Pan's Coolest Features

(WP4 June 2002 Workshop)

Lionel Cons

21/06/2002

Different Output Styles

```
object template test;

"/flag" = true;
"/nums" = list(7, 3.14);
```

```
$ pan --stdout --xmlstyle=piotr test.tpl
```

```
<?xml version="1.0" encoding="utf-8"?>
<resource name="profile" type="nlist">
  <property name="flag"
    type="boolean">true</property>
  <resource name="nums" type="list">
    <property name="0"
      type="long">7</property>
    <property name="1"
      type="double">3.14</property>
  </resource>
</resource>
```

Different Output Styles

```
$ pan -s -x=tim test.tpl
<profile cfg:type="resource">
  <flag cfg:type="boolean">true</flag>
  <nums cfg:type="resource">
    <long cfg:type="long">7</long>
    <double cfg:type="double">3.14</double>
  </nums>
</profile>
```

```
$ pan -s -x=pan test.tpl
<nlist name="profile">
  <boolean name="flag">true</boolean>
  <list name="nums">
    <long name="0">7</long>
    <double name="1">3.14</double>
  </list>
</nlist>
```

Arbitrary Strings

```
"/s1" = "<foo>ok&ok</foo>";
"/s2" = "\x01\x6f\x6b\x21";
```

```
<nlist name="profile">
  <string name="s1">
    &lt;foo&gt;ok&amp;ok&lt;/foo&gt;
  </string>
  <string name="s2" encoding="base64">
    AW9rIQ==</string>
</nlist>
```

Here-document Syntax

```
"/system/services" = <<EOT;
tcpmux    1/tcp
echo      7/tcp
echo      7/udp
discard   9/tcp sink null
discard   9/udp sink null
EOT
```

```
"/system/ssh_host_key" =
  base64_decode(<<EOK);
    H4sIAOwLyDwAA02PQQ7DMAgE731FX9BT1f8Q
    Z52iYhthEiW/r2SitCdmxCK0E3W8no+36n2G
    GGLobbUUX7pT+pxkXJc/5Bx5p0ki7Cgq5Kcc
    GrCR8PzruUfP2xfJgVqHCgEAAA==
EOK
```

Automatic Compression

```
"/big" = <<EOT;
12345678901234567890123456789012345678901234567890
 ... [ 250 other lines like these ] ...
12345678901234567890123456789012345678901234567890
EOT


<nlist name="profile">
  <string name="big" encoding="base64,gzip">
    H4sIAAAAAAAAAA+3MqQEAIBDAMM80HD/7L8YIiiL
    NxNU20Puba59ZIVMnMGAwGg8FgMBgMBoPBYDAY
    DAaDwWAwGAwGg8FgMBgMBoPBYD7MAy83ufHwJw
    AA
  </string>
</nlist>
```

User Defined Functions

```
define function facto = {
  if (argc != 1)
    error("facto(): wrong number of args");
  if (!is_long(argv[0]))
    error("facto(): arg is not a long");
  if (argv[0] < 0)
    error("facto(): arg is negative");
  if (argv[0] < 2)
    return(1);
  return(argv[0] * facto(argv[0] - 1));
};


"/test" = facto(6); # will be 720
```

Perl Compatible Regexp

```
define function valid_ipv4 = {
   result = matches(argv[0],
      '^(\d+)\.(\d+)\.(\d+)\.(\d+)$');
   if (length(result) == 0)
      error("invalid string");
   i = 1;
   while (i <= 4) {
      x = to_long(result[i]);
      if (x > 255)
         error("too big: " + result[i]);
      i = i + 1;
   };
   return(true);
};
```

PCRE & Validation

```
valid "/greeting" =
  match(self, '^(?i)hello$');

define type ipv4 =
  string with valid_ipv4(self);

define type foo = string with
  match(self, '((\w)\2).+\1');

valid "/system/swap" =
  self >= 2 * value("/system/ram");
```

Types

```
define type ubyte = long(0..255);

define type udouble = double(0..);

define type str16 = string(..16);

define type list4 = element[4];
```

```
type "/system/mail/sendmail_cf" = fetch;
"/system/mail/sendmail_cf" =
   "http://cern.ch/cfg/cms/sendmail.cf";
```

Types (cont'd)

```
define type srvname = string
  with match(self, '^(?:\w+-)*\w+$');

define type service = {
  "name"      : srvname
  "port"      : long(0..65535)
  "protocol"  : string with
                match(self, '^(tcp|udp)$')
  "aliases"   ? srvname[]
};

type "/system/services" = service[];

"/system/services/0" = nlist(
  "name",       "ssh",
  "port",         22,
  "protocol",   "tcp",
);
```

Types (cont'd)

```
define type partition = {
   "size" : long
   "type" : string
   "id"   : string
};

define type filesystem = {
   "partition" : link
   "mntpt"     : string
   "type"      : string
};

type "/system/partitions" = partition{};
type "/system/filesystems" = filesystem{};

"/system/filesystems/root/mntpt" =
   "/";
"/system/filesystems/root/partition" =
   "/system/partitions/hda5";
```

Types (cont'd)

```
define type CONTROLLER = {
   "ProtocolSupported" : string
   "MaxNumberControlled" : long
};

define type SERIAL_CONTROLLER = {
   include CONTROLLER
   "MaxBaudRate" : long
};

define type SCSI_CONTROLLER = {
   include CONTROLLER
   "MaxDataWidth" : long
   "MaxTransferRate" : long
};
```

Types (cont'd)

```
define type srvname = string
  with match(self, '^[\w-]+$')
  description "internet service name";

define type service = {
  "name"    : srvname
              description "primary name"
  "port"    : long(0..65535)
              description "port used"
  "proto"   : string
              with match(self, '^(tcp|udp)$')
              descro "protocol (udp or tcp)"
  "aliases" ? srvname[]
              descro "list of alias names"
} descro "internet service definition";
```

```
$ pan --doc=latex types.tpl > schema.tex
```

Possible Additions

Context handling via XML multiplexing

Global variables

Inter-objects links (for "spanning maps")