

```

Jul 30, 04 15:24      Is-IR.rb      Page 1/8
1  #!/usr/bin/env ruby
2
3  # $Id: ls-IR.rb 24 2004-07-30 06:24:36Z shugo $
4  # <URL:http://svn.shugo.net/src/llw2004/trunk/ls-IR.rb>
5
6  require "shellwords"
7
8  module LsIR
9    class Node
10     attr_reader :line, :parent, :name, :size
11
12     def initialize(line, parent, name, size)
13       @line = line
14       @parent = parent
15       @name = name
16       @size = size
17     end
18
19     def to_s
20       return @line
21     end
22
23     def path
24       if parent.nil?
25         return "/"
26       end
27       return File.expand_path(name, parent.path)
28     end
29
30     def relative_path(base)
31       if base == self
32         return "."
33       end
34       return File.join(parent.relative_path(base), name)
35     end
36
37     def directory?
38       return false
39     end
40
41     def accept(visitor)
42       raise ScriptError, "subclass must override Node#accept"
43     end
44
45     class FileNode < Node
46       def type
47         return "f"
48       end
49
50       def accept(visitor)
51         visitor.visit_file(self)
52       end
53     end
54
55     class DirectoryNode < Node
56       attr_reader :children
57
58       def initialize(line, parent, name, size)
59         super
60         @children = []
61       end
62
63       def type
64         return "d"
65       end
66
67       def get_child(name)
68         child = children.detect { |child| child.name == name }
69

```

```

Jul 30, 04 15:24      Is-IR.rb      Page 2/8
70     if child.nil?
71       raise format("no such file or directory - %s", name)
72     end
73     return child
74   end
75
76   def get_descendant(path)
77     first, rest = path.split("/", 2)
78     child = get_child(first)
79     if rest.nil?
80       return child
81     else
82       return child.get_descendant(rest)
83     end
84   end
85
86   def directory?
87     return true
88   end
89
90   def accept(visitor)
91     visitor.visit_directory(self)
92   end
93 end
94
95 class Parser
96   NODE_CLASSES = {
97     "f" => FileNode,
98     "d" => DirectoryNode
99   }
100
101   def initialize
102     @input = nil
103     @directories = nil
104     @root_directory = nil
105   end
106
107   def parse(input)
108     @input = input
109     @directories = {}
110     @root_directory = nil
111     loop do
112       begin
113         parse_files
114       rescue EOFError
115         break
116       end
117     end
118     return @root_directory
119   end
120
121   private
122
123   def parse_files
124     line = @input.readline
125     dirname = line.slice(/(.*)\n/, 1)
126     if @directories.key?(dirname)
127       dir = @directories[dirname]
128     else
129       dir = @root_directory =
130         DirectoryNode.new("", nil, "", 0)
131     end
132     total = @input.readline
133     @input.each_line do |line|
134       line.chomp!
135       break if line.empty?
136       file = parse_file(line, dir)
137       dir.children.push(file)
138       if file.directory?

```

```

Jul 30, 04 15:24      Is-IR.rb      Page 3/8
139     @directories[File.join(dirname, file.name)] = file
140   end
141 end
142 @directories.delete(dirname)
143 end
144
145 def parse_file(line, parent)
146   m = /^(.*)\S+\s+d+\s+w+\s+w+\s+(d+)\s+.[12]\s+(.*)/n.match(line)
147   type = m[1].tr("-", "f")
148   size = m[2].to_i
149   name = m[3]
150   file = NODE_CLASSES[type].new(line, parent, name, size)
151 end
152
153
154 class Command
155   attr_reader :shell
156
157   def initialize(shell)
158     @shell = shell
159   end
160
161   def exec(args)
162     raise ScriptError, "subclass must override Command#exec"
163   end
164
165
166 class NullCommand < Command
167   def exec(args)
168     raise "no such command"
169   end
170 end
171
172 class QuitCommand < Command
173   def exec(args)
174     exit
175   end
176 end
177
178 class PwdCommand < Command
179   def exec(args)
180     shell.print(shell.current_directory.path, "\n")
181   end
182 end
183
184 class CdCommand < Command
185   def exec(args)
186     if args.length < 1
187       return
188     end
189     path = args[0]
190     dir = shell.get_file(path)
191     unless dir.directory?
192       raise format("not a directory - %s\n", path)
193     end
194     shell.current_directory = dir
195   end
196 end
197
198 class LsCommand < Command
199   def exec(args)
200     shell.current_directory.children.each do |file|
201       shell.print(file, "\n")
202     end
203   end
204 end
205
206 class Expression
207   def evaluate(file)

```

```

Jul 30, 04 15:24      Is-IR.rb      Page 4/8
208     raise ScriptError, "subclass must override Expression#evaluate"
209   end
210
211   def null?
212     return false
213   end
214
215   class NullExpression
216     def evaluate(file)
217       return true
218     end
219
220     def null?
221       return true
222     end
223   end
224
225
226 class SimpleExpression < Expression
227   def initialize(value)
228     @value = value
229   end
230 end
231
232 class NameExpression < SimpleExpression
233   def evaluate(file)
234     return File.fnmatch(@value, file.name)
235   end
236 end
237
238 class TypeExpression < SimpleExpression
239   def evaluate(file)
240     return file.type == @value
241   end
242 end
243
244 class SizeEqExpression < SimpleExpression
245   def evaluate(file)
246     return file.size == @value
247   end
248 end
249
250 class SizeLtExpression < SimpleExpression
251   def evaluate(file)
252     return file.size < @value
253   end
254 end
255
256 class SizeGtExpression < SimpleExpression
257   def evaluate(file)
258     return file.size > @value
259   end
260 end
261
262 class NotExpression < Expression
263   def initialize(expression)
264     @expression = expression
265   end
266
267   def evaluate(file)
268     return !@expression.evaluate(file)
269   end
270 end
271
272 class BinaryExpression < Expression
273   def initialize(left, right)
274     @left = left
275     @right = right
276   end

```

```

Jul 30, 04 15:24      Is-IR.rb      Page 5/8
277 end
278
279 class AndExpression < BinaryExpression
280   def evaluate(file)
281     return @left.evaluate(file) && @right.evaluate(file)
282   end
283 end
284
285 class OrExpression < BinaryExpression
286   def evaluate(file)
287     return @left.evaluate(file) || @right.evaluate(file)
288   end
289 end
290
291 class FindExpressionParser
292   def initialize
293     @tokens = nil
294   end
295
296   def parse(tokens)
297     @tokens = tokens.dup
298     return exprs
299   end
300
301   private
302
303   def exprs
304     result = NullExpression.new
305     loop do
306       token = lookahead
307       if token.nil? || token == ")"
308         break
309       end
310       e = or_expr
311       if result.nil?
312         result = e
313       else
314         result = AndExpression.new(result, e)
315       end
316     end
317     return result
318   end
319
320   def or_expr
321     result = and_expr
322     while lookahead == "-o"
323       shift_token
324       right = and_expr
325       result = OrExpression.new(result, right)
326     end
327     return result
328   end
329
330   def and_expr
331     result = not_expr
332     while lookahead == "-a"
333       shift_token
334       right = not_expr
335       result = AndExpression.new(result, right)
336     end
337     return result
338   end
339
340   def not_expr
341     if @tokens.first == "!"
342       shift_token
343       return NotExpression.new(not_expr)
344     else
345       return primary_expr

```

```

Jul 30, 04 15:24      Is-IR.rb      Page 6/8
346 end
347
348 def primary_expr
349   case lookahead
350   when "-name"
351     return name_expr
352   when "-type"
353     return type_expr
354   when "-size"
355     return size_expr
356   when "("
357     return paren_expr
358   else
359     raise format("unknown expression - %s", lookahead)
360   end
361 end
362
363 def name_expr
364   shift_token
365   val = shift_token
366   return NameExpression.new(val)
367 end
368
369 def type_expr
370   shift_token
371   val = shift_token
372   if !/\A[\d]\z/n.match(val)
373     raise format("invalid argument for -type - %s", val)
374   end
375   return TypeExpression.new(val)
376 end
377
378 SIZE_EXPRESSION_CLASSES = {
379   nil => SizeEqExpression,
380   "+" => SizeGtExpression,
381   "-" => SizeLtExpression
382 }
383
384 def size_expr
385   shift_token
386   val = shift_token
387   m = /\A[\+|-]?(\d+)\z/n.match(val)
388   if m.nil?
389     raise format("invalid argument for -size - %s", val)
390   end
391   return SIZE_EXPRESSION_CLASSES[m[1]].new(m[2].to_i)
392 end
393
394 def paren_expr
395   shift_token
396   result = exprs
397   match("(")
398   return result
399 end
400
401 def lookahead
402   return @tokens.first
403 end
404
405 def shift_token
406   return @tokens.shift
407 end
408
409 def match(pat)
410   token = shift_token
411   unless pat === token
412     raise format("invalid token - %s (expected %s)", token, pat)
413   end
414 end

```

```

Jul 30, 04 15:24      Is-IR.rb      Page 7/8
415   return token
416 end
417
418 class DfsScheduler
419   def initialize
420     @stack = []
421   end
422
423   def next_node
424     return @stack.pop
425   end
426
427   def schedule(nodes)
428     @stack.concat(nodes.reverse)
429   end
430
431 end
432
433 class BfsScheduler
434   def initialize
435     @queue = []
436   end
437
438   def next_node
439     return @queue.shift
440   end
441
442   def schedule(nodes)
443     @queue.concat(nodes)
444   end
445
446 end
447
448 class SearchCommand < Command
449   def initialize(shell, scheduler)
450     super(shell)
451     @expr_parser = FindExpressionParser.new
452     @scheduler = scheduler
453   end
454
455   def exec(args)
456     @expr = @expr_parser.parse(args)
457     @scheduler.schedule([@shell.current_directory])
458     while node = @scheduler.next_node
459       node.accept(self)
460     end
461
462   def visit_file(file)
463     if @expr.evaluate(file)
464       shell.print(file.relative_path(@shell.current_directory), "\n")
465     end
466
467   def visit_directory(dir)
468     visit_file(dir)
469     @scheduler.schedule(dir.children)
470   end
471 end
472
473 class Shell
474   attr_reader :root_directory
475   attr_accessor :current_directory
476
477   def initialize
478     @current_directory = nil
479     null_cmd = NullCommand.new(self)
480     @commands = Hash.new(null_cmd)
481     @commands["quit"] = @commands["exit"] = QuitCommand.new(self)
482     @commands["pwd"] = PwdCommand.new(self)

```

```

Jul 30, 04 15:24      Is-IR.rb      Page 8/8
484   @commands["cd"] = CdCommand.new(self)
485   @commands["ls"] = LsCommand.new(self)
486   dfs_scheduler = DfsScheduler.new
487   @commands["dfs"] = SearchCommand.new(self, dfs_scheduler)
488   bfs_scheduler = BfsScheduler.new
489   @commands["bfs"] = SearchCommand.new(self, bfs_scheduler)
490 end
491
492 def run(filename)
493   parser = Parser.new
494   @current_directory = @root_directory =
495     open(filename) { |f| parser.parse(f) }
496   each_cmd do |cmd, *args|
497     begin
498       @commands[cmd].exec(args)
499     rescue
500       print($!, "\n")
501     end
502   end
503 end
504
505 def get_file(path)
506   abs_path = File.expand_path(path, current_directory.path)
507   return root_directory if abs_path == "/"
508   rel_path = abs_path.sub(/^\//n, "")
509   return root_directory.get_descendant(rel_path)
510 end
511
512 def print(*args)
513   $stdout.print(*args)
514 end
515
516 private
517
518 def each_cmd
519   while line = readline
520     yield(Shellwords.shellwords(line))
521   end
522 end
523
524 def readline
525   if $stdin.tty?
526     $stdout.print("ls-lR> ")
527     $stdout.flush
528   end
529   return $stdin.gets
530 end
531
532 if $0 == __FILE__
533   if ARGV.length < 1
534     STDERR.printf("usage: %s <ls-lR file>\n", $0)
535     exit(1)
536   end
537   filename = ARGV.shift
538   shell = LsLR::Shell.new
539   shell.run(filename)
540 end

```