RUBY: Input-Output and File Handling

Dheren Gala

November 29, 2004

About Ruby

- Ruby is a scripting language like Perl.
- It is Object Oriented like C++.
- Programs close to problem domain; not obscured by syntax.
- Everything is a class or a method in some class.
- We look at the classes and methods related to I/O and file handling in Ruby.

The puts method

- Outputs its argument followed by a newline.

Example:

```
gets “Hello World!”
gets “Ruby”
```

Output:

```
Hello World!
Ruby
```

The print method

- Same as the puts statement except that it does not print a newline at the end.

Example:

```
print “Hello World! ”
gets “Ruby”
```

Output:

```
Hello World! Ruby
```

The printf method

- Prints its arguments under the control of a format string (like printf in C).

Example:
printf “Number: %5.2f, String: %s”, 1.23, “hello”

Output:
Number: 1.23, String: hello

The gets method

- Used for reading input into the program.
- Similar to the puts method.

Example:

```
line = gets
print line
```

The puts method contd.

- Has the side effect of storing the line just read into the global variable $_.
- $_ can be used in many circumstances. For eg.
  - It can be used as a default argument in the print method.
  - If the condition in an if or a while statement contains only a regular expression, the expression is matched with $_.

Example:

```
while gets     # assigns line to $_
  if /Ruby/    # matches against $_
   print       # prints $_
  end
end
```

Other Simple I/O methods

- open
- putc
- readline
- readlines

An IO Object

- A single base class IO handles input and output.
- Two subclasses to provide specialized behavior:
  - File
  - BasicSocket
- An IO object is a bidirectional channel between a Ruby program and some external resource.

Opening and Closing Files

- A new file object can be created using File.new.
- The mode of file (reading, writing or both) is specified using the mode string.
- After opening the file we can read/write data as needed.
- Finally, the file must be closed to ensure that all buffered data is written and that all related resources are freed.

Example:

```
aFile = File.new("testfile", "r")

# ... process the file

aFile.close
```

Reading Files

- The methods used for simple I/O are also available for file objects.
- For eg., Like gets reads a line from standard input afile.gets reads a line from the file object.
- I/O objects also have additional set of access methods (explained in next slide).

IO#each_byte

- As well as using the usual loops to read data from an IO stream, we can also use various Ruby iterators.
- For eg., the iterator IO#each_byte invokes a block with the next 8-bit byte from the IO object.

Example:

```
aFile = File.new(“testfile”)
aFile.each_byte { |ch| putc ch; putc ?. }
```

Output:

```
T.h.i.s. .i.s. .l.i.n.e. .o.n.e.
.T.h.i.s. .i.s. .l.i.n.e. .t.w.o.
.T.h.i.s. .i.s. .l.i.n.e. .t.h.r.e.e.
.A.n.d. .s.o. .o.n.......
```

.

IO#each_line

- The IO#each_line method calls the block with the next line from the file.

Example:

```
aFile.each_line {|line| puts “Got #{line.dump}” }
```

Output:

```
Got “This is line one\n”
Got “This is line two\n”
Got “This is line three\n”
Got “And so on...\n”
```

IO.foreach

- The IO.foreach method takes the name of an I/O source, opens it for reading, calls the iterator once for every line in the file, and then closes the file automatically.

Example

```
IO.foreach(“testfile”) { |line| puts line }
```

Output

```
This is line one
This is line two
This is line three
And so on...
```

Writing to files

- Writing to file done using the IO.print method.
- Objects that aren't strings are converted by calling their to_s method.

Example:

```
$stdout.print(“This is”, 100, “percent.\n”)
```

Output:

```
This is 100 percent.
```

Other methods from the IO class

Some of the other important methods in the IO class are listed below:

- select
- eof
- fcntl
- flush
- lineno
- pos
- seek
- stat
- sysread