SCL: Site Construction Language

Sudip Das, Clark Landis,
Mohamed Nasser

COMS W4115
Programming Languages and Translators
Columbia University
May 13, 2003

What is SCL?

- A scripting language for building websites
  - Efficiently and intelligently merges
    - Text
    - HTML
    - Graphics
    - Executable CGI scripts
  - Automates index generation
  - Simple yet Powerful
  - Integrates well with Unix

What SCL is not

- GUI-based web design software
  - Time-consuming to construct an entire website
  - Difficult to integrate with dynamic files
- Server side scripting (PHP, Perl + libraries)
  - These generate pages on the fly
  - Not as efficient as SCL’s precompiled pages
- SCL works best when combined with the above

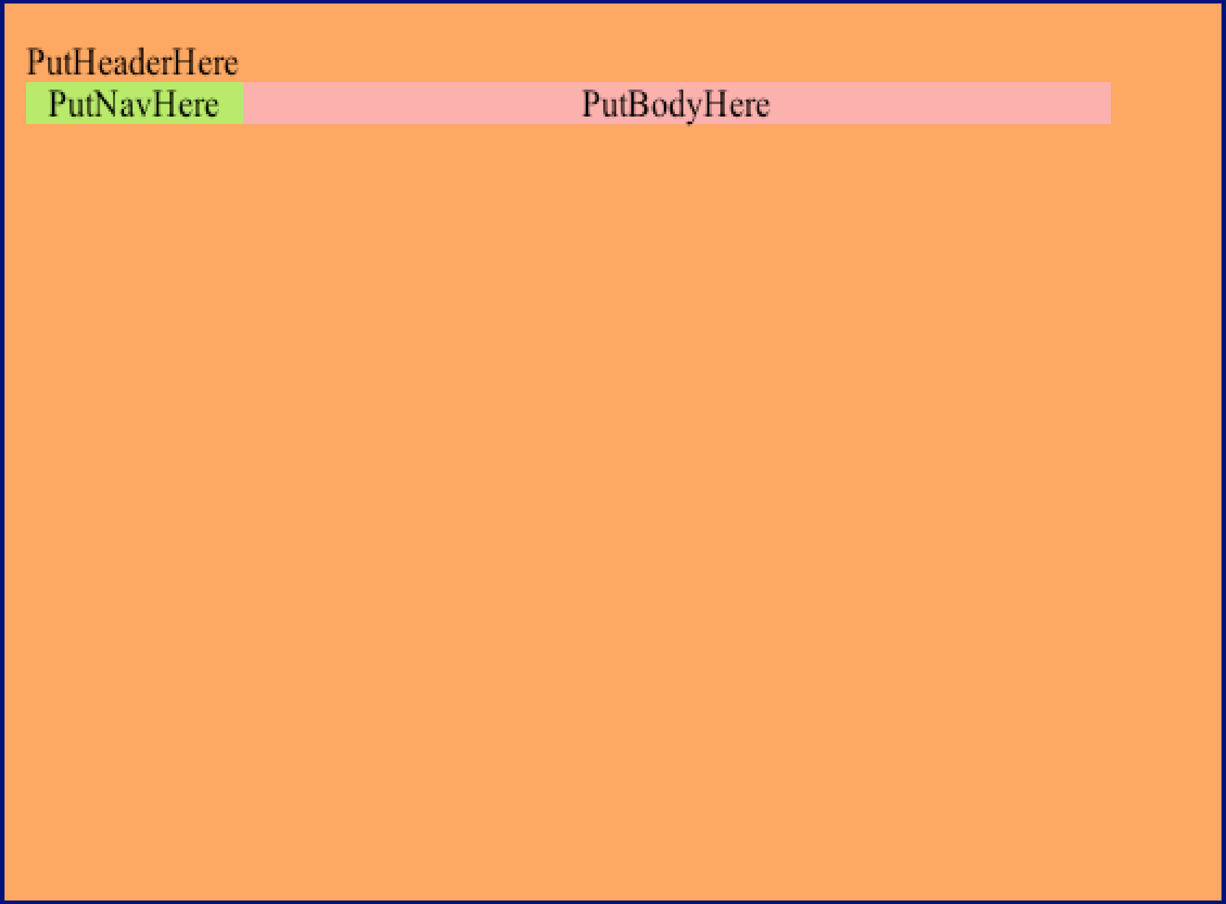
SCL Program Components

- Template
  - an HTML skeleton file
- Data files
  - text, HTML, graphics, CGI, etc.
- SCL file
  - code written in SCL language

Example: Template

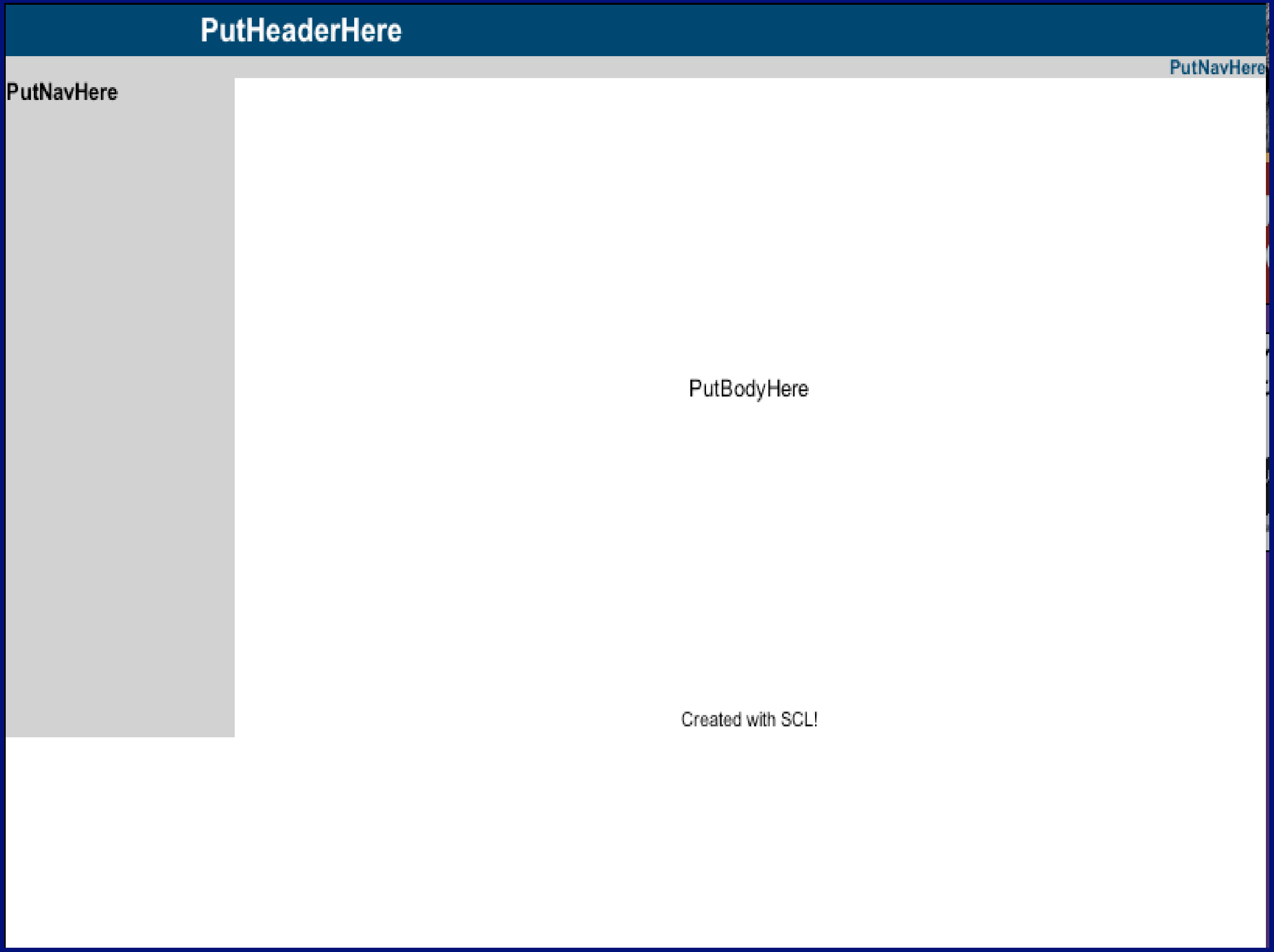
```
<HTML><HEAD></HEAD><TITLE></TITLE>
<BODY bgColor=#ff9933 >
   PutHeaderHere
 <TABLE cellspacing=0 cellpadding=0 border=0  >
  <TR align=center>
   <TD width=100 bgColor=#99ff33>
    PutNavHere
   </TD>
   <TD width=400 bgColor=#ff9999>
    PutBodyHere
   </TD>
  </TR>
 </TABLE>
</BODY>
```

Example: Template


PutHeaderHere

PutNavHere PutBodyHere

Example: Template


PutHeaderHere

PutNavHere

PutNavHere

PutBodyHere

Created with SCL!

Bindings

- Associates a data file with each placeholder in the template

```
bind mybindings {
    PutHeaderHere : "header.html"  ;
    PutBodyHere : "body.jpg"  ;
    PutNavHere : "nav.html" : SSI ;
}
```

Makepage

- Takes the template and “smartly” inserts the substitutions specified in binding

```
makepage(“template.html”,mybindings)
```

- HTML files are copied in
- Text / code files are translated to HTML & copied in
- Image files are linked with <IMG> tag

Other built-in functions

- Link: add a link to a file
- Read: copy contents of file to a variable
- Write: write a string to
  - a variable
  - a file
- Writepage: makepage + write

Foreach

- Iterates over each element of a list

```
foreach $file in "messy.jpg halloween.jpg bath.jpg
      intro.html todo.html"
{
  bind mybindings {  PutBodyHere   : $file ; }
  writepage($mytemplate,mybindings,$file.".shtml");
}
```

Variables

- All variables are strings
  - Can be treated like numbers, e.g. math
- No declarations
  - Variables have default value “null”
- Identifier preceded by a $, e.g. $foo
- Dynamic Scope
  - defined in inner scope => not seen in outer
  - defined in outer, changed in inner => changed in outer

User-Defined Functions

- Function declaration

```
function #foot ($name){
        $Return="Foot of ".$name;
}
```

  – accepts one variable as input
  – $Return is the string returned by the function

- Function reference

```
#foot("Clark");
```

- Can define functions inside functions
- Can do recursion

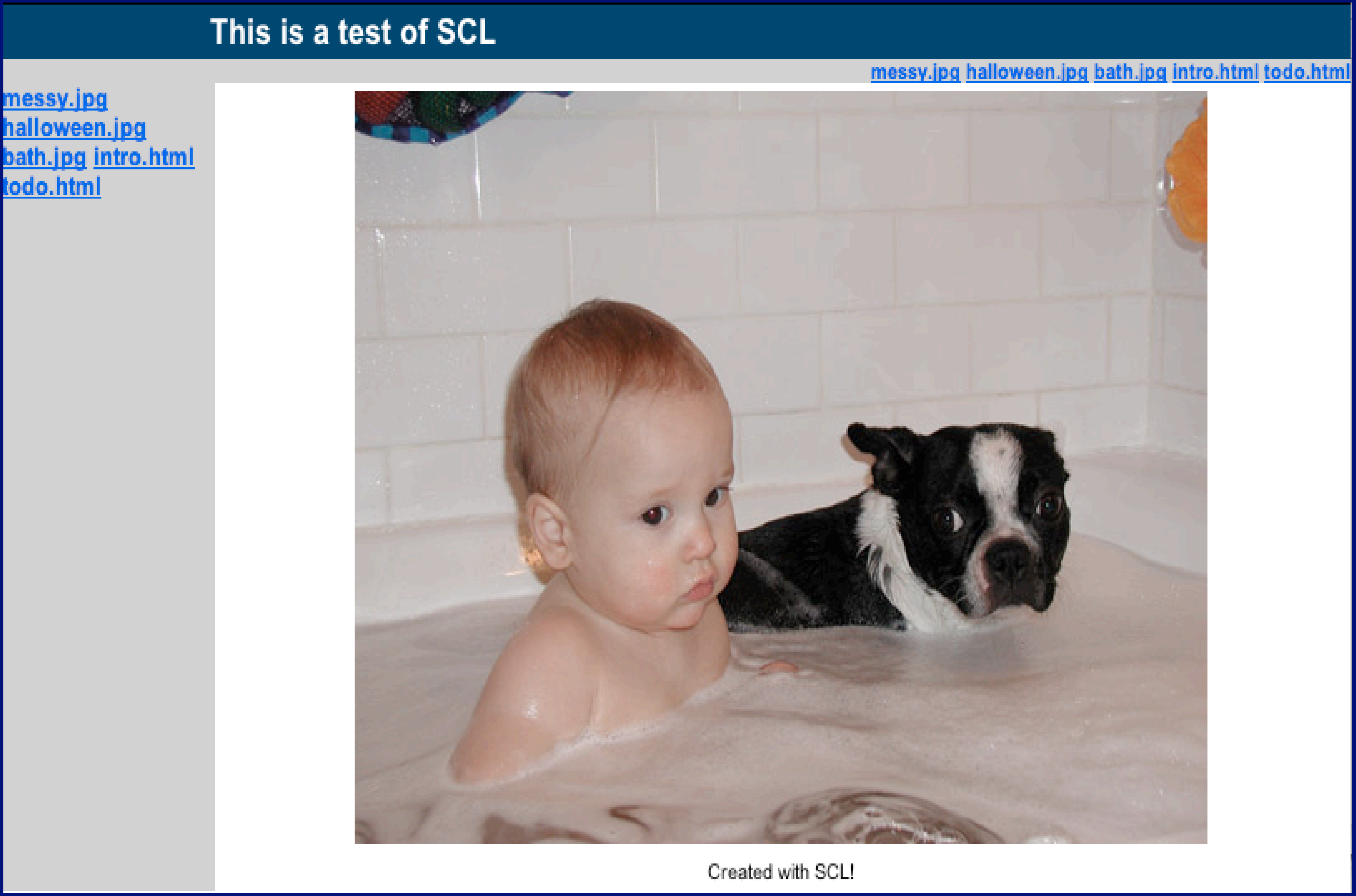
Example Code

```
$mytemplate=read("template.html");

bind mybindings {
   PutHeaderHere : "header.html"  ;
   PutNavHere : "nav.html" : SSI;
}

write ("","nav.html");
foreach $file in "messy.jpg halloween.jpg bath.jpg
intro.html todo.html"
{
    bind mybindings {  PutBodyHere   : $file ; }
    writepage($mytemplate,mybindings,$file.".shtml");
    link ($LastPageLink,$file,"nav.html");
}
writepage($mytemplate,mybindings,"index.shtml");
```

Output of Example Code



This is a test of SCL

messy.jpg halloween.jpg bath.jpg intro.html todo.html

messy.jpg
halloween.jpg
bath.jpg intro.html
todo.html

Created with SCL!

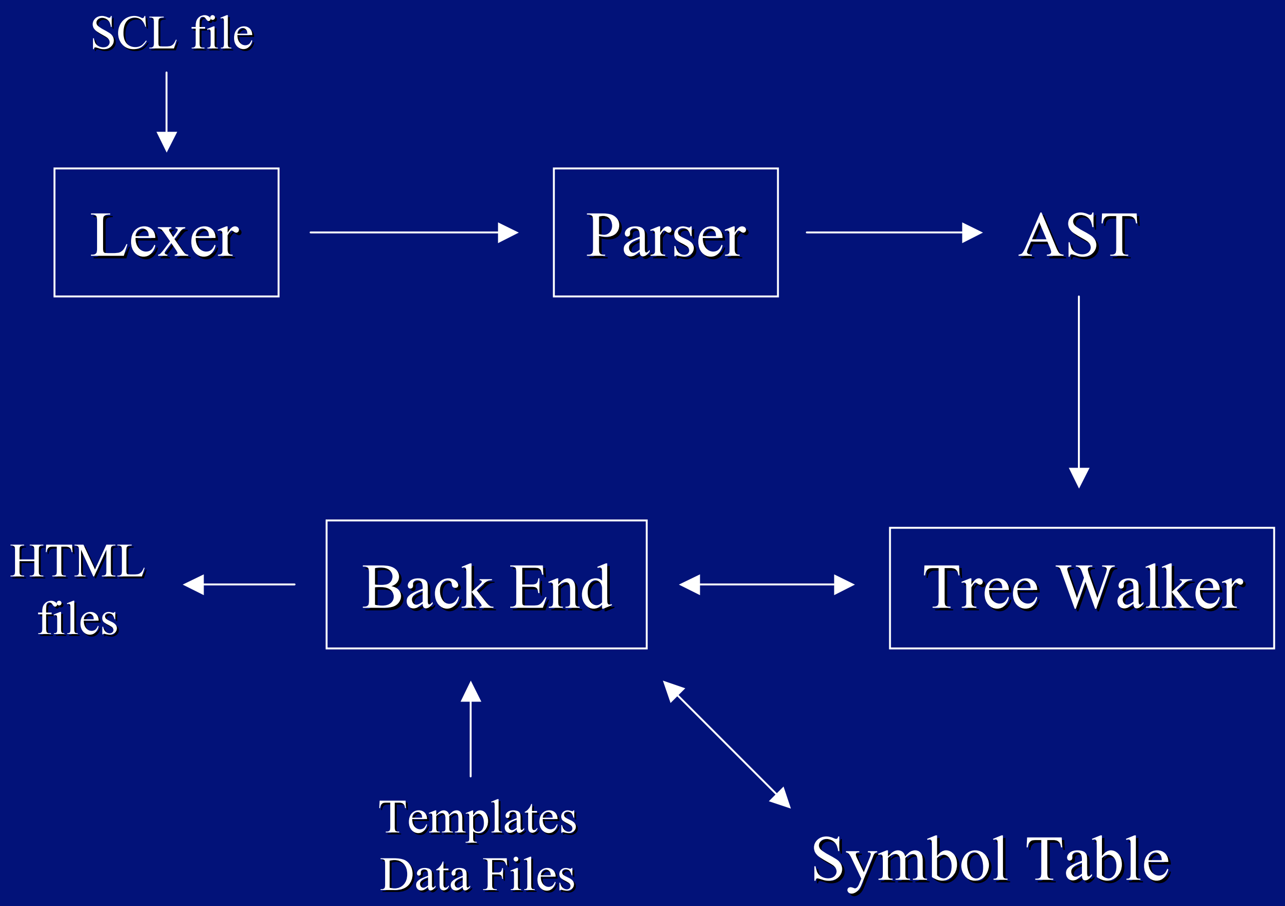
Example Code: Scoping

```
function #foot ($name){
    $Return="Foot of ".$name;
}

function #dog ( $name ) {
   function #foot ( $name ) {
       $x="Hello from foot of dog!";
       $Return = $name." is a paw.  ".#toe($name."'s toe");
   }
   $Return=$name." is a Dog. ".#foot($name."'s foot");
}

echo(#dog("Neutron"));
echo($x);
$x="Hello from OuterScope";
echo($x);
echo(#dog("Electra"));
echo($x);
echo(#foot("Clark"));
```

Compiler Architecture



Compiler Implementation

- ANTLR Java Parser Generator
  - SCLLexer
  - SCLParser
  - SCLTreeWalker
- Other Java Classes
  - SCLBE (Back End)
  - SCL (executes the compiler)
    - java SCL filename.scl

Compiler Output

- No code is generated
  - The SCL file is interpreted
- The output is a collection of HTML files

Demo

Testing

- Test suite
  - set of tests that run over every line of code
  - using simple scripts
- After each update of the source code...
  - run the tests as they are
    - should return the same values
  - add .scl files to suite, to test new features
    - should return same values + results of new test

Testing

- What was tested
  - basic language constructs
    - statements
    - function calls
    - simple commands
  - page generation
  - other subcategories
    - new things added
    - complex commands

Lessons Learned

- Division of Labor
  - worked well, even with only 3 people
    - coder
    - tester
    - documenter
- Keep it simple
- Read documentation carefully to avoid frustration, e.g. with ANTLR

Acknowledgements

- Peter Palfrader - code2html
- Seth Doe - txt2html
- Terrence Parr - ANTLR
- J.S. Mills - ANTLR Tutorial

- Prof. Edwards
- Peter Davis