Brian Cullinan

Compilers

Project 1: SourceReader and CompileError classes

9/3/08

I went into this assignment with the assumption that it would be fairly easy being the first assignment. I had initially planned on writing a wrapper for PHPs global file operation functions. I soon discovered that would not be a possibility considering PHP does not have peek functionality. This complicated the problem. I then realized I could easily solve the problem by loading a buffered amount in to the reader, and then adding 1 character read to the end of the buffer every time the function is called. The function would return the first character off the front of the buffer, so the buffer would always be large enough for peek operations.

The CompileError class could use PHPs error fire functions, but there isn’t actually a class for errors already built in. So my class is built from the ground up. It uses the reader class to obtain the line information, so that the space and character replacement functionality only has to be programmed in once.

The driver simple prints out some test data, and can be seen at the following address:

<http://209.250.30.30/compiler/index.php>

Index.php (Driver)

<code>

<pre>

<?php

// include some needed classes

require\_once 'reader.php';

require\_once 'error.php';

// create a reader object

$reader = new SourceReader('reader.php');

// show the content of the source file

print 'Here is the entire source file: ' . "\n";

while(true)

{

$int = $reader->read();

if($int == -1)

break;

print htmlspecialchars(chr($int));

}

print "\n";

// this is the end of the file (the last positon)

print 'The current position is: ' . "\n";

print\_r($reader->pos());

// create a sample error using the CompileError class

print 'Here is a sample error: ' . "\n";

$error = new CompileError('reader.php', 57, 33, 'You obviously don\'t know how to program');

print $error->error\_query();

?>

</pre>

</code>

Error.php (CompileError)

<?php

require\_once 'reader.php';

class CompileError

{

var $file;

var $line;

var $col;

var $msg;

function CompileError($file, $line, $col, $msg)

{

$this->file = $file;

$this->line = $line;

$this->col = $col;

$this->msg = $msg;

}

function query()

{

return 'line ' . $this->line . ', col ' . $this->col;

}

function line\_query()

{

// open the file and get the specified line

$reader = new SourceReader($this->file);

while($reader->line < $this->line)

{

$int = $reader->read();

if($int == -1)

break;

}

$line = '';

while($reader->line == $this->line)

{

$int = $reader->read();

if($int == -1)

break;

$line .= chr($int);

}

return $line . sprintf('%' . ($this->col - 1) . 's', '') . '^';

}

function error\_query()

{

return 'At ' . $this->query() . ': ' . $this->msg . "\n\n" . $this->line\_query() . "\n";

}

}

?>

Reader.php (SourceReader)

<?php

class SourceReader

{

var $file;

var $fp;

var $fpos = 0;

var $buffer;

const buffer\_length = 2048;

var $col = 1;

var $line = 1;

function SourceReader($file)

{

// store file name/path

$this->file = $file;

// open the file for reading

$this->fp = fopen($this->file, 'r');

}

function read()

{

if(!isset($this->fp))

return -1;

if(!feof($this->fp) || strlen($this->buffer) > 0)

{

// read a lot of information from the file

if(!isset($this->buffer))

$this->buffer = fread($this->fp, SourceReader::buffer\_length);

// only read 1 character

else

$this->buffer .= fread($this->fp, 1);

// return the first character on the buffer

$return = $this->buffer[0];

$this->buffer = substr($this->buffer, 1);

// special cases for modifying other things

switch($return)

{

case chr(10):

$this->line++;

$this->col = 1;

break;

case chr(13):

$this->col = 1;

break;

case chr(9):

$return = ' ';

$this->buffer = ' ' . $this->buffer;

break;

default:

$this->col++;

}

return ord($return);

}

else

{

fclose($this->fp);

unset($this->fp);

return -1;

}

}

function peek($offset)

{

if(!isset($this->fp));

return -1;

// return the character at the specified position without modifying the buffer

return ord($this->buffer[$offset - 1]);

}

function pos()

{

return array('line' => $this->line, 'col' => $this->col);

}

}

?>