CS3157: Advanced Programming

Lecture #6

Feb 13

Shlomo Hershkop
shlomo@cs.columbia.edu

Outline

- Wrapping up perl part of the course
  - Debugging
  - Object Oriented programming
  - Text processing
  - Network programming
  - Testing
- Intro to C
- If you are having issues with CGI, please see me asap.

Announcements

- HW1 is going to be due soon
- Office hours + email + IM
- Reading: please wrap up the perl reading and start on C (make sure you have a book by the end of the week and read (see class page for reading list)

Announcements

- Midterm Scheduled for March 1
  - I will post online notes for a review
    - What you need to know for the exam
  - Make sure you understand what we covered and what you did in the lab….you will be asked

Debugging process

- This is a general programming idea:
- We have some code instructions, would like to examine them as they run:

1. Output test cases for each line (hope it doesn't crash)
2. Run it within another program allowing us to fine tune control of running process and interaction with the running environment

Running perl debugger

- Can use a graphical based debugger
- Should learn to use the text based one…never know when it can come in usuafull (actually it will, once we start c…
- Perl –d nameofscript.pl
- Perl –d –e 0

What you see

- Current scope (main::)
- What line it will execute next
- No need for comma, ‘enter’ key is a signal

- Can do many things
  - Evaluate expressions
  - Check variables
  - Step through code

Common commands

- h
  - Get help
- x variable [, var2, var3..]
  - Examine a variable (or set)
- p
  - To print something
- V ???
  - Examine all variables in scope/package ?? (example main)
- s
  - Step through next instruction (including sub)
- n
  - Jump over subs

Objects

- Perl can be programmed in an object oriented fashion with objects/classes etc

Class declaration

- package Person;

- Will define the scope until the end of the current file

methods

- Accessor and modifier methods are defined in the current package
- Use arrow notation to access methods
- Variable => name of method()

```
sub print {
  my ($self) = @_;

  #print info
  print $self->firstName . " ". $self-
  >lastName;
  }
```

constructor

```
sub new {
   my $self = {

   _firstName => undef;
   _lastName => undef;

   };
   bless $self, ‘Person’;
   return $self;
   }
```

Instantiating

- my $shlomo = new Person();

accessing

```
sub firstName {
  my ( $self, $firstName) = @_;

  $self->{_firstName} = $firstName if
  defined ($firstName);

  return $self->{_firstName};
  }
```

Inheritance

- Classes can be used to extend the functionality of a class using a process called inheritance
- @ISA
  - perl keyword showing inheritance

Text handling

- One of the exciting developments in the last decade of computer science is data processing/mining/learning
- Many other area in and out of CS need data to be analyzed or presented in some (controlled but arbitrary fashion)

Handling data

- Using chiseled stone
- By hand (literally copy paste)
- Early mechanics (typwriters)
- Take 3157 ☺

Outputting text

- Many times will have multiple fields per line
- Arbitrary delimiters:
  - Comma
  - Tabs
  - Pipe |
- Make sure whatever you choose
  - Is either not/can't be present in the data
  - What if it is? How to represent these delimiters ??

Approach

- Memory vs disk based handling
- Brute force
- Divide and conquer
- Regexp is your friend

Ahead!

- Because CGI/Internet involves network based thinking, I will illustrate a quick example now.

Socket

- In order to communicate across computer networks (or between processes on the same computer)
- Need to decide on rules of communication
  - Language of protocol
  - Directional vs bi-directional
  - Throwaway or continuous
  - Life time of communication
  - Overhead
  - Priority
  - Location

IO:Socket client

```
Use IO::Socket::INET;

$socket = IO::Socket::INET->new(
   PeerAddr => $remote_host,
   PeerAddr => $remote_port,
   PeerAddr => “tcp”,
   PeerAddr => SOCK_STREAM) or die…
```

```
#writing out
print $socket “hello World”;

$answer = <socket>;

close($socket);
```

Server version

```
$server = IO::Socket::INET->new( LocalPort=>
   $server_port,
   Type => SOCK_STREAM,
   Reuse => 1,
   Listen = 10) or die….

while($client = $server->accept()) {
   #...
}
```

Other topics

- Multi threading
  - Fork processes
  - Process space
- Communication
  - Pipes
  - Sockets

Testing environment

- Many times need to test code

- Large projects
- Bugs cost time and money
- Bugs hurt morale
- Human are programmers…humans make mistakes

Automated testing

- Humans hate testing…
- Fast verification that new feature has not broken code
- Verify all code on a regular basis
- No grumble if test to rerun test ☺

packages

- There are packages out there (Test::Simple and Test::Harness) to automatically run tests
- Verify what happens on good/bad input
- Verify variables/method behavior
- Usually .t files have tests

Shift gears

- Will now start c component of the course

C Phase

- Intro to C
  - Background
  - *Compiling*
  - *Basic data structures*
  - *Basic I/O*
  - *Types conversion*
  - *Loops*
  - *Branching*

Roadmap

- How this all fits together
  - We covered perl (duct-tape programming)
  - CGI programming USING perl
  - Will now move to c, which is a more low level programming language
  - Will learn to work with c, and then CGI+c
  - Then CGI+perl+c etc
  - Get the best of any programming language in a project

Why Learn C ?

- C provides stronger control of low-level mechanisms such as memory allocation, specific memory locations
- C performance is usually better than Java and usually more predictable (very task dependant)

Why Learn c continued

- Java hides many details needed for writing code, but in C you need to be careful because:
  - memory management responsibility left to you
  - explicit initialization and error detection left to you
  - generally, more lines of (your) code for the same functionality
  - more room for you to make mistakes
- most older code is written in C, will need it if upgrading or interfacing

Background

C

– Dennis Ritchie in late 1960s and early 1970s
– systems programming language
– make OS portable across hardware platforms
– not necessarily for real applications—could be written in Fortran or PL/I

Background II

C++

– Bjarne Stroustrup (Bell Labs), 1980s
– object-oriented features

Java

– James Gosling in 1990s, originally for embedded systems
– object-oriented, like C++
– ideas and some syntax from C

Background III

- C is early-70s, procedural language
- C advantages:
  - direct access to OS primitives (system calls)
  - more control over memory
  - fewer library issues— just execute
- C disadvantages:
  - language is portable, but APIs are not
  - no easy graphics interface
  - more control over memory (i.e., memory leaks)
  - pre-processor can lead to obscure errors

C vs Java

- Java program
  - collection of classes
  - class containing main method is starting class
  - running java StartClass invokes StartClass.main method
  - JVM loads other classes as required
- C program
  - collection of functions
  - one function – main() – is starting function
  - running executable (default name a.out) starts main function
  - typically, single program with all user code linked in— but can be dynamic libraries (.dll, .so)

Example

- Java

```
public class hello {
    public static void main( String[] args ) {
        System.out.println( "hello world! " );
    }
}
```

- C

```
#include <stdio.h>
    int main() {
    printf( "hello world!" );
    return 0;
    }
```

- #include <stdio.h> to include header file stdio.h
- # lines processed by pre-processor

- No semicolon at end of pre-processor lines
- Lower-case letters only— C is case-sensitive

- int main() { ... } is the only code executed

- printf(" /* message you want printed */ ");

- \n = newline, \t = tab

- \ in front of other special characters

C vs Java …Running

- Java programs are compiled and interpreted:
  - javac converts foo.java into foo.class
  - class file is not machine-specific— it is byte code
  - byte code is then interpreted by JVM
  - and each JVM is machine-specific

- C programs are compiled into object code and then linked into executables

(to allow for multiple object files and libraries to be compiled together into one program):

  - gcc compiles foo.c into foo.o and then links foo.o into a.out
  - you can skip writing foo.o if there is only one object file used to create your executable
  - a.out is executed by OS and hardware
  - the C compiler is machine-specific, creating code that executes on specific OS/hardware

For next time:

- Lab on Wednesday

- Start reading c book