

SPL Feature Show

Clifford Wolf

SPL - <http://www.clifford.at/>

Csync2 - <http://oss.linbit.com/csync2/>

ROCK Linux - <http://www.rocklinux.org/>



Introduction

- Kickstart
- What is SPL
- Language Feature Overview
- Module Overview

Language Features

Module Features

References

Introduction



Kickstart

Introduction

● Kickstart

- What is SPL
- Language Feature Overview
- Module Overview

Language Features

Module Features

References

Let's start with a simple example:

```
for (var d,i=<obfuscated>just</obfuscated>,j=function
){d~=i~(defined(i=next[*],i)?" ":"");},just,another
SPL,hacker;defined i||({write("$d\n");return;});j())
```

```
$ splrun kickstart.spl
just another SPL hacker
```



What is SPL

Introduction

● Kickstart

● What is SPL

● Language Feature Overview

● Module Overview

Language Features

Module Features

References

- SPL is a powerful scripting language
- Small runtime (<200kB including compiler)
- Easy C-API and easy to embedd
- C-like syntax - Easy to learn
- Statefull - The VM state can be dumped and resumed
- Support for precompiled bytecode
- Well documented - manual with about 180 A4 pages



Language Feature Overview

Introduction

- Kickstart
- What is SPL

● Language Feature Overview

- Module Overview

Language Features

Module Features

References

- Dynamically typed
- Arrays, Hashes, Complex data types
- Flexible function arguments passing
- Real OO programming
- Builtin localization support
- String substitutions and regular expressions
- Powerful and extendable template language



Module Overview

Introduction

- Kickstart
- What is SPL
- Language Feature Overview
- **Module Overview**

Language Features

Module Features

References

- Builtins library with some basic math and string functions
- Arrays, Bit-Manupulation, File-I/O, Time handling
- CGI support and two Web Application Frameworks
- Complete Qt and KDE bindings
- Powerful XML module
- SQL Support (SQLite, MySQL and PostgreSQL)
- OpenGL, FANN, ...



Language Features

[Introduction](#)

[Language Features](#)

- Operators
- Variables (1/2)
- Variables (1/2)
- OO Programming
- Advanced Functions
- String Substitutions
- Templates
- Localization
- Regular Expressions

[Module Features](#)

[References](#)



Operators

Introduction

Language Features

● Operators

- Variables (1/2)
- Variables (1/2)
- OO Programming
- Advanced Functions
- String Substitutions
- Templates
- Localization
- Regular Expressions

Module Features

References

■ Dynamically typed operators:

+ - * / % ** ...

■ Integer operators:

+ # - # * # / # % # ** ...

■ Floating point operators:

. + . - . * . / . % . ** ...

■ Object operators:

(+) (-) (*) (/) (%) (**) ...

■ Casting operators:

(. . .) . (. . .)



Variables (1/2)

Introduction

Language Features

● Operators

● Variables (1/2)

● Variables (1/2)

● OO Programming

● Advanced Functions

● String Substitutions

● Templates

● Localization

● Regular Expressions

Module Features

References

- Only one extrem flexible variable type.
- Looks a bit like a hash with ordering for the elements and an optional scalar value.
- In fact there are much more optional fields (code pointer, context pointer, class pointer, ...)
- Member variables are accessed using `var[42]` or `var.foobar`.
- There are `foreach` and `foreach[]` loops.
- As well as special `next` and `prev` keywords.



Variables (1/2)

Introduction

Language Features

- Operators
- Variables (1/2)
- Variables (1/2)
- OO Programming
- Advanced Functions
- String Substitutions
- Templates
- Localization
- Regular Expressions

Module Features

References

- Arrays, hashes and structured can be defined using the `[...]` operator:

```
var a1 = [ "a", "b", "c" ];
```

```
var a2 = [ "x" => "a", "y" => "b", z: "c" ];
```

```
var a3 = [ x: "a", 100 => "b", "c" ];
```

- Scalar variables are passed and copied by value, complex variables are passed by reference.
- The `:=` operator can be used to create recursive copies.



OO Programming

Introduction

Language Features

- Operators
- Variables (1/2)
- Variables (1/2)
- OO Programming
- Advanced Functions
- String Substitutions
- Templates
- Localization
- Regular Expressions

Module Features

References

- SPL supports real OO programming:
 - ◆ Inheritance and multiple inheritance
 - ◆ Member functions and member variables
 - ◆ Operator overloading, Type classes, Closures
- Objects and classes are almost the same in SPL
- Objects are like read-write derives of classes

```
object Value {  
    var value = 42;  
}
```

```
object GetsetValue Value {  
    method get() { return value; }  
    method set(val) { value=val; }  
}
```



Advanced Functions

Introduction

Language Features

- Operators
- Variables (1/2)
- Variables (1/2)
- OO Programming
- Advanced Functions
- String Substitutions
- Templates
- Localization
- Regular Expressions

Module Features

References

```
// example_advfunc.spl
```

```
function a(prefix, @args, %opts) {  
    foreach[] i (args)  
        debug "$prefix: $i -> ${opts[i]}";  
}
```

```
function b(@args, %opts) {  
    a("example", @args, "a", a: "test", %opts);  
}
```

```
var opts = [ "hello" => "Sol-3",  
             "this" => "is", "a" => "demo" ];
```

```
b("hello", "this", hello: "world", %opts,  
   foo: "bar", a: "program");
```



String Substitutions

Introduction

Language Features

- Operators
- Variables (1/2)
- Variables (1/2)
- OO Programming
- Advanced Functions

● String Substitutions

- Templates
- Localization
- Regular Expressions

Module Features

References

All string substitutions start with the dollar sign.
Some examples:

```
"The value of variable 'xyz' is $xyz."
```

```
"The answer is ${40 + 2} and always has been."
```

```
"foo $( var x = "SQL"; x =~ s/Q/P/; return x; ) bar"
```

```
"little $[ this is a comment ] demo"
```

Strings can be quoted using single or double quotes. In both styles dollar substitutions and backslash sequences are supported.



Templates

Introduction

Language Features

- Operators
- Variables (1/2)
- Variables (1/2)
- OO Programming
- Advanced Functions
- String Substitutions
- **Templates**
- Localization
- Regular Expressions

Module Features

References

A template is a special style of here document with some nice additional feature. Example given:

```
write(<:foobar>
    : This is an SPL template
    <spl:foreach var="i" list="a">
        : Element $i has the value '${a[i]}'.
    </spl:foreach>
    <spl:if code="x < y">
        : The variable x is smaller than y.
    </spl:if>
    <spl:else>
        : The variable x is not smaller than y.
    </spl:else>
    <?spl for (var i=0; i<10; i++) { ?>
        : This is line number $i.
    <?spl } ?>
</foobar>);
```



Localization

Introduction

Language Features

- Operators
- Variables (1/2)
- Variables (1/2)
- OO Programming
- Advanced Functions
- String Substitutions
- Templates
- Localization
- Regular Expressions

Module Features

References

The string prefix `_` or `_textdomain_` can be used to localize a string:

```
var a = "tiny", b = "test";  
debug _"This is a $a localization $b.";
```

The dollar substitutions are automatically replaced by placeholders which are later substituted after the translation has been looked up. The special compiler mode `-XN` can be used to produce dummy C file for `xgettext`:

```
$ splrun -XN demo.spl  
/* Dummy C file for xgettext */  
gettext("This is a {0} localization {1}.");
```



Regular Expressions

Introduction

Language Features

- Operators
- Variables (1/2)
- Variables (1/2)
- OO Programming
- Advanced Functions
- String Substitutions
- Templates
- Localization

● Regular Expressions

Module Features

References

- SPL is using PCRE for regular expressions.

- All the regex features of Perl are also available in SPL, and more. Some examples:

```
"the answer is 42" =~ /( ?P<answer>\d+)/I;  
debug "What is the question that leads to $answer?";
```

```
foreach[] word ("this is a test" =~ /\S+/Ag)  
    debug word =~ s/[aeiou]/<$0>/Rg;
```

```
var text = "A = #A, B = #B, C = #C, D = #D";  
debug text =~ e/#(.)/Rg ord($1);
```




[Introduction](#)

[Language Features](#)

[Module Features](#)

- XML
- Qt/KDE
- CGI and WebSPL
- WSF, W2T and WebDebug
- More Demos...

[References](#)

Module Features



XML

[Introduction](#)

[Language Features](#)

[Module Features](#)

● XML

● Qt/KDE

● CGI and WebSPL

● WSF, W2T and WebDebug

● More Demos...

[References](#)

```
load "xml";
```

```
load "file";
```

```
var x = xml_parse(file_read("example_xml.xml"));
```

```
foreach[] p (x["//person"].nodes)  
    debug "${p["name"]} is ${p["age"]} years old.";
```

```
delete x["//person[age < 26]"];
```

```
foreach[] p (x["//person"].nodes)  
    debug "Person '${p["@id"]}' is >25 years old.";
```

```
write(xml_dump(x));
```



Qt/KDE

[Introduction](#)

[Language Features](#)

Module Features

- XML
- **Qt/KDE**
- CGI and WebSPL
- WSF, W2T and WebDebug
- More Demos...

[References](#)

```
load "kde";
qt_kdeinit("simplebrowser", "HTML Browser", "1.0");

var kapp = new qt.KApplication();
var win = new qt.QVBox();
var browser = new qt.KHTMLPart(win);

qt_signal_callback(browser.browserExtension(),
    "openURLRequest(const KURL&,const KParts::URLArgs&
    function(url) { browser.openURL(new qt.KURL(url));

var url = "file:/opt/postgresql/doc/html/index.html"
browser.openURL(new qt.KURL(url));

win.resize(640, 480);
win.show();
kapp.setMainWidget(win);
kapp.exec();
```



Introduction

Module Features

- ## References

- ```
write("That's it!\n");
```



# WSF, W2T and WebDebug

[Introduction](#)

[Language Features](#)

[Module Features](#)

- XML
- Qt/KDE
- CGI and WebSPL
- **WSF, W2T and WebDebug**
- More Demos...

[References](#)

- WSF and W2T are two application development frameworks based on WebSPL.
- Both are ment for web applications, not dynamic web pages.
- Devellopping dynamic web pages can be easily done directly with WebSPL.
- WebDebug is a little web-based debugger that can be used in any WebSPL programs.
- **Demos:** SPLCMS, w2tdemo\_\*.webspl, SPL Poker



# More Demos...

[Introduction](#)

[Language Features](#)

[Module Features](#)

- XML
- Qt/KDE
- CGI and WebSPL
- WSF, W2T and WebDebug
- **More Demos...**

[References](#)

If there is still some time left..

- **Optimizer example**
- **SDL Bindings (Pong)**
- **SPL Ticket**
- **Quick review of the manual**
- **QCake Demo**



[Introduction](#)

[Language Features](#)

[Module Features](#)

[References](#)

● [URLs](#)

# References



# URLs

Introduction

Language Features

Module Features

References

● URLs

- SPL Homepage:  
<http://www.clifford.at/spl/>
- Clifford Wolf:  
<http://www.clifford.at/>