of FILENAME, and [2] extracts lines containing NEO)
- help about the parameters can be obtained with `--help`
  `grep --help`
- if you don't know, which command to use try Google, e.g.,
  bash remove duplicate lines

Basic concepts

- **variables**
  directly assign a value, e.g. P="/home/sjentsch/Desktop"
  assign result of a command, e.g. F=$(find *.wav | head -n 1)
  references to a variable need to be enclosed in ${VAR}, e.g.,
  `echo ${P}` (would output „/home/sjentsch/Desktop" to the screen)
- **pipes**
  diverts the output of one command to the next or into a file
  |, >, >>, <
- **regular expressions:** [a-Z], [A-Z], [0-9], \w, \n, ^, $, *, ?, .
- **boolean operators: ||, &&**
- **comparisions: ==, gt, lt, le, ...**

Dealing with files

- **for … in … do**
  goes through a list of files the match the pattern, e.g.,

```
for F in ^[0-9]*_S*.wav; do
   echo ${F}
done
```

(goes though all wav-files starting with a number and shows their name)

- **exist, [-e FileName]**
  checks the presence of a file, usually with if

```
if [ -e FILENAME ]; then ...; fi
```

- **cp / mv**
  copies or moves files

```
cp ${F} /home/sjentsch/Desktop/Destination
```

Control structures

- **if … then**
  checks whether a codition is satisfied, e.g. a file exists

```
if [ -e FileName ]; then echo "File exist"; fi
```

- **while-loops**
  runs while a condition is satisfied, e.g.

```
N = 0; while [ ${N} <= 100 ]; do N =$[${N} + 1]; done
while [ -e FileName ]; do ...; done
```

for-loops
runs for a defined number of iterations, e.g. a file list

```
for N in $(seq 1 45); do echo ${N}; done
for F in *.wav; do echo ${F}; done
```

Freie Universität Berlin

Extracting or changing information (from files)

- **grep**
  extracts lines that match a pattern, e.g.
  ```
  grep Stimulus X01_01.log
  ```
  you can use regular expressions, e.g. ^, $, etc.

- **head / tail / sed -n**
  extracts particular lines from a file
  ```
  head -n 10 FileName
  sed -n 2,4p FileName
  ```

Extracting or changing information (from files)

- **sed s**
  replaces all occurrences of the first pattern with the second
  `sed s/OLD/NEW/ FileName`
  also takes regular expressions (`sed -e`)

- **awk**
  script language for data extraction and reporting

```
echo Hallo Welt  | awk '{printf "%s %s!\n",$1,$2}'
echo 1 2 | awk '{printf "%03d %03d\n", $1, $2, $3}'
```

- **cut**
  extracts information from a string, e.g. characters or fields

```
echo ABCD_EFGH.wav | cut -d _ -f 2 | cut -d . -f 1
```

Dr. Sebastian Jentschke
Cluster of Excellence „Languages of Emotion“

Further useful commands

- more / less
- read -n X NAME
- sleep XX
- printf, e.g., printf "%6.3f" 4
- echo
- seq
- wc

Dr. Sebastian Jentschke
Cluster of Excellence „Languages of Emotion“

Getting help...

Information about a commands' syntax:

```
command --help
```

Introduction to shell scripting:

- http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html
- http://www.gnu.org/software/bash/manual/bash.html
- tcsh (Mac OS X) vs. bash (most Linux OS)

… and: Google is your friend

Dr. Sebastian Jentschke
Cluster of Excellence „Languages of Emotion“

That's it folks...

Thank you for your attention!