LING 408/508: Programming for Linguists

Lecture 4
September 2\textsuperscript{nd}
Administrivia

• Did people manage to install Ubuntu as a guest OS inside VirtualBox?
• (We'll try to reserve a bit of time at the end.)

• Today’s Topics:
  – Guest Additions on Vbox
    • (resizing the screen works after install...)
    • https://www.virtualbox.org/manual/ch04.html
  – Some simple tasks using bash
  – Homework 3: write a program
VirtualBox Guest Additions

activated using VirtualBox: shows up as a disc in your Ubuntu system
VirtualBox Guest Additions

Removing existing VirtualBox non-DKMS kernel modules ...done.
Building the VirtualBox Guest Additions kernel modules
The headers for the current running kernel were not found. If the following
module compilation fails then this could be the reason.

Building the main Guest Additions module ...done.
Building the shared folder support module ...done.
Building the OpenGL support module ...done.
Doing non-kernel setup of the Guest Additions ...done.
Starting the VirtualBox Guest Additions ...done.
Installing the Window System drivers
Installing X.Org Server 1.15 modules ...done.
Setting up the Window System to use the Guest Additions ...done.

You may need to restart the hal service and the Window System (or just restart
the guest system) to enable the Guest Additions.

Installing graphics libraries and desktop services components ...
done.
Press Return to close this window...

a Linux popup will ask you to if it’s okay to run the disc
Ubuntu

• Ubuntu Software Center
  – App store
  – (full screen to see Search box)

• Software packages
  – Terminal: `sudo` apt-get install `<pkg-name>`
  – `sudo` prefix: means execute the apt-get command with superuser privileges (typically needed for packages)

• How to find Terminal: use search
Ubuntu

• Terminal:
  – runs a shell: **bash**
  – enter commands: some are built-in to the shell, others are executable files in specified directories ($PATH), still others will require apt-get
  – command history is editable (up-arrow to retrieve...)
  – pre-defined environment variables: **env**
  – lots of packages are pre-loaded: **wish, python, perl**, etc.
  – **dpkg** (package manager for Debian)
  – **man command-name** (brings up manual page)

```
sandiway@sandiway-VirtualBox:~$ swipl
The program 'swipl' is currently not installed. You can install it by typing:
sudo apt-get install swi-prolog-nox
```
Shell

• simple commands:
  – `pwd` print working directory
  – `ls (ls -a)` list current directory
    (`-a` option: show . (dot) files too)
  – `cd` change directory
  – `mkdir` create a new directory
  – `which name` prints the directory where command `name` is located, or nothing if it can’t be found in the PATH
  – `man name` display manual page for command `name`
  – `echo $SHELL` prints the shell ($ prefixes a variable)
  – `echo $PATH` shows the directories where the shell will look for commands
Shell

Directory shortcuts:

- your home directory: ~
- current directory: .
- parent directory: ..

Examples:
- `cd ..` (go to parent directory)
- `mkdir ~/tmp` (create a new directory called tmp in your home directory)
- `touch tmp` (create a new file tmp in the current directory if tmp doesn’t already exist, or update the timestamp)
- `ls -l tmp` (list attributes of file tmp in long format)

```
-rw-r--r--  1 sandiway  staff  0 Sep  4 09:26 tmp
```

<table>
<thead>
<tr>
<th>permissions</th>
<th>owner</th>
<th>group</th>
<th>size (bytes)</th>
<th>date time modified</th>
<th>filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>user group all</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r = read</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w = write</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x = execute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
cat command

• See http://www.linfo.org/cat.html

1. `cat file1` (print contents of file1)
2. `cat file1 > file2` (‘>’ = redirect output to file2)
3. `cat file2 | more` (‘|’ = pipe output to command more)
4. `more file1 – easier` (stops at end of screen, hit space to show more)
5. `less file1 – easier` (allows page by page display)
6. `cat > file1` (create file1 with input from terminal until Control-D EOF)
7. `cat` (input from terminal goes to terminal)
8. `cat >> file1` (append input from terminal to file file1)
9. `cat file1 > file2` (file copy)
10. `cp file1 file2 – easier` (cp = copy)
11. `cat file1 file2 file3` (prints all 3 files)
12. `cat file1 file2 file3 > file4` (prints all 3 files to file4)
13. `cat file1 file2 file3 | sort > file4` (3 files sorted alphabetically to file4)
14. `cat – file5 > file6` (‘-’ = input from terminal)
15. `cat file7 - > file8`
Shell Arithmetic

• at the shell prompt:
  – expr 1 + 3  
    (cf. expr 1+3)
  – echo `expr 1 + 3`
  – i=2  
    (NO SPACES! cf. i = 2)
  – expr $i + 1
  – let x=1+3  
    (cf. let x=1 + 3)
  – echo $x
  – let i=$i+1  
    (also ok let i=i+1)
  – echo $i
  – ((x = 1+ 3))  
    (spaces not significant)
  – echo $x
  – echo $((1+3))
  – (i=i+1)  
    (also ok let i=$i+1)
Comparison operators

• Format:
  
  if [ $x OP $y ]; then
  ...
  (else/elif...)
  fi
  
  [ .... ] is known as test

• OP:
  • -eq equals
  • -ne not equals
  • -gt greater than
  • -ge greater than or equals
  • -lt less than
  • -le less than or equals

• Examples:
  
  – echo $x $i
  2 5
  
  – test $x -le $i
  
  – echo $?  
  (exit status)
  0

  – test $x -le $i -a $i -lt $x
  – echo $? 
  1
Input

• At a terminal:
  – read –p “Name: ” name
  – read –p “Enter X and Y: ” x y
  – echo $x
  – echo $y
Positional Parameters

• In a shell script:
  – $1: first parameter
  – $#: number of parameters

• Program:
```bash
#!/bin/bash
echo "Number of parameters: $#"
if [ $# -eq 1 ]; then
echo "1st parameter: $1"
fi
```

• Output:
  – sh test.sh
  Number of parameters: 0
  – sh test.sh 45
  Number of parameters: 1
  1st parameter: 45
  – sh test.sh 45 56
  Number of parameters: 2
Running shell scripts

• Supply program filename as a parameter to sh/bash:
  - sh test.sh
  - bash test.sh
  - source test.sh
  - . test.sh
  - (. = source)

• Run the program in the current directory: (./ needed if current directory is not in PATH)
  - ./test.sh
    -bash: ./test.sh: Permission denied

  - ls -l test.sh
    -rw-r--r-- 1 sandiway staff 98 Sep 4 09:14 test.sh

  - chmod u+x test.sh
    - ls -l test.sh
    -rw-r-xr-- 1 sandiway staff 98 Sep 4 09:14 test.sh

  - ./test.sh
  Number of parameters: 0
First Non-trivial Program

• Let’s write a simple shell-script BMI calculator — it can solicit input from the terminal or take command line arguments

<table>
<thead>
<tr>
<th>Measurement Units</th>
<th>Formula and Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kilograms and meters (or centimeters)</td>
<td>Formula: weight (kg) / [height (m)]^2</td>
</tr>
<tr>
<td></td>
<td>With the metric system, the formula for BMI is weight in kilograms divided by height in meters squared. Since height is commonly measured in centimeters, divide height in centimeters by 100 to obtain height in meters.</td>
</tr>
<tr>
<td></td>
<td>Example: Weight = 68 kg, Height = 165 cm (1.65 m)</td>
</tr>
<tr>
<td></td>
<td>Calculation: 68 / (1.65)^2 = 24.98</td>
</tr>
</tbody>
</table>

| Pounds and inches                        | Formula: weight (lb) / [height (in)]^2 x 703                                                    |
|                                          | Calculate BMI by dividing weight in pounds (lbs) by height in inches (in) squared and multiplying by a conversion factor of 703. |
|                                          | Example: Weight = 150 lbs, Height = 5'5" (65")                                                  |
|                                          | Calculation: [150 / (65)^2] x 703 = 24.96                                                      |
First Non-trivial Program

• First pass:

```bash
#!/bin/bash
if [ $# -ne 2 ]; then
    echo "usage: weight in kg, height in m"
    exit 1
fi
((bmi = $1 / ($2 * $2)))
echo $bmi
```
BMI calculator

• Did you notice bash can only do integers?
  – can use bc (an arbitrary precision calculator)
  – scale = # of decimal places
  – echo "scale=2;2/3" | bc –q
    .66
  – but test comparisons (–gt etc.) would then be a pain in the butt

  – can re-scale the formula:
    Example:
    – weight in kg * 1,000,000 / (height in cm)**2
    – echo $((68* 1000000 / (165 * 165)))
    2497 (24.97)

  Explanation:
  1. echo "..." | means send "..." to next command
  2. bc –q means quiet mode

pi=`echo "scale=50; 4*a(1)" | bc –l`
  echo $pi
  3.14159265358979323846264338327950288419716939937508

  Explanation:
  1. `...` is command substitution
  2. bc -l, use the mathlib
  3. In bc, a(1) computes tan^{-1}(1), which is \pi/4
First Non-trivial Program

• Some possibilities:

```bash
#!/bin/bash
if [ $# -ne 2 ]; then
    echo "usage: weight in kg, height in cm"
    exit 1
fi
(((bmi = $1 * 1000000 / ($2 * $2))))
echo $bmi
```

```bash
#!/bin/bash
if [ $# -ne 2 ]; then
    echo "usage: weight in kg, height in cm"
    exit 1
fi
(((bmi = $1 * 1000000 / ($2 * $2))))
echo "scale=2;$bmi/100" | bc -q
```

```bash
#!/bin/bash
if [ $# -ne 2 ]; then
    echo "usage: weight in kg, height in m"
    exit 1
fi
echo "scale=2;$1/($2*$2)" | bc -q
```
Homework 3

• Modify the BMI calculator to:
  1. accept either command line arguments or read from the terminal if they’re missing
  2. print the weight status message according to the following table:
  3. modify the calculator to accept input in both metric and traditional units

<table>
<thead>
<tr>
<th>BMI</th>
<th>Weight Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Below 18.5</td>
<td>Underweight</td>
</tr>
<tr>
<td>18.5 – 24.9</td>
<td>Normal</td>
</tr>
<tr>
<td>25.0 – 29.9</td>
<td>Overweight</td>
</tr>
<tr>
<td>30.0 and Above</td>
<td>Obese</td>
</tr>
</tbody>
</table>
Homework 3

• Submit your homework by midnight on Sunday to sandiway@email.arizona.edu
• Show your program and sample input and output
• One PDF file please