Lecture 3: Unix Recap, Java (cont’d)

CS 170, Section 000, Fall 2009
3 September 2009
Lecture Plan

• **Logistics (important announcements)**
• Lab1 postmortem
• Unix: commands, examples, resources
• Java recap from Lecture 1
• Variables
• Assignments and Expressions
• Numeric Data and Operations
Important Announcement

• For the next 3 (three) lectures I will be **away** at a conference in Russia ([http://romip.ru/russir2009/eng/index.html](http://romip.ru/russir2009/eng/index.html)) with limited e-mail access.

• **Sebastien Siva**, the instructor for the other section of this course ([http://www.mathcs.emory.edu/~cs170001/](http://www.mathcs.emory.edu/~cs170001/)) will teach the lectures on **Tue 9/8, Thu 9/10, and Tue 9/15**.

• If you have questions, come to **Sebastian’s** office hours in **N414**:
  – M 10:00am - 12:00pm
  – W 1:45pm - 3:00pm

• He will assign the first **homework** which will be due **after** I return.

• I will be back teaching you for the rest of the semester (no planned travel) starting **Thu, Sept 17th**.
Lab 1 Postmortem

• Skills acquired:
  – Navigating directory/file structure in Unix
  – Copying, creating, moving files and directories
  – Compiling and running Java programs
  – Creating and executing Java printing statements

• Main problems:
  – Command-line interface to Operating System (OS)
  – Unfamiliar commands

• Solution:
  – Do this exercise (on your own time). Also available from “Resources” page on class website
  – Working from home: instructions posted on Resources
Unix Review/Tutorial
Unix review

• Unix is an **operating system**
• Manages processes (programs)
• Processes can start **child** processes
  – Unix shell (terminal) is one very important process
  – Processes can be **strongly** or **weakly** bound to parent
  – Strongly bound processes *usually* die when parent exits; otherwise, they are known as **zombie** processes
  – Weakly bound processes can live after parent exits
• There are hundreds of processes running at any time
• Run **top** command to see process, owners, resources
UNIX Commands

• `ls [names]` – list files contained in a directory *name* or that match a file *name*. If no *name* is given list those files in current directory.
  – `ls –a` list all files including hidden files
  – `ls –l` list in long format (including details like permissions, owner, size, etc.), works very much like `dir`
  – `ls –al` list all files (including hidden files) in long format
  – `ls –dl dir_name` lists information about the directory, “dir_name”.
What is a Directory?

Your home directory might contain a `public_html` directory. Your `public_html` directory might contain an "index.html" file.
What is a Directory?

A file cannot hold a directory or a file!

c3063xxx

public_html

index.html

New_file_or_directory
What is directory?

Directories can hold files and other directories.
What’s a directory?

• Files are grouped in the directory structure. The file-system is arranged like hierarchical tree (inverted)structure.

• The top of the tree is called “root” which usually contains several sub-directories. In UNIX “/” (forward slash) is used to present the “root”.
Pathnames

- **Absolute Pathnames**
  - In the previous tree `/users/usern/file1` is an absolute pathname.

- **Relative pathnames**
  - If you are already in the `users` directory, the relative pathname for `file1` is `usern/file1`. 
What is the **absolute path** to `index.html`? 

```
/  
  |  
bin  etc  users  tmp  backup  
  |    |   |    |    |    |  
user1  user2  ...  usern  public_html  file1  
  |    |    |  |    |    |  
         index.html
```
Specifying Paths

What is the **relative path** to `index.html` (assuming that `usern` is your **current dir**)?
More UNIX commands

• `pwd` – let you know the absolute pathname of your current working directory (Print Working Directory)

• `cd [dir]` – change directory
  
  – `cd ..` – go back to parent directory. “..” is the relative pathname to the parent directory.
  
  – “.” - stands for current (working) directory.
  
  – “~” – the tilde ~ character can refer your home directory.
In-Class Exercise: Navigating Directories

Shell Tutor by Sebastien Siva:
http://www.mathcs.emory.edu/~ssiva/ShellTutor/
More UNIX commands

• `mkdir directories` — create one or more directories. You can specify them by absolute or relative pathnames.

• `cp`
  – `cp file1 file2` — copy `file1` to `file2`. If there’s already a `file2`, the old one will be overwritten.
  – `cp file(s) directory` — `file(s)` will be copied to the `directory`. 
More UNIX commands

- `mv sourcefile targetfile` – rename `sourcefile` to `targetfile`.
  - If there’s a file with the same name as `targetfile`, it may be overwritten.
  - `mv` works for directories the same way.
Exercise: Make your homepage

• Create directory `share/public_html` in your home directory
• Change to this directory
• Open or create file `index.html` using `gedit`
  – Insert content into `index.html`:
    ```html
    <html>
    <h1>Hello World!</h1>
    </html>
    ```
  – Save file
• Run `firefox` in **background** (`firefox &`)
• Open URL http://www.mathcs.emory.edu/~usern/
Permissions

• There are three types of file access supported by UNIX.
  
  – r – read, view the contents of a file or a directory
  
  – w – write, edit file/directory contents
  
  – x – execute, run executable file
Permissions

Here’s an example

– Suppose you type in `ls -l` and the result is

```
-rwx r-x r-- 1 hans doc 858 Aug 22 22:28 hw1
```

What do all these symbols mean?
Permissions

- rwx r-x r-- 1 hans doc 858 Aug 22 22:28 hw1
Permissions

- **Who:**
  - u: User – the person who created the file.
  - g: Group – the group owns the file.
  - a: All – the rest of the world

- **What:**
  - r: can read (access)
  - x: can execute (run)
  - w: can modify (delete/create)

- **chmod mode file(s)** – changes file or directory permissions
  - chmod a+rx pub
  - chmod a+r index.html: make index.html readable by all

- Try to open your homepage again (reload)
- Voila!
Java (Cont’d)
Running Example: ComputeArea.java

• Make directory for today’s example code:
  cd ~/cs170
  mkdir inclass
  cp –r /home/cs170000/inclass/sept3/ inclass
  cd inclass/sept3

• gedit ComputeArea.java &
public class ComputeArea{
    // computes area of a circle
    public static void main(String[] args){
        // read radius
        ...
        // compute the area
        ...
        // print the area
        ...
    }
}
Variables

- A variable is a name for a location in memory used to hold a data value.
- Type, name and contents
- Using a variable
  - Declaring a variable – type and name
    - Instructs the compiler to reserve a portion of main memory to hold a particular type of value referred by a particular name
  - Assign a value to a variable
Syntax: Variable Definition

```plaintext
typeName variableName;
```

**Example:**
```
int luckyNumber;
```

**Purpose:**
To define a new variable of a particular type
Syntax: Assignment

Syntax:

```
variableName = expression;
```

Example:

```
luckyNumber = 12;
luckyNumber = 5+7;
```

Purpose:

To assign a new value to a previously defined variable.
Data Types

- Fundamental or primitive data types and object types
- 8 primitive types
  - 6 number types: four integer types and two floating point types
  - 1 character type
  - 1 boolean type
### Numerical Data Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Range</th>
<th>Storage Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>$-2^7$ (-128) to $2^7-1$ (127)</td>
<td>8-bit signed</td>
</tr>
<tr>
<td>short</td>
<td>$-2^{15}$ (-32768) to $2^{15}-1$ (32767)</td>
<td>16-bit signed</td>
</tr>
<tr>
<td>int</td>
<td>$-2^{31}$ (-2147483648) to $2^{31}-1$ (2147483647)</td>
<td>32-bit signed</td>
</tr>
<tr>
<td>long</td>
<td>$-2^{63}$ to $2^{63}-1$ (i.e., -9223372036854775808 to 9223372036854775807)</td>
<td>64-bit signed</td>
</tr>
<tr>
<td>float</td>
<td>Negative range: -3.4028235E+38 to -1.4E-45</td>
<td>32-bit IEEE 754</td>
</tr>
<tr>
<td></td>
<td>Positive range: 1.4E-45 to 3.4028235E+38</td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>Negative range: -1.7976931348623157E+308 to -4.9E-324</td>
<td>64-bit IEEE 754</td>
</tr>
<tr>
<td></td>
<td>Positive range: 4.9E-324 to 1.7976931348623157E+308</td>
<td></td>
</tr>
</tbody>
</table>
Floating Point Numbers

Calculations involving floating-point numbers are approximated because these numbers are not stored with complete accuracy. For example,

```java
System.out.println(1.0 - 0.1 - 0.1 - 0.1 - 0.1 - 0.1);
```
displays 0.5000000000000001, not 0.5, and

```java
System.out.println(1.0 - 0.9);
```
displays 0.0999999999999998, not 0.1. Integers are stored precisely. Therefore, calculations with integers yield a precise integer result.
Number demo

• An excellent tool to demonstrate how numbers are stored in a computer was developed by Richard Rasala. You can access it at

http://www.ccs.neu.edu/jpt/jpt_2_3/bitdisplay/applet.htm
Identifiers

- An identifier is a sequence of characters that consist of letters, digits, underscores (\_), and dollar signs ($).

Rules for Java identifier

- An identifier must start with a letter, an underscore (\_), or a dollar sign ($). It cannot start with a digit.
- An identifier cannot be a reserved word. (See Appendix A, “Java Keywords,” for a list of reserved words).
- An identifier cannot be `true`, `false`, or `null`.
- An identifier can be of any length.
Identifier Conventions

• variable names start with a lowercase letter
• class names start with an uppercase letter
• meaningful names
• Camel case
  – E.g. luckyNumber
Quiz

- Which of the following are legal identifiers?
  - Greeting1
  - g
  - void
  - 101dalmatians
  - Hello, World
  - <greeting>

- 2. Define a variable to hold **radius**. Use camel case in the variable name.
Declaring Variables - Examples

- int x;  // Declare x to be an integer variable;
- double radius; // Declare radius to be a double variable;
Syntax: Assignment

variableName = expression;

Example:
  luckyNumber = 12;
  luckyNumber = 5+7;

Purpose:
To assign a new value to a previously defined variable.
To be continued

• See you on Sept 17th!