Designing a Development Environment to Support Creation of Standard-Compliant Applications

Denis Silakov

Institute for System Programming, RAS
http://isprats.ru/
Linux Verification Center
http://linuxtesting.org/

SYRCoSE 2009. 28-29 May, 2009, Moscow
Using Standards

Developers

Standard

Implementation 1

Implementation 2

Implementation N
Develop for Standard?

Implementation

Std Part

Non-Std Part

Build Toolchain
Possible Approaches

- “Careful” development
- Ideal implementation
  - mobile device emulators
  - “sample implementations” (LSB, OpenGL)
- Systematic testing
- Restricted environment inside the real implementation
  - `-std` option of gcc
  - LSB Development Environment (LSB SDK)
**Idea**
Take any compliant implementation and drop non-standardized items

**Challenge**
- Standard evolves
- Implementation evolves

How to reflect the changes in the environment?
Database Driven Approach

Existing Implementation

Data Collection Tool

Database

Generator

Adopted Implementation

Workgroup
Database Design

What to store?
- Everything that depends on the standard
- Data used in more than one tool

Configuration flags
- Is particular entry is included in the standard?

Item interdependencies
- Dependencies in the real world → foreign keys in the database
Skeleton + Data from the DB = Generated Environment
## Configuration Flags

<table>
<thead>
<tr>
<th>Function</th>
<th>Header</th>
<th>Incuded?</th>
</tr>
</thead>
<tbody>
<tr>
<td>gets</td>
<td>stdio.h</td>
<td>No</td>
</tr>
<tr>
<td>fgets</td>
<td>stdio.h</td>
<td>Yes</td>
</tr>
<tr>
<td>puts</td>
<td>stdio.h</td>
<td>Yes</td>
</tr>
<tr>
<td>fputs</td>
<td>stdio.h</td>
<td>Yes</td>
</tr>
</tbody>
</table>

```c
/* begin stdio.h */
extern int puts (const char *);
extern int fputs (const char *, FILE *);
extern char *fgets (char *, int, FILE *);
/* end stdio.h */
```
Temporal Database

- Time intervals – *appeared in v1, dropped in v2*
- Extra fields for extra status – *optional in v3*

Specifics

- Discrete time values
- *Valid* time only
- Small number of possible values
- Correlations in time intervals for interdependent items
### Time Intervals

<table>
<thead>
<tr>
<th>Function</th>
<th>Header</th>
<th>Appeared</th>
<th>Withdrawn</th>
</tr>
</thead>
<tbody>
<tr>
<td>gets</td>
<td>stdio.h</td>
<td>1.0</td>
<td>1.2</td>
</tr>
<tr>
<td>fgets</td>
<td>stdio.h</td>
<td>1.0</td>
<td>NULL</td>
</tr>
<tr>
<td>puts</td>
<td>stdio.h</td>
<td>2.0</td>
<td>NULL</td>
</tr>
<tr>
<td>fputs</td>
<td>stdio.h</td>
<td>2.0</td>
<td>NULL</td>
</tr>
</tbody>
</table>
- Header files *(generated)*
- Stub Libraries *(generated)*
- Compiler wrapper – forces system compiler to use LSB headers and link against LSB libraries
Headers – driven by LSB_VERSION constant

```c
#if LSB_VERSION >= 10
#if LSB_VERSION < 20
  extern char *gets (char *);
#endif
extern char *fgets (char *, int, FILE *);
#endif
```

Libraries – separate file for every LSB version
### Generated Code vs Generator Code

<table>
<thead>
<tr>
<th></th>
<th>Generators</th>
<th>Generated Code</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SLOC</strong></td>
<td>2,500</td>
<td>156,300</td>
</tr>
<tr>
<td><strong>Development effort estimate</strong></td>
<td>0.6 person-years (7 person-months)</td>
<td>39 person-years</td>
</tr>
<tr>
<td><strong>Total estimated cost to develop</strong></td>
<td>$70,000</td>
<td>$5,250,000</td>
</tr>
</tbody>
</table>
Approach Advantages

- Create environment not from scratch
- Consider only important parts of implementation (database schema = abstraction model)
- Automated synchronization (in case of automated tools)
- Support for several versions of the standard
URLs & Contacts

- LSB Infrastructure Project
  [http://ispras.linuxfoundation.org](http://ispras.linuxfoundation.org)
- LSB at the Linux Development Network
  [http://ldn.linuxfoundation.org/lsb](http://ldn.linuxfoundation.org/lsb)
- Denis Silakov
  [silakov@ispras.ru](mailto:silakov@ispras.ru)